Transforming Programming Education: Paradigm Shifts and Curricular Reconstruction in the AI Era

Xingna Hou[1,2], Jianxin Hu [3], Zhiping Zhou [2] and Shouhong Chen[1*]

[1] Guangxi Key Laboratory of Automatic Detecting Technology and Instruments, School of Electronic Engineering & Automation, Guilin University of Electronic Technology, Guilin 541004, China, *cshgl@guet.edu.cn

[2] School of Architecture and Transportation Engineering, Guilin University of Electronic Technology, Guilin, 541004, China

[3] School of Law, Guilin University of Electronic Technology, Guilin,541004, China

*cshgl@guet.edu.cn

Abstract. With the widespread application of generative AI technology in the field of programming, college programming courses are facing a core controversy regarding their 'retention or abolition' and 'reconstruction'. The traditional teaching model centered on 'writing code' has difficulty meeting the current era's demand for innovative programming talents. Drawing on constructivist learning theory and Bloom's Taxonomy of Cognitive Objectives, this paper systematically examines the current state and challenges of university programming education in the AI era. It proposes a curriculum reconstruction plan centered on two cores: 'basic capabilities' and 'AI collaborative capabilities.' The plan covers three dimensions—curriculum content, teaching models, and evaluation systems—with the goal of offering practical implementation strategies for reforming university programming courses in the AI age. Ultimately, it aims to support universities in shifting their focus from training 'pure code writers' to developing 'human-machine collaborative innovative developers.

Keywords: Generative AI; Programming Curriculum Reconstruction; Human-Machine Collaboration

1 Introduction

In recent years, the rise of generative AI tools like ChatGPT and DeepSeek has transformed traditional programming models, particularly in areas such as code generation, debugging optimization, and requirement analysis[1]. This shift has sparked widespread debate in university programming education. One perspective suggests that generative AI could replace manual coding, rendering university programming courses unnecessary. Another viewpoint maintains that the core value of programming courses lies in fostering students' logical thinking and problem-solving skills, arguing that AI should serve as an educational aid to enhance learning rather than replace traditional course content[2-4].

© The Author(s) 2026

A. T. Patanasorn et al. (eds.), *Proceedings of the 3rd International Conference on Educational Development and Social Sciences (EDSS 2026)*, Advances in Social Science, Education and Humanities Research 1010,
https://doi.org/10.2991/978-2-38476-569-0_28

Based on this, this paper, rooted in the principle that 'technology empowers education rather than replaces it,' combines constructivist learning theory and Bloom's Taxonomy of Cognitive Objectives to systematically investigate the reconstruction path of university programming courses in the AI era. By defining the course's core ability framework, it presents a practical teaching reform plan to address the question of where programming courses should evolve. This work offers theoretical support and practical guidance for universities aiming to cultivate innovative programming talent adaptable to the digital economy's development.

2 Current Status and Challenges of Programming Education in Universities in the AI Era

2.1 Traditional Programming Course Teaching Faces Significant Challenges

Niversity programming courses currently face two main issues: emphasizing syntax over logical thinking and prioritizing theory at the expense of practice. In terms of content, courses typically focus heavily on explaining programming language syntax rules, with curricula often disconnected from real-world industry development needs. Regarding teaching methods, the common approach is a one-way knowledge transfer—teacher-led instruction followed by passive student exercises—with insufficient interactive and inquiry-based learning elements. Assessment methods still rely primarily on traditional written exams and final code submissions for summative evaluation, which fail to adequately measure students' problem-analysis and logical-reasoning skills. These rigid teaching practices lead to students who, despite mastering basic syntax, often lack the ability to break down complex real-world problems and the systematic thinking needed for architectural design.

2.2 Limitations of Existing Educational Reform Attempts

In response to the impact of AI technologies, some universities have already initiated exploratory reforms in programming education[5-7]. However, these efforts are often characterized by extreme or fragmented approaches. Specifically, one common strategy is to impose a blanket ban on AI tools,ultimately leading to a disconnect between curricular content and real-world industry demands. Another approach involves uncritically embracing AI technologies, integrating AI tools as primary instructional means without establishing clear guidelines or boundaries for their use. This risks undermining the development of students' core programming competencies. Furthermore, many reform measures remain confined to isolated aspects of teaching. As a result, these fragmented initiatives fail to address the fundamental challenges facing programming education in the AI era.

3 Reconstruction Scheme of Programming Courses Integrating AI

3.1 Course Reconstruction Objective: Theory as Foundation, Competence as Core

Constructivist learning theory suggests that knowledge is actively built by learners through contextual interactions, rather than passively absorbed. Thus, programming instruction in the AI era should design a learning framework of 'problem context - human-machine interaction - collaborative inquiry,' encouraging students to actively engage in the full process of knowledge construction. Bloom's Taxonomy of Cognitive Objectives categorizes cognitive processes into six levels. In the AI era, programming education should move beyond basic cognitive levels like 'remembering' and 'understanding,' while focusing on developing higher-order cognitive skills such as 'analyzing,' 'evaluating,' and 'creating' to elevate students' cognitive capabilities. Drawing on these two theories and aligning with real-world industry needs, this paper proposes a core competency system for university programming education in the AI era. It transcends the traditional single-focus on 'code writing' and adopts a comprehensive competency model that includes 'basic programming ability + critical thinking + prompt engineering ability + system architecture ability + technical ethics literacy.'

First, basic programming skills form the foundation for human-machine collaboration. These include core competencies like fundamental syntax, data structures, algorithms, and code debugging, much like how a calculator relies on arithmetic operations—AI tools cannot replace the essential value of these foundational abilities. Second, critical thinking is crucial, involving the ability to analyze and assess the logical validity, security, and efficiency of AI-generated code. Third, prompt engineering skills are necessary, which involve crafting precise prompts to guide AI in producing code that meets specific requirements, along with mastering optimization techniques such as chain-of-thought reasoning and role-playing. Fourth, system architecture capabilities are important, encompassing the ability to break down complex real-world problems, design modular components, and integrate systems. Fifth, technical ethical literacy is vital, requiring a clear understanding of AI tool usage boundaries, adherence to academic standards, and the cultivation of 'responsible software development' principles.

3.2 Curriculum Content Reconstruction: Emphasis on Fundamentals and Collaboration

The reconstruction of course content follows the principle of 'strengthening the foundation, collaborative empowerment, and ethical bottom line'.

The basic module preserves core program design fundamentals, such as variables, flow control, functions, data structures, algorithms, and debugging techniques. However, it reduces the emphasis on direct instruction delivery, minimizes focus on 'grammar memorization,' and instead strengthens training in 'logical understanding' and

'problem decomposition.' For instance, in C language program design teaching, mechanical explanations of syntax rules are decreased. Instead, methods like 'implementing classic algorithms' and 'analyzing code debugging cases' are employed to help students grasp the logical principles underlying the syntax.

Secondly, the AI Collaboration Module. New course content titled 'Fundamentals and Application Boundaries of LLMs' has been introduced to explain the working mechanisms, code generation logic, and limitations of large language models, helping students develop a rational understanding of AI tools. A specialized training program on 'Prompt Engineering' has also been established. Through this program, students learn prompt optimization techniques—including chain-of-thought reasoning, example-driven prompting, and role setting—via case studies, and participate in practical tasks focused on 'Precise Prompt Design'. Additionally, a practical session on 'Verification and Optimization of AI-Generated Code' has been added. By comparing AI-generated code with optimal solutions, students enhance their skills in code review, debugging, and refactoring.

Thirdly, the Ethics and Norms module. This module integrates content on programming ethics and academic standards, clarifying the usage boundaries of AI tools. It prohibits AI use during basic training stages but allows reasonable use in complex projects, provided that AI-involved sections are labeled. Through activities like 'Case Discussions on AI Programming Ethics' and 'Analysis of Academic Misconduct Cases,' it fosters students' ethical awareness and sense of responsibility.

3.3 Innovation in Teaching Models: Constructing a 'Teacher-AI-Student' Tripartite Interactive Model

Replacing the traditional one-way teaching model, a hybrid teaching model that integrates a problem-driven approach, human-machine collaboration, and group cooperation has been developed. This creates a three-part interactive ecosystem featuring teacher guidance, AI empowerment, and student leadership, as illustrated in Figure 1.

Firstly, teaching is implemented, dividing the learning process into two stages: foundation and collaboration. In the foundation stage, the use of AI tools is forbidden. Students develop fundamental programming skills through teacher-led detailed lectures, classic case exercises, and basic project practice, ensuring they acquire the essential literacy for independent programming. In the collaboration stage, students may use AI tools appropriately, engaging in full-process training that follows the workflow of 'requirement analysis - AI generation - manual verification - iterative optimization'.

Secondly, it is driven by real-world projects. Teaching cases are rooted in actual enterprise needs and current industry trends, such as data visualization initiatives and intelligent algorithm optimization tasks, replacing traditional textbook exercises.

Thirdly, interactive training. Organize a 'Multi-version AI Solution Comparison' activity where students assess the strengths and weaknesses of AI-generated code and suggest optimizations; guide students to analyze differences in AI outputs based on varied prompt designs, fostering critical thinking; use AI study companion systems to

offer differentiated support, providing tiered guidance for students with weaker foundations .

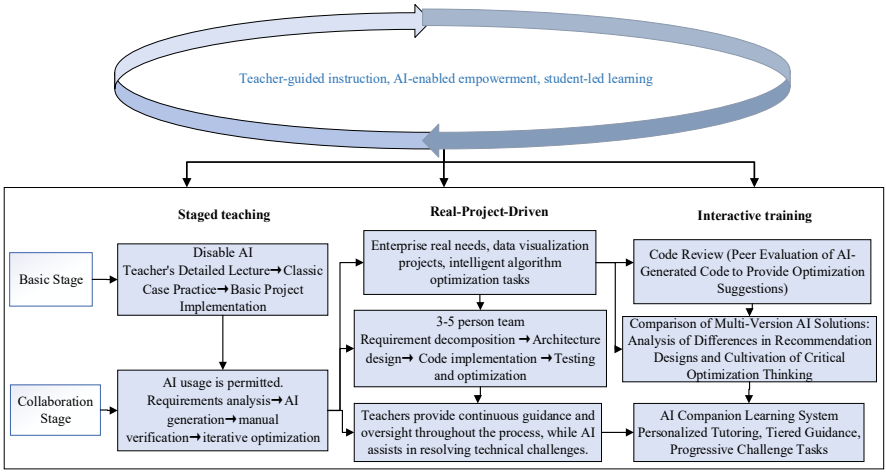

Fig. 1. Construction of the Tripartite Interactive Model

3.4 Evaluation System Transformation: From 'Result-Oriented' to 'Process + Competency-Oriented'

Develop a multi-dimensional, dynamic evaluation framework to comprehensively assess students' overall abilities, moving beyond traditional single-mode assessment methods. The first component evaluates basic competencies. Written exams assess fundamental theoretical knowledge, such as grammar rules and algorithm principles, while computer-based testing evaluates practical abilities like basic code writing and algorithm implementation. This ensures students' foundational competencies meet the required standards, and this section constitutes 30% of the overall assessment.

Secondly, process-based evaluation. The learning management system records the full process of students' learning data, encompassing prompt design logs, modification traces of AI-generated code, group discussion records, code review comments, and project phase reports. This evaluation focuses on assessing students' human-machine collaborative abilities, critical thinking skills, and collaborative skills. It constitutes 40% of the overall assessment and serves as its key component.

Thirdly,valuating outcome quality. Project outcome evaluation considers not just functional implementation but also multiple dimensions, including the rationality of architectural design, code readability, completeness of exception handling, and the appropriateness of AI tool usage. This assessment combines teacher evaluations and group peer reviews and constitutes 20% of the total evaluation.

Fourth, ethical literacy assessment. Evaluate students' understanding and application of AI programming ethics and academic standards through methods like course papers, thematic discussions, and project ethical statements. A 'one-vote veto' policy applies if

academic misconduct is detected. This component makes up 10% of the overall evaluation.

4 Conclusion

The velopment of AI technology aims not to replace university programming courses but to drive their paradigm shift. Drawing on constructivist learning theory and Bloom's Taxonomy of Cognitive Objectives, this paper presents a reconstruction plan for university programming courses in the AI era. The core focus is on establishing a composite core capability system of 'basic capabilities + AI collaborative capabilities'. This is accomplished through three key approaches: reconstructing curriculum content to emphasize 'equal importance of basics and collaboration', innovating the 'teacher-AI-student' ternary interactive teaching model, and transforming the evaluation mechanism to prioritize 'process as guidance and capability as core'. The ultimate goal is to move beyond training 'pure code writers' toward nurturing 'human-machine collaborative innovative developers'.This will foster more innovative programming talent suited to the digital economy era, thus truly realizing the core vision of 'technology empowering education'.

Acknowledgments

The research work reported in this paper is supported by Guangxi Academic Degree and Postgraduate Education Reform Research Topic (Grant No:JGY2020085).

References

1. Zheng, R. Y., & Yang, W. S. (2025). "Exploration on the reform of computer-related courses and integration of ideological and political education for the major of Building Environment and Energy Application Engineering in the era of artificial intelligence". In (Eds.), Proceedings of the 2nd National Conference on Ideological and Political Construction and Teaching Reform Research and Practice for the Major of Building Environment and Energy Application Engineering in Institutions of Higher Education (pp. 389–391). Guangdong University of Technology. https://doi.org/10.26914/c.cnkihy.2025.066797
2. Zhang, L. L., & Lei, J. J. (2025). "For the AI Era: Design and Application of the Teaching Framework for Programming Courses". Journal of Hubei University of Education, 42(08), 89–93. https://doi.org/10.26931/j.cnki.1674-344x.2025.08.014
3. Pan, Y. N., Ma, P. P., Chen, H. J., Wang, M. Z., & Xia, B. (2025). "Research on the Construction of Digital Courses with AI-Empowered Teaching—Taking the Digital Electronic Technology Course as an Example". Science & Technology Vision, 15(28), 90–92.
4. Yu, S. X., & Jia, S. (2024). "Based on the OBE Concept: Teaching Reform and Practice of Computer Programming Courses". Journal of Shazhou Professional Institute of Technology, 27(03), 38–41.

5. Wu, X. Y., Qing, X. D., Su, J. Y., Chen, Q., Qing, M. Y., & Hu, W. J. (2025). "Based on the OBE Concept: Reform of the Computer Simulation Course of Control Systems". Journal of Higher Education, 11(S2), 129–132. https://doi.org/10.19980/j.CN23-1593/G4.2025.S2.032
6. Jiang, H. H. (2023). "Discussion on teaching of principles of single-chip microcomputers in the era of AI automatic programming". Equipment Manufacturing Technology, (11), 112–114+132.
7. Liu, B., & Tan, W. B. (2025). "Under the background of the AI era: Exploration of teaching reform in the C++ object-oriented programming course". University Education, (22), 15–19+29.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.