Journal of Education and Educational Research
ISSN: 2957-9465 | Vol. 16, No. 2, 2025

Curriculum Reform of Compiler Principles in the Context of New Engineering Disciplines

Lulu Rao

Anhui Xinhua University, Hefei, Anhui, China

Abstract: The proposal of the "New Engineering Disciplines (NED)" concept marks a new phase in China's higher engineering education, characterized by interdisciplinary integration, technology-driven development, and innovation-oriented training. As a core foundational course for computer-related majors, Compiler Principles plays an irreplaceable role in fostering students' systematic thinking, engineering practice capabilities, and cognitive understanding of underlying technologies. However, traditional teaching of Compiler Principles faces prominent issues such as overly theoretical content, monotonous teaching methods, fragmented practical training, and a lack of Ideological and Political Education (IPE). These problems make it difficult to meet the requirements of NED for cultivating high-quality innovative talents. Based on the teaching practice of Compiler Principles at Anhui Xinhua University, this paper starts with the reform background and current situation, clarifies the three-dimensional objectives of the curriculum reform, systematically elaborates on the core paths including the optimization of teaching content, innovation of teaching methods, and strengthening of practical links, and analyzes the key problems to be solved and the distinctive features of the reform.

Keywords: New Engineering Disciplines; Compiler Principles; Curriculum Reform; Ideological and Political Education (IPE); Practical Teaching.

1. Introduction

Driven by the dual impetus of the global technological revolution and industrial transformation, the country has put forward higher requirements for engineering and technical talents in terms of innovation capability, practical ability, and interdisciplinary literacy. At the Seminar on the Development of Higher Engineering Education held by Fudan University, the concept of "New Engineering Disciplines (NED)" was formally proposed for the first time. Its core essence lies in "interdisciplinary integration, emerging technology leadership, innovative talent cultivation, educational model reform, and deepening the integration of production and education." It is not only an in-depth reflection on traditional engineering education but also a proactive response to national strategic needs and industrial development trends [1]. As shown in Figure 1, the core objectives of NED talent cultivation focus on fostering high-quality innovative engineering professionals. As a core foundational course for majors such as Computer Science and Technology and Software Engineering, Compiler Principles covers key content including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization. It serves as a bridge connecting the underlying hardware and upper-layer applications of computers, playing a vital role in cultivating students' logical thinking, system design capabilities, and technological innovation awareness.

However, the current teaching of Compiler Principles faces numerous practical dilemmas: the teaching content focuses overly on theoretical derivation, being disconnected from the application of emerging technologies and actual industrial needs; the teaching method is dominated by classroom lectures, lacking interactivity and innovation, resulting in low student participation; the design of practical links is fragmented, failing to form a systematic training system, which leads to insufficient cultivation of students' practical abilities; the lack of Ideological and Political Education (IPE) elements makes it difficult to achieve the organic unity of knowledge impartment and value guidance [2]. These problems have seriously restricted the improvement of the course teaching quality, running counter to the core goal of NED to "cultivate high-quality innovative engineering and technical talents."

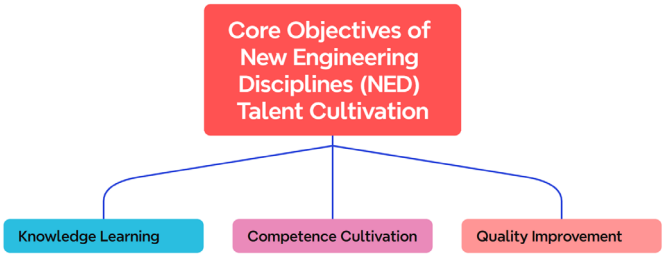

Figure 1. Core Objectives of New Engineering Disciplines (NED) Talent Cultivation

The Compiler Principles course is offered to majors such as Computer Science and Technology and Software Engineering. It boasts a teaching team composed of teachers with rich teaching experience and outstanding research capabilities, having accumulated a number of teaching achievements including the construction of Ideological and Political Education (IPE) demonstration courses, MOOC development, and the establishment of practical teaching platforms. Based on this, combined with the requirements of New Engineering Disciplines (NED) construction, this paper conducts in-depth research on the teaching reform of the Compiler Principles course, exploring the reform path of "content optimization, method innovation, practice strengthening, and IPE integration." It aims to provide a feasible plan for improving the course teaching quality and cultivating high-quality talents that meet industrial needs.

2. Core Problems in Traditional Teaching

(1) Disconnection between Teaching Content and the Needs of the Times

The teaching content of traditional Compiler Principles courses focuses mainly on classical theories and algorithmic derivations, which are overly abstract and theoretical, leading to a disconnection from practical industrial applications and the development of emerging technologies. On one hand, the update of course content is slow, with insufficient coverage of cutting-edge technologies such as parallel compilation and AI-assisted compilation, failing to meet the requirement of technological advancement in New Engineering Disciplines (NED). On the other hand, interdisciplinary integration is inadequate, as the course fails to fully integrate knowledge from related disciplines such as computer architecture, software engineering, and operating systems, making it difficult for students to establish a complete knowledge system [8]. Additionally, the lack of Ideological and Political Education (IPE) elements restricts teaching to the imparting of professional knowledge, neglecting the cultivation of students' family and country feelings, professional ethics, and social responsibility, which is inconsistent with the talent training goal of "cultivating individuals with both moral integrity and professional competence."

(2) Monotonous and Rigid Teaching Methods

Traditional Compiler Principles teaching is dominated by the "teacher-centered lecture + student listening" model, where teachers occupy a dominant position in the classroom, and students passively receive knowledge, lacking effective interaction and participation links. This monotonous teaching method is difficult to stimulate students' learning interest and initiative, resulting in superficial understanding and weak mastery of abstract Compiler Principles knowledge. Although some courses include experimental sessions, most are confirmatory experiments where students mechanically complete operational tasks, failing to effectively cultivate their innovative thinking and problem-solving abilities [3]. Meanwhile, information-based teaching tools are not fully utilized, and the teaching model lacks flexibility and pertinence, making it difficult to adapt to the learning characteristics of students in the new era.

(3) Weak Practical Teaching Links

Practical teaching is a core component of the Compiler Principles course, but traditional practical teaching faces numerous problems: first, the experimental content is fragmented. Experimental projects such as lexical analysis, syntax analysis, and semantic analysis are independent of each other, lacking coherence and systematicness, which makes it difficult for students to form an overall understanding of the compilation process; second, the experimental projects lack innovation, mostly repeating classic cases in textbooks, with insufficient comprehensive projects combined with actual engineering scenarios; third, the practical evaluation system is single, relying only on experimental reports or code submissions as evaluation criteria, failing to fully reflect students' practical abilities and innovative levels [4].

(4) Disconnection between Talent Cultivation and Industrial Needs

The training goal of traditional Compiler Principles courses focuses on the mastery of theoretical knowledge, neglecting the industrial requirements for talents' practical abilities, innovation capabilities, and interdisciplinary literacy. The course content is disconnected from actual enterprise application scenarios, requiring students to spend a long time adapting to work positions after graduation; there is a lack of effective industry-university-research cooperation mechanisms between schools and enterprises, and students have insufficient opportunities for practical training in real engineering environments, resulting in a gap between talent cultivation quality and industrial needs [4].

3. Curriculum Reform Objectives of Compiler Principles in the Context of New Engineering Disciplines

Aligning with the core requirements of New Engineering Disciplines (NED)—"interdisciplinary integration, innovation leadership, practice orientation, and deepening the integration of production and education"—the curriculum reform of Compiler Principles establishes a trinity of training objectives: "Knowledge and Skills, Competence and Quality, and Curriculum-Industry Connection." It aims to achieve a transformation from "knowledge impartment" to "competence cultivation" and "value shaping."

(1) Knowledge and Skills Dimension

Students are required to deeply understand the core concepts, basic principles, and key algorithms of Compiler Principles, and proficiently master the implementation methods of various links including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization. They should also understand the compilation characteristics of different programming languages, master the usage skills of mainstream compilation tools, and possess the ability to develop small compilers or conduct basic maintenance and optimization of existing compilers—laying a solid technical foundation for engaging in software development, system design, and other related work. Additionally, students need to grasp the application scenarios and core methods of compilation technology in fields such as artificial intelligence (AI) and big data to broaden their knowledge horizons.

(2) Competence and Quality Dimension

Emphasis is placed on cultivating students' systematic thinking ability, enabling them to grasp the overall relationship between the compilation system and components such as computer hardware and operating systems. It is also crucial to stimulate students' innovative thinking, encouraging them to explore new compilation algorithms and optimization strategies. Through practical projects, students' hands-on practical ability and complex problem-solving ability will be enhanced, including skills in requirement analysis, architecture design, code implementation, and debugging optimization. Moreover, students' teamwork spirit and communication skills will be fostered to meet the collaborative needs in engineering practice. Meanwhile, students' professional ethics and social responsibility will be strengthened to shape correct values.

(3) Curriculum-Industry Connection Dimension

The curriculum content will be adjusted to adapt to the needs of emerging industries for compilation technology, integrating compilation application cases from fields such as AI and cloud computing into teaching. Industry-university-

research cooperation mechanisms will be established to strengthen exchanges between schools and enterprises, providing students with real engineering practice opportunities. The goal is to cultivate high-quality talents that meet industrial requirements, shorten the transition period from school to the workplace, and make curriculum teaching better serve industrial development and economic construction [5].

4. Core Content and Implementation Paths of Compiler Principles Curriculum Reform in the Context of New Engineering Disciplines

The teaching reform of Compiler Principles advances mainly from three dimensions: first, optimizing teaching content by integrating Ideological and Political Education (IPE) elements; second, innovating teaching methods through the adoption of two forms—Rain Classroom and flipped classroom; third, strengthening practical links through the implementation of project-based teaching, as shown in Figure 2.

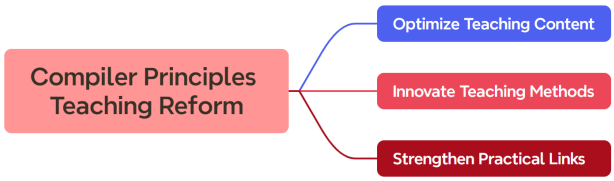

Figure 2. Teaching Reform Approaches of Compiler Principles

(1) Optimizing Teaching Content: IPE Integration and Cutting-Edge Expansion

1. Integrating IPE Elements to Achieve Value Guidance

The integration of Ideological and Political Education (IPE) aims to embed ideological and political education throughout the entire teaching process, realizing the organic unity of knowledge impartment and value guidance. On one hand, major achievements of China in the field of compilation technology are explored to stimulate students' national pride and patriotic feelings. On the other hand, IPE concepts are organically combined with professional knowledge. For example, when explaining syntax trees, an analogy is drawn to the rigorous structure and hierarchical beauty of traditional Chinese architecture, guiding students to appreciate the "craftsmanship spirit of striving for excellence." When discussing compiler optimization, the green development concept of "efficiency and energy conservation" is emphasized to cultivate students' social responsibility [6]. Meanwhile, drawing on teaching experience from IT professional ethics courses, content such as code standards and intellectual property protection is integrated into Compiler Principles teaching to strengthen students' professional literacy.

2. Updating the Knowledge System to Align with Technological Frontiers

Following the development trend of compilation technology, teaching content is updated in a timely manner: first, cutting-edge technologies such as parallel compilation, distributed compilation, and AI-assisted compilation are introduced, with explanations of their principles and application scenarios; second, interdisciplinary integration is strengthened—combining computer architecture knowledge in the code generation stage and integrating software engineering project management methods into compiler project development to help students establish an interdisciplinary knowledge system; third, practical application cases are added, such as analyzing the working principle of the open-source compiler GCC and introducing application examples of compilation technology in AI and big data fields (e.g., code optimization methods in deep learning frameworks), making teaching content more closely aligned with industrial reality [8].

(2) Innovating Teaching Methods: Intelligent Tools and Interactive Modes

1. Combining Rain Classroom with Flipped Classroom to Enhance Teaching Interaction

Leveraging information-based teaching tools, a hybrid teaching model of "Rain Classroom + flipped classroom" is constructed. Before class, teachers push preview materials (e.g., videos on core concepts of Compiler Principles, articles on cutting-edge technologies) and assessment tasks (e.g., multiple-choice questions, thinking questions) via Rain Classroom. Students conduct independent learning and complete tasks, while teachers accurately identify students' knowledge weaknesses based on Rain Classroom's feedback data to design targeted in-class teaching content. During class, teachers focus on explaining key and difficult points, pushing case analysis questions and code practical exercises in real time through Rain Classroom. Students answer immediately on their mobile devices, and teachers quickly grasp students' understanding through data statistics for targeted explanations. The bullet comment function of Rain Classroom is used to encourage students to ask questions and express opinions in real time, creating an active classroom atmosphere. Meanwhile, visualized materials such as compilation process animations and compiler operation videos are pushed to intuitively present abstract knowledge, helping students understand complex concepts [7]. After class, extended tasks (e.g., open-source compiler source code analysis, small-scale technical discussions) are assigned via Rain Classroom to extend learning effects.

2. Adopting Case-Based Teaching to Strengthen Knowledge Application

Typical cases are selected to run through the teaching process. For example, taking the development of a simple arithmetic language compiler as the main line, knowledge points such as lexical analysis, syntax analysis, and semantic analysis are integrated into case explanations, enabling students to clarify the practical application scenarios of each knowledge point. Meanwhile, actual enterprise project cases (e.g., compiler performance optimization, Domain-Specific Language (DSL) compilation tool development) are introduced to help students understand real industrial needs [8]. In case analysis, students are guided to discuss solutions in groups to cultivate their critical thinking and problem-solving abilities.

(3) Strengthening Practical Links: Project-Driven and System Construction

1. Building a Systematic Practical Teaching System

Breaking the fragmented limitations of traditional practical teaching, a three-level practical teaching system of "basic verification - comprehensive design - innovative expansion" is constructed. The basic verification layer focuses on core knowledge points, setting up basic experiments such as lexical analyzer design and syntax analyzer implementation to consolidate students' theoretical knowledge. The

comprehensive design layer takes complete projects as carriers, requiring students to develop small compilers (e.g., a language compiler supporting basic arithmetic operations and variable assignment) in groups, completing the entire process from requirement analysis, architecture design, and code implementation to testing and optimization, thereby cultivating students' system design capabilities and teamwork spirit. The innovative expansion layer encourages students to carry out innovative practices combined with cutting-edge technologies (e.g., AI-based compilation optimization, domain-specific compiler development) to stimulate their innovative thinking [9].

2. Implementing Project-Based Teaching to Improve Practical Abilities

Driven by projects, practical teaching is closely integrated with engineering reality. In the comprehensive design layer, "development of a simple arithmetic language compiler" is selected as the core project, requiring students to complete the entire process of lexical analysis (identifying numbers, variables, operators), syntax analysis (constructing syntax trees), semantic analysis (checking variable definition rules), and code generation (generating intermediate code or machine language). During project implementation, students collaborate in groups, adopt software engineering iterative development methods, conduct regular project reports and reviews, and teachers provide full-process guidance and technical support. Through this project, students can integrate scattered knowledge points, establish a comprehensive understanding of the compilation system, and exercise engineering practical abilities such as requirement analysis, architecture design, code implementation, and debugging optimization [10].

3. Improving Practical Teaching Support Conditions

Relying on the university's laboratories, a practical teaching platform for Compiler Principles is built, providing GCC compilation tools, virtual machines, distributed development environments, and other supporting resources. Students are organized to participate in compilation-related project development (e.g., compiler performance optimization, customized compilation tool development) to accumulate practical experience in real engineering environments. The practical evaluation system is improved, adopting a "process evaluation + outcome evaluation" approach. Process evaluation includes project progress, teamwork, and code standards, while outcome evaluation covers function implementation, performance optimization, and innovative points, comprehensively reflecting students' practical abilities and innovative levels [11].

5. Key Problems to Be Solved by the Reform

(1) Teaching Content Level

Addressing the disconnection between theory and practical application: By introducing practical cases and conducting project-based practical teaching, students can apply theoretical knowledge to actual compilation tasks, improving their practical abilities and problem-solving skills.

Solving the lag in content update: Timely integrating cutting-edge technologies such as parallel compilation and AI-assisted compilation, updating textbook content and teaching cases to ensure the timeliness of the curriculum.

Overcoming insufficient interdisciplinary integration: Strengthening knowledge integration with disciplines such as computer architecture and software engineering, and designing interdisciplinary practical projects to cultivate students' comprehensive literacy.

(2) Teaching Method Level

Resolving the monotony of teaching methods: Adopting diversified methods such as hybrid teaching, case-based teaching, and project-driven teaching to stimulate students' learning interest and initiative, and improve classroom participation.

Tackling weak practical teaching: Building a systematic practical teaching system, designing comprehensive and innovative practical projects, improving practical teaching platforms and evaluation systems, and strengthening the cultivation of practical abilities.

(3) Student Competence Cultivation Level

Addressing insufficient systematic thinking ability: Guiding students to grasp the structure and process of the compilation system as a whole through complete compiler development projects, cultivating their systematic thinking.

Solving inadequate innovation capacity training: Stimulating students' innovative thinking and exploratory spirit through innovative expansion projects and encouraging them to explore the application of cutting-edge technologies.

6. Main Features of the Reform

(1) In-Depth Integration of IPE and Professional Knowledge

Breaking the limitation of "formalistic integration" in traditional IPE, this reform organically combines IPE elements with Compiler Principles professional knowledge. Through case analogy, professional ethics infiltration, and other methods, it realizes the synchronous advancement of value guidance and knowledge impartment, cultivating high-quality talents with both moral integrity and professional competence.

(2) Intelligent Innovation of Teaching Modes

Making full use of intelligent teaching tools such as Rain Classroom, a full-process interactive teaching model of "pre-class - in-class - post-class" is constructed. It achieves precise and personalized teaching processes, solves the problems of insufficient interaction and weak pertinence in traditional teaching, and improves teaching effects.

(3) Systematic and Improved Practical System

The three-level practical teaching system of "basic verification - comprehensive design - innovative expansion" is built, with project-based teaching as the core. It realizes the systematization, hierarchicalization, and innovation of practical content, breaks the fragmented limitations of traditional practical teaching, and comprehensively improves students' engineering practical abilities and innovation capabilities.

7. Conclusion

In the context of New Engineering Disciplines (NED), the curriculum reform of Compiler Principles is an inevitable choice to adapt to the development trend of engineering education and meet industrial talent needs. Based on the teaching practice of Anhui Xinhua University, this paper constructs a reform system of "IPE integration, intelligent teaching, project-driven, and industry-university-research integration" from three core dimensions: teaching content, teaching methods, and practical links. It clarifies the trinity of training objectives—"Knowledge and Skills, Competence

and Quality, and Curriculum-Industry Connection"—aiming to solve prominent problems in traditional teaching such as disconnected content, monotonous methods, and weak practical links.

Through the implementation of the reform, it is expected to significantly improve students' theoretical level, practical abilities, and innovative thinking, enhance the quality of curriculum teaching, and cultivate high-quality engineering and technical talents that meet NED requirements. Meanwhile, the experience and model formed by the reform can provide useful references for the reform of other computer professional courses, the development of interdisciplinary education, and the deepening of university-enterprise cooperation, with important practical value and promotion significance. In the future, we will continuously track the reform implementation effect, dynamically optimize the reform plan according to technological development and industrial needs, and constantly improve the teaching quality and talent training level of the Compiler Principles course.

Acknowledgments

University-Level Quality Engineering Project of Anhui Xinhua University (2024) Project Name: Teaching Reform of Compiler Principles Course Under the Background of New Engineering Disciplines Project No.: 2024jy039.

University-Level Scientific Research Project of Anhui Xinhua University (2024) Project Name: Research on Classification and Recognition of Weld Defects Based on Feature Fusion Project No.: 2024zr019.

References

[1] FU Yangwu, HU Chuanbo, and CHEN Shuhong. Teaching Reform and Practice of Physical Chemistry Course in Local Universities under the Background of "New Engineering Disciplines" [J]. University Chemistry, 2024, 10(23): 45-46.

[2] CUI Guangyu. Research on the Current Situation and Innovation of Compiler Principles Teaching [J]. Wireless Internet Technology, 2017, (24): 81-82.

[3] GAO Xueyao and ZHANG Chunxiang. Research on Teaching Reform of Compiler Principles under the Background of New Engineering Disciplines [J]. Journal of Higher Education, 2024, 10(1): 35-38.

[4] JIANG Yu, YANG Jianzhong, QU Wei, et al. Promoting the Cultivation of Computer-Related Professionals through Disciplinary Competitions under the Background of New Engineering Disciplines [J]. Modern Business Trade Industry, 2021, 42(17): 67-68.

[5] YANG Xu. Analysis of Teaching Reform of Compiler Principles Course Based on Blended Teaching under the Background of New Engineering Disciplines [J]. Computer Products & Circulation, 2019, (12): 225.

[6] HUANG Xianying and CAO Qiong. Rethinking and Implementation of the Relocation of "Compiler Principles" under the Background of New Engineering Disciplines [J]. Fujian Computer, 2017, 33(9): 78, 80.

[7] CHEN Yiren and WANG Yibin. Teaching Design and Practice of Compiler Principles MOOC [J]. Fujian Computer, 2020, 36(1): 44-45.

[8] YAO Zhun, WANG Ling, ZHANG Hao, et al. Reform and Exploration of Big Data Analysis Course under the Background of New Engineering Disciplines [J]. Journal of Higher Education, 2024, 10(30): 144-147.

[9] XIONG Ying, LIU Maofu, and MAO Xuesong. Construction of Outcome-Based Experimental Assessment and Evaluation System for Compiler Principles [J]. Computer Education, 2024, (7): 213-218.

[10] MENG Wenwen, ZHANG Xin, and CHENG Zhi. Analysis of Smart Classroom Teaching Practice of Compiler Principles Course: A Case Study of Hefei University [J]. Computer Knowledge and Technology, 2023, 19(19): 154-156.

[11] HE Xi, CHEN Jia, and NONG Jian. Discussion on the Teaching Reform of "Compiler Principles" Course [J]. Journal of Wuzhou University, 2021, 31(6): 85-89.