ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

Analysis and Research on the Impact of Generative Artificial Intelligence on Engineering Programming Courses

Lili Zhang[a,*], Wei Wei[b], Jing Li[c], Wentao Wu[d], Ge Wang[e]

College of Information Engineering, Beijing Institute of Petrochemical Technology, Beijing, China
[a]zhanglili@bipt.edu.cn, [b]weiwei@bipt.edu.cn, [c]bipt_lijing@bipt.edu.cn, [d]wuwentao@bipt.edu.cn, [e]wangge@bipt.edu.cn
**Corresponding author*

Abstract: *With the breakthroughs of generative artificial intelligence in the fields of natural language processing and code generation, basic programming tasks can gradually be undertaken by large models, leading to the transformation of the programmer's role and challenges to traditional programming-related teaching content. Based on analyzing the reshaping of the programming workflow by large models, the transformation of the programmer's role, and the impacts and opportunities brought by large models to the core programming courses for engineering majors, this paper proposes comprehensive reform suggestions for the curriculum system, teaching concepts, and learning methods. By cultivating students' abilities in basic knowledge, logic, condensation and summary, and thinking in a more in-depth way, they can better use large models for programming and adapt to the needs of the era of artificial intelligence large models.*

Keywords: *Generative artificial intelligence; Engineering major; Programming*

1. Introduction

In recent years, generative artificial intelligence technologies, represented by DeepSeek, ChatGPT, Ernie Bot, Tongyi Qianwen, Doubao, and Kimi[1-5], have experienced vigorous development and widespread adoption. These systems have achieved remarkable breakthroughs in natural language processing and code generation, sparking extensive discourse not only within the technology sector but also across the broader educational landscape. This technological advancement has exerted a profound and far-reaching impact on programming positions within research and development (R&D) enterprises, as well as on conventional programming instruction in higher education institutions, presenting unprecedented challenges.

Core programming courses occupy a pivotal position within the curriculum architecture of engineering programs. They serve as the structural backbone of the entire curriculum framework, supporting the principal edifice of students' knowledge architecture. Furthermore, these courses function as a critical bridge connecting students with R&D enterprises and prospective employers, directly determining students' adaptability and competitiveness upon entering the professional workforce. Most importantly, they constitute the foundational bedrock for product design and system development, providing essential technical support for innovative practice across engineering domains.

As early as 2018, the Ministry of Education issued the Action Plan for Artificial Intelligence Innovation in Higher Education, which explicitly called for "guiding higher education institutions to focus on the world's scientific and technological frontiers, continuously enhancing capabilities in scientific and technological innovation, talent cultivation, and international cooperation and exchange in the field of artificial intelligence, thereby providing strategic support for the development of a new generation of artificial intelligence in China." This initiative provided strong national-level impetus for the integration of artificial intelligence and related technologies into higher education, particularly within engineering disciplines. However, the rapid and somewhat abrupt emergence of generative artificial intelligence has caught many educators in engineering higher education—especially instructors of core programming courses—somewhat unprepared. On one hand, faculty members face the challenge of how to update instructional content, as traditional programming curricula may no longer adequately address the requirements of the generative AI era[6-8]. On the other hand,

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

pedagogical methods must also be innovated to guide students in the appropriate and effective utilization of generative AI as an auxiliary tool for program design.

Nevertheless, it is undeniable that the use of generative artificial intelligence to assist in or even lead program design has become an inevitable trend in the future development of software and hardware R&D. Consequently, the pressing issue at present is to clarify and construct pedagogical philosophies, curriculum systems, and experimental practice formats that are adaptive to this trend. In terms of pedagogical philosophy, the focus should shift from the mere transmission of knowledge toward cultivating students' ability to solve practical problems using generative AI. Regarding curriculum architecture, it is necessary to appropriately integrate generative AI-related knowledge and application cases and to optimize course configurations. Experimental and practical formats must also evolve with the times, incorporating practice projects based on generative AI to enable students to master emerging technologies through hands-on engagement. This series of initiatives holds significant importance for promoting the healthy development of generative AI applications within core programming courses for engineering majors, contributing to the cultivation of engineering professionals better attuned to the demands of the era, and providing robust talent support for China's technological advancement and industrial upgrading[9-10].

2. The Reshaping of Programming Workflows by Generative Artificial Intelligence

(1) The Feasibility and Inevitability of Generative AI Assuming Basic Programming Tasks

With the continuous breakthroughs of generative artificial intelligence in natural language processing, code generation, and related domains, its assumption of foundational program design and coding tasks has become technically feasible. Through learning and comprehending massive repositories of code data, generative AI systems can rapidly identify programming patterns and generate corresponding code segments based on input requirements. From simple data structure operations—such as array sorting, linked list insertion and deletion—to the implementation of common algorithms, including Quicksort and Dijkstra's shortest path algorithm, and extending even to the provision and improvement of deep learning algorithms, generative AI demonstrates the capacity to deliver efficient and accurate solutions across numerous scenarios. This trajectory of capability development renders the gradual replacement of basic programming tasks by generative AI an inexorable trend. As the technology continues to mature, the efficiency and accuracy of generative AI in handling foundational programming tasks will further improve, securing its increasingly prominent role in the foundational work of programming.

(2) The Transformation of the Programmer's Role: From Code Writer to Product Designer

Once generative AI fully assumes responsibility for basic programming tasks, the focal point of programmers' work will undergo a fundamental shift—from the historically tedious work of code writing toward the domain of product design. This transition imposes higher competency requirements upon programmers. First and foremost, they must possess enhanced capability in requirements analysis, enabling them to thoroughly excavate the latent needs underlying users' explicitly stated requirements. Through multi-round communication with users, market research, and acute insight into industry trends, they must accurately grasp user needs. Simultaneously, programmers must demonstrate exceptional product design capability, translating abstract user requirements into clear, precise, and logically coherent functional specifications. This demands that programmers not only be well-versed in various product design principles and methodologies but also understand the characteristics of different platforms and user experience standards. For instance, when designing mobile applications, it is essential to give full consideration to the screen dimensions and operational habits of mobile devices to ensure that product functionality not only satisfies user requirements but also delivers a seamless user experience. Programmers must evolve from being mere implementers of code to becoming product designers equipped with holistic thinking, attending to the entire lifecycle of a product from conceptualization to deployment.

(3) Collaborative Development of Complex Functionality: Interaction Between Programmers and Generative AI

For the program design of complex functionalities, collaborative development between generative AI and programmers is of paramount importance. Leveraging its formidable code generation capabilities, generative AI can accomplish the preliminary coding of programs, thereby establishing a foundation for the entire development workflow. However, code generated by generative AI is

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

typically predicated upon general logic and patterns, making it difficult to fully align with the complex requirements of specific business scenarios. At this juncture, programmers must leverage their professional expertise to conduct in-depth, modular, and phased design and optimization of the program based on specific business logic and performance requirements. During the modularization process, programmers partition the entire program into multiple independent yet interrelated modules according to business functionality, clearly defining the responsibilities and interfaces of each module to ensure low coupling and high cohesion among them. In phased optimization, comprehensive considerations are made ranging from code readability and maintainability to performance optimization—such as reducing memory footprint and improving execution speed. Through close interaction between programmers and generative AI, fully harnessing the respective strengths of both parties, complex functional program design can be accomplished efficiently to satisfy increasingly complex business demands.

3. Challenges Posed by Generative AI to Traditional Programming Courses and the Characteristics of Programming in the New Era

(1) Adjustment of Instructional Content

In traditional programming curricula, instructional emphasis has long been concentrated on the minutiae of programming languages. A substantial proportion of class hours has been devoted to the repeated exposition of grammatical rules—such as the complex statement structures, extensive data type definitions, and diverse control structure implementations found in various programming languages—with students investing considerable time in rote memorization of these elements. However, as we enter the generative artificial intelligence era, such curricular proportions urgently require adjustment. Today, generative AI, with its powerful code generation capabilities, is capable of assuming the majority of basic code writing tasks. This implies that students no longer need to expend excessive effort on memorizing specific grammatical rules; rather, their learning focus should shift toward the underlying logic and core principles of program design. For example, students should attain a deep understanding of the design rationale behind data structures—why linked lists, stacks, and queues are designed as they are, and the scenarios to which each is best suited. They should master methods for algorithmic efficiency analysis, learning to evaluate the relative merits of different algorithms in terms of time complexity and space complexity. They should also internalize the principles of modular program design, understanding how to decompose a complex program into multiple functionally independent and easily maintainable modules.

To prepare students for the programming demands of the generative AI era, programming courses must incorporate knowledge modules related to interacting with generative AI. This encompasses multiple dimensions: in requirements description, students must learn how to articulate program requirements to generative AI with precision and clarity, avoiding substantial deviations between generated code and expectations due to ambiguous expression; at the code comprehension level, they must be capable of understanding the logical flow of code generated by generative AI, even when the coding style diverges from their own conventions; in evaluation and optimization, they must apply professional knowledge and tools to conduct comprehensive assessment of code output by generative AI, optimizing and refining it from the perspectives of security, performance, and maintainability.

(2) Transformation of Laboratory Practice Formats

Traditional programming laboratory exercises place heavy emphasis on developing students' manual code writing and program debugging capabilities. Students are required to repeatedly execute programs, identify, and repair syntactic and logical errors. While this approach effectively enhances students' mastery of code, it also suffers from inefficiency and disconnect from real-world programming scenarios. In the generative AI era, laboratory formats must be reformed to more extensively incorporate elements of program design utilizing generative AI. For example, at the commencement of a laboratory session, students may leverage generative AI to rapidly generate foundational code frameworks, subsequently building upon these frameworks to add and modify personalized functionalities. This approach not only conserves time but also enables students to devote greater attention to the overall architecture and functional implementation of programs.

Beyond transforming the fundamental format of laboratory exercises, the design of more comprehensive project-based experiments is critically important. These projects should no longer be limited to assessing students' code writing abilities; rather, they should emphasize students' comprehensive capacity to employ generative AI and other related tools to solve practical problems.

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

For instance, given a complex data analysis task, students would be required to comprehensively utilize generative AI to generate foundational code for data processing, combine it with professional data analysis tools to perform data cleaning, analysis, and visualization, and ultimately produce a comprehensive data analysis report. Through such comprehensive project-based experiments, students can better adapt to the programming demands they will encounter in actual professional work.

(3) The Trend of Universal Programming

The Reduction of Programming Barriers and the Acceleration of Programming Popularization. Historically, programming was regarded as a highly specialized discipline; mastering programming skills required prolonged systematic study, progressing from foundational programming language concepts to advanced knowledge of data structures and algorithms, supplemented by extensive practical experience. However, the emergence of generative artificial intelligence has fundamentally transformed this landscape, dramatically lowering the barriers to programming. Today, provided that one can articulate one's ideas and requirements in a clear and organized manner, even individuals without extensive programming backgrounds can achieve simple program design with the assistance of generative AI.

The Arrival of an Era Where Everyone Can Program. The reduction of programming barriers heralds the arrival of an era in which universal programming capability becomes the norm. In the future, programming will no longer be the exclusive domain of a small cadre of specialists; practitioners across all industries will be able to leverage programming to solve problems in their work and daily lives. The characteristics of an era in which everyone can program, everyone can develop, and everyone is a product manager will become increasingly pronounced, with programming evolving into an indispensable fundamental skill for people's personal and professional lives.

4. Evolving Enterprise Demands and Responsive Strategies for Universities, Faculty, and Students

(1) New Requirements for Programming Professionals in Enterprise Settings

1) Emphasis on and Demand for Comprehensive Competencies

With the widespread application of generative AI technology in software development, enterprise competency requirements for programming professionals have undergone significant changes, with greater emphasis placed on the cultivation and enhancement of comprehensive capabilities. In terms of interacting with generative AI, programming professionals must possess precise requirements articulation capability. They must be able to translate complex business requirements into clear and accurate natural language descriptions to enable generative AI to comprehend and generate code that satisfies the specified requirements.

Programming professionals must also possess the ability to understand and optimize code generated by generative AI. While code generated by generative AI may satisfy basic functional requirements, in practical application it may require adjustment and optimization based on specific business scenarios and performance requirements. Programming professionals must thoroughly understand the logical structure of code generated by generative AI, analyze its strengths and weaknesses, and identify areas for improvement. In generated code, issues such as algorithmic inefficiency or insufficiently clear code structure may exist, which programming professionals must address by applying their professional knowledge and experience to optimize the code, thereby improving execution efficiency and maintainability.

Product design capability represents another important enterprise requirement for programming professionals. In the generative AI era, programming professionals are no longer merely executors of code; they are designers and innovators of products. They must adopt the user's perspective, thoroughly understanding user needs and pain points, and designing products that are both innovative and user-friendly.

2) Integration and Application of Cross-Domain Knowledge

In the generative AI era, enterprises have imposed higher requirements on programming professionals' ability to integrate and apply cross-domain knowledge. As generative AI technology achieves deep integration with artificial intelligence, big data, the Internet of Things (IoT), and other domains, programming professionals must possess multidisciplinary knowledge and skills to effectively address complex project requirements.

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

(2) Responsive Measures for Universities and Faculty

1) Reconstruction of Pedagogical Philosophy

Transformation of Teaching Philosophy. Faculty roles are undergoing profound transformation; instructors must actively reconstruct their pedagogical philosophy, regarding generative AI as a critical teaching tool and resource and integrating it comprehensively into the instructional process. Faculty members are no longer mere transmitters of knowledge; rather, they serve as mentors who guide students in utilizing generative AI for autonomous learning and exploration.

Transformation of Learning Philosophy. For students, the updating of learning philosophy is imperative. They must move away from the historical model of passively receiving knowledge and proactively leverage generative AI to acquire knowledge and solve problems.

2) Reconstruction of Curriculum and Laboratory Practice

Content Reconstruction. In the process of reshaping curricular content, the proportion of pure programming language study should be appropriately reduced. Historical overemphasis on the memorization of programming language grammar rules has often resulted in students lacking practical application and innovation capabilities. Going forward, content concerning how to precisely describe requirements, efficiently interact with generative AI, and product design should be increased. Taking computer science courses as an example, dedicated curricular modules may be established to instruct students in how to formulate questions to generative AI in concise and precise language to obtain effective code recommendations. In product design courses, students may be guided to utilize generative AI for market research analysis and user requirement excavation, thereby designing products that better align with market demands.

Format Reconstruction. Regarding the innovation of laboratory exercise formats, greater emphasis should be placed on the design of open-ended, comprehensive project-based experiments. These projects are appropriately conducted in a collaborative group format. For example, in software development laboratory exercises, group members may jointly determine project requirements, utilize generative AI to complete code writing, testing, and optimization, and in the process, students not only learn how to leverage generative AI to accomplish complex program design but also develop comprehensive competencies such as teamwork and communication skills, thereby establishing a solid foundation for their future entry into society.

(3) The Elevation of Key Student Competencies

1) The New Connotation of Foundational Knowledge

The Importance of Deeply Understanding Program Design Principles. In the generative AI era, foundational knowledge remains the cornerstone upon which programming competency is built; however, the connotation of foundational knowledge has undergone certain transformations. Students no longer need to devote substantial effort to memorizing the specific syntax of programming languages; rather, they should place greater emphasis on understanding the fundamental principles of program design.

The Mastery and Application of Computer Systems Knowledge. Beyond program design principles, students must also acquire knowledge of computer systems, encompassing computer hardware architecture, operating system principles, and network communication principles. This knowledge will assist students in better understanding the operational mechanisms of programs within computer systems, thereby enabling more effective utilization of generative AI for program design.

2) The Central Role of Logical Reasoning

Logical Requirements for Requirements Description. In the process of interacting with generative AI and engaging in product design, logical reasoning plays a critically important role. First, when describing program requirements to generative AI, rigorous logic is essential. Only clear, precise, and logically rigorous requirements descriptions can enable generative AI to generate code that conforms to expectations.

The Application of Logic in Code Comprehension and Optimization. When generative AI produces code, students must employ logical reasoning to understand the code’s structure and functionality. Simultaneously, when optimizing and improving code, logical analysis is required to identify potential issues and propose reasonable optimization solutions.

3) The Significance of Conciseness and Synthesis Capability

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

The Ability to Rapidly Extract Critical Information. The capability for conciseness and synthesis holds significant importance for program design in the generative AI era. In the process of interacting with generative AI, students must rapidly extract the critical information embedded within their own ideas and requirements and accurately convey this information to generative AI.

The Ability to Summarize Code Functionality and Issues. When reading and understanding code generated by generative AI, conciseness and synthesis capability also enables students to rapidly summarize the primary functionality of code and identify potential issues therein.

4) The Deepening of Critical Thinking Capability

The Necessity of Deeply Contemplating Program Functionality. Critical thinking capability has become more important than ever in the generative AI era. Students must engage in deep contemplation of program functionality and implementation approaches, rather than relying solely on the output generated by generative AI.

The Ability to Explore Novel Program Design Methodologies. Beyond deeply contemplating program functionality, students must also leverage their thinking capabilities to explore new program design methodologies and technologies.

5. Conclusion

The application of generative artificial intelligence will bring profound impact and tremendous opportunity to core programming courses in emerging engineering programs. Curriculum architecture, pedagogical philosophy, and students' learning methodologies all require comprehensive reform and innovation to meet the requirements of the new era. By placing greater emphasis on cultivating students' competencies in foundational knowledge, logical reasoning, conciseness and synthesis, and critical thinking, we can enable them to better leverage generative artificial intelligence for program design and to distinguish themselves in the generative AI era. This represents a matter that urgently warrants in-depth consideration by universities and faculty in computer science and related disciplines at the present time.

Acknowledgement

This work is supported in part by General Project of Beijing City Higher Education Society in 2025 (Project Number: 125) .

References

[1] Bi, W. X. Challenges and Responses of Generative Artificial Intelligence to the Education Industry: An Analysis with ChatGPT as the Case Study. Jiangsu Higher Education, 2023, (08): 13-22.
[2] Wei, S. P, Fan, X. J., Wang, X. X., et al. The Potential and Risks of Applying ChatGPT in Higher Education: Experiences and Insights from American Universities. Modern Distance Education, 2024, (03): 18-27.
[3] Tao, W., Shen, Y. From ChatGPT to Sora: Four-Ability Education and Paradigm Innovation for AIGC. Modern Educational Technology, 2024, 34(04): 16-27.
[4] Zheng, Y. H., Zhou, D. H., Zhang, Y. H., et al. ChatGPT from the Perspective of Computational Pedagogy: Connotation, Themes, Reflections, and Challenges. Journal of East China Normal University (Educational Sciences Edition), 2023, 41(07): 91-102.
[5] Qi, Y. Y, Wang, L. M. Application of Generative Artificial Intelligence in Open Education: Opportunities, Challenges, and Application Scenarios. Adult Education, 2024, 44(06): 56-61.
[6] Lu, Y., Yu, J. L., Chen, P. H., et al. Research and Prospects on the Educational Application of Multimodal Generative Artificial Intelligence. E-Education Research, 2023, 44(06): 38-44.
[7] Huang, R. H. Integrating Artificial Intelligence into Education: Conceptual Transformation, Morphological Reshaping and Key Measures. People's Tribune Academic Frontiers, 2024, (14): 23-30.
[8] Wu, D., Li, H., Chen, X. Analysis of the Impact of General-Purpose Artificial Intelligence on Educational Applications. Open Education Research, 2023, 29(02): 19-25+45.
[9] Li, Q. Y., Geng, Y. L., Peng, W. J, et al. "Private Tutor" or "Ghostwriter": Exploration of Computer Practical Teaching Based on Generative Artificial Intelligence. Experimental Technology and Management, 2024, 41(05): 1-8.

ISSN 2522-6398 Vol. 9, Issue 6: 52-58, DOI: 10.25236/FER.2026.090607

[10] Wang, Q., Guo, F., Zhang, Y. X., et al. Research on Deep Classroom Reform Based on Generative Artificial Intelligence. Open Education Research, 2024, 30(04): 104-112.