THE IMPACT OF ARTIFICIAL INTELLIGENCE ASSISTANTS ON COGNITIVE DEPENDENCY AND CODE QUALITY IN PROGRAMMING EDUCATION

Ismailov I T
Samarkand branch of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi ilxomismailov1988161287@gmail.com
Saydazimov A. B
Samarkand branch of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi saydazimovalisher08@gmail.com

Abstract: *This article comprehensively and critically analyzes the profound cognitive changes arising from the rapid integration of generative artificial intelligence (AI) tools and automated code assistants into the teaching of programming fundamentals in higher education institutions, as well as the quality indicators of the generated software code [1], [6]. The primary objective of the research is to scientifically evaluate the risk of cognitive dependency caused by an overreliance on machine intelligence among novice programming students, to study the decline in memory and analytical capacity, and to reveal the direct impact of these technologies on algorithmic thinking and code reliability based on empirical evidence [2]. The study was conducted through a systematic review of leading international scientific sources, the synthesis of modern pedagogical methods for teaching programming, and a comparative assessment of the experiences of advanced technological universities. The obtained results firmly confirm that although AI assistants increase the speed of coding and initial prototyping to an unprecedented degree [5], their completely uncontrolled use without ethical and didactic norms weakens students' fundamental logical thinking skills and directly causes an increase in hidden vulnerabilities within the software code that pose a future risk to the entire system [4]. At the conclusion of the study, deeply strategic, scientifically grounded proposals and recommendations are put forward for forming a "culture of critical code review," keeping the human factor at the center of control, and implementing rules of digital hygiene in practice, rather than banning AI tools in the higher education system [7].*

Keywords: *programming education, artificial intelligence assistants, cognitive dependency, code quality, generative models, teaching methodology, academic integrity, algorithmic thinking, technical debt, code audit.*

INTRODUCTION

In the era of the rapid development of the global digital economy and global technological transformation, education in Information Technology (IT) and computer science is going through an unprecedented stage of profound shifts and revolutionary changes [6]. In the fields of software engineering, the widespread practical implementation of modern AI-based code assistants (e.g., GitHub Copilot, Tabnine) and exceptionally large language models (such as ChatGPT, Claude) has conceptually,

philosophically, and practically completely transformed the traditional methods of training future programmers that have been developed over decades [1]. In the past, a student just beginning to learn programming would read numerous materials and spend hours conducting independent research to solve a complex algorithmic problem, find logic errors (bugs), and reach a systematic conclusion. Today, advanced generative models provide ready-made, complex, and logically complete software solutions in fractions of a second [3].

However, behind this attractive convenience and technological ease, highly serious concerns are surfacing, such as a decline in the level of comprehension of the most fundamental principles of programming, the weakening of analytical thinking, and most dangerously, the formation of cognitive dependency on computer intelligence among learners [2], [7]. In the international scientific community, particularly among computer science educators, debates are becoming increasingly fierce regarding the risk that for novice students, these smart tools are not mere assistants, but rather a passivating factor that degrades their independent analytical capacity, ability to build logical chains, and problem-solving skills [6]. During the educational process, when a student blindly copies and successfully runs lines of code they do not fully understand, it creates an illusion that covers up deep gaps in their knowledge rather than signifying that they actually found the solution.

Simultaneously, the internal quality, stability, optimization level, and systemic security of the code generated by generative models have also become one of the global engineering problems today [4]. While the software code provided by machine intelligence often appears perfect on the surface, syntactically flawless, and functionally operational, there is an exceptionally high risk that its complex architectural structure contains serious hidden logic errors, inconvenient loops that consume excessive computing power, and open vulnerabilities that leave the entire system defenseless from a cybersecurity perspective [3], [4]. Therefore, this major research work is aimed not at entirely discarding the capabilities of AI, but rather at developing international standards for its safe integration within the framework of educational ethics.

Research Methodology

To comprehensively illuminate this international-level problem, methods of systematic literature review, deductive synthesis, and comparative evaluation of quantitative and qualitative indicators were extensively utilized in this research. The definitive methodological foundation of the study relies on the synthesis of fundamental scientific works published over the last three years in the world's most prestigious scientific and technical databases (IEEE Xplore, ACM Digital Library, Scopus) that empirically investigated the impact of AI in programming education through real-world experiments [1], [7].

In purposefully selecting the literature and practical experimental results, strict scientific inclusion criteria were followed in accordance with PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) international standards:

• Experimental Validation: Results of measurable real-world experiments conducted over several months on novice learners taking introductory computer science courses (CS1/CS2) at higher education institutions [2].

• Code Quality Measurement: Specific statistical and technical data on the metric quality indicators (cyclomatic complexity, number of logic errors and bugs, cybersecurity level) of software code written independently by a human (student) versus code generated with the help of AI (GitHub Copilot, ChatGPT) [4].

• Cognitive and Psychological Factor: Pedagogical surveys and analytical conclusions measuring the volume of cognitive load, logic retention rate, and dependency level of the student during the process of working with reliance on a machine [3], [7].

The data collected based on these strict criteria were not only classified theoretically but also mutually compared from the perspective of practical educational efficiency. This allowed for drawing objective conclusions aligned with international standards by generalizing the similarities and differences of various approaches.

Results

The extremely deep meta-analyses and reviews of international experiments conducted firmly confirmed with factual evidence that students who had access to the continuous use of AI code assistants in educational institutions managed to drastically reduce the total time spent solving a specific complex problem and creating a ready-to-run initial prototype (MVP) by an average of 40 to 50 percent [2], [5]. AI showed an absolutely positive result in terms of the speed of obtaining initial results. However, the other, highly dangerous side of the coin is that in the subsequent control stages conducted to check the quality of assimilation and comprehension of the code, scientific experiments definitively proved that the ability of the students in this exact group to independently identify hidden errors in the ready code, explain them, and adapt the written algorithmic logic to a completely different, new system remained significantly lower (almost half as effective) compared to a traditional control group that did not use AI at all [2].

Deep automated technical analyses conducted on the quality, stability, and internal architecture of the software code showed that while fully automatically generated code blocks work without errors upon initial launch, they often lead to an increase in geometric progression of the amount of "technical debt," which seriously hinders the future development of the program and the updating of its codebase [4]. Because machine language models have also trained on old and erroneous code from massive open-source repositories like GitHub, researchers have repeatedly noted that their proposed standard solutions often contain unoptimized algorithmic approaches that excessively consume computing resources (RAM and CPU time) and fail to meet the latest security standards [3], [4]. When a student simply copies this code, they transfer these vulnerabilities to the entire project.

Furthermore, international global educational experience, specifically the situation in leading educational institutions in the US and Europe, firmly proves that attempting to completely ban AI tools in programming classes using special software (proctoring) and applying punitive mechanisms is entirely ineffective [6]. Conversely, the world's leading technological universities are strictly implementing a policy of "transparent collaboration" in the educational process, formally requiring students to fully and transparently declare exactly which modules in their submitted practical work were written by a human and which directly with the help of AI, as well as to explain the AI-written code in detail during the defense [7]. This practice prevents the student from hiding their knowledge behind artificial intelligence.

Discussion

Finding and maintaining the highly delicate balance between a student's academic integrity, their true level of knowledge in the field of information technology, and their continuous personal cognitive development remains the most prioritized and strategic direction of today's modern digital education system [6]. According to our deep analyses, taking into account the fiercest debates in international scientific circles, maintaining the ultimate legal, technical, and moral responsibility for the digital product being created in programming education and future engineering practice—not in some unconscious computer algorithm, but directly in the mind of the human specialist—must be the foundation of global educational ethics [7].

Based on the general conclusions of researchers, it can be said that AI assistants should remain under control as passive assistants in the activities of future programmers and engineers, rapidly executing only mentally fatiguing, constantly repetitive, and routine tasks (e.g., writing boilerplate templates, automating simple and identical basic tests, formatting code) [5]. When a student fully hands over their chain of systemic and logical thinking, cause-and-effect connections, and the function of algorithmic problem solving to the discretion of a machine and its generative memory, it is absolutely inevitable that this exact point of convenience will serve as a point of evolutionary decline and degradation for their development as a true professional [2].

Currently, the rigid, traditional teaching methodologies applied in most higher education institutions and educational standards are extraordinarily late in preparing students for an entirely new working environment, namely the collaboration process in the format of "AI pair programming" [5]. The modern digital era demands primarily highly cognitive skills from the new generation of IT specialists: not to write code letter by letter starting from a blank page, but to be able to read, rapidly analyze, refactor, and critically check the systemic security of thousands of lines of complex code generated by an alien computer (AI) in seconds [1], [4]. Therefore, shifting the education system from assessing code writing to assessing code comprehension has become the main core of discussions.

Conclusion

At the end of the extremely detailed and deep comparative analyses conducted within the framework of this fundamental work, the following singular, objective, and firm conclusions were reached: in the modern technological education system, attempting to completely exclude and strictly block generative AI tools and intelligent code generators from students' lives and university laboratories is an invalid and retrograde approach that wastes valuable time and artificially halts technological progress [6]. Ministries governing educational processes and higher education, as well as university administrations, must urgently shift their focus away from various illogical information bans; instead, it is a pressing necessity to focus more than ever on instilling in students, from the very first days of their academic years, a strict sense of "digital hygiene," technological accountability, and the ethics of cultured interaction with AI models through queries (prompt engineering ethics) [7].

As a practical conclusion of our research, we firmly recommend that, alongside traditional engineering subjects existing in technical and IT-oriented universities, entirely new practical modules that fully meet modern requirements—such as "Academic Programming and Authorship Ethics," "Engineering of Working with Large Language Models and Prompts," and "Methodology of Safe Synergistic Coding with AI"—must be integrated into educational programs based on mandatory state standards. This step will create a solid, unshakable foundation to ensure the firm competitiveness of our future generation of graduates in the international labor market and their ability to adapt to advanced technologies flawlessly and ethically [6], [7].

REFERENCES:

1. Becker, B. A., et al. (2023). "Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation". Proceedings of the 54th ACM Technical Symposium on Computer Science Education, 500-506.

2. Kazemitabaar, M., et al. (2023). "Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming". Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, 1-14.

3. Vaithilingam, P., et al. (2022). "Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models". CHI Conference on Human Factors in Computing Systems Extended Abstracts, 1-7.

4. Dakhel, A. M., et al. (2023). "GitHub Copilot AI pair programmer: Asset or Liability?". Journal of Systems and Software, 203, 111734.

5. Chen, Y., et al. (2023). "Is GitHub Copilot a Substitute for Human Pair-programming? An Empirical Study". Proceedings of the 45th International Conference on Software Engineering (ICSE).

6. Denny, P., et al. (2023). "Computing Education in the Era of Generative AI". Communications of the ACM, 66(6), 56-67.

7. Qadir, J. (2023). "Engineering Education in the Era of ChatGPT: Promise and Pitfalls of Generative AI for Education". 2023 IEEE Global Engineering Education Conference (EDUCON), 1-9.