Educating Augmented Programmers

Mary Sánchez-Gordón, Østfold University College

Edmundo Tovar and **Ricardo Colomo-Palacios**, Universidad Politécnica de Madrid

Nelson Piedra, Universidad Técnica Particular de Loja

Manuel Castro, Spanish University for Distance Education

There is an artificial intelligence-based technology that has the potential to augment the work of human programmers. This article discusses some capabilities built around generative artificial intelligence and large language models that impact programming education.

Over the last three years, venture capital companies have invested around US$1.7 billion in generative artificial intelligence (GAI) solutions, with the most funding for AI software coding and AI-enabled drug discovery.[1] The application of AI-based technologies in the daily tasks of programmers is delivering on the promise of augmented programming. Research is increasingly locating tasks where AI works alongside traditional tools and human workflow. Popular tools like ChatGPT and GitHub Copilot can assist programmers in code generation, competition, and optimization. At the same time, specific tools in the market, like TabNine or Replit Ghostwriter, help programmers in other tasks like refactoring or documenting. As the generative pretrained transformer (GPT) technology and other advanced forms of AI continue to evolve and reach wider adoption, it becomes crucial to consider how these innovations will impact the future of the job market. It is essential to identify and understand the skills that will be most valuable in programming education to prepare ourselves for this new technological era. These AI-based tools are envisaged to increase the amount

Digital Object Identifier 10.1109/MC.2023.3313325
Date of current version: 13 November 2023

EDITORS
GEORGE HURLBURT U.S. Federal Service (Retired), USA; gfhurlburt@gmail.com
SOREL REISMAN California State University, USA; sreisman@computer.org

of work performed by programmers, providing a way to combat the shortage of IT talent. In this article, the authors review the impacts of AI-based programming tools on programmers' professional practice and propose a way to adapt initial professional education to this new scenario in the context of the Computing Curricula 2020 (CC2020), a joint initiative by the ACM and the IEEE Computer Society.

IS THIS THE END OF COMPUTER PROGRAMMING?

No, it is not the end of computer programming. Computer programming continues to be a crucial skill and profession, with increasing demand year over year. While advancements in AI and automation may streamline specific programming tasks and increase efficiency, they do not replace the need for human programmers. These AI-based programming tools function essentially as specialized assistants: they can clarify concepts, answer questions, detect errors, and explain why a snippet of code is not working. They can also write explanations for poorly documented code snippets and offer code suggestions to carry out routine tasks, thereby enhancing productivity. Regardless of the experience level of programmers, they need to understand the code, tasks, and programming concepts.[2] They spend over half of their time on program comprehension.[3] As software systems continue to evolve and increase in complexity and magnitude, skilled programmers will always need to create, maintain, and improve them. So far, the cognitive load on the human programmer persists.[4] However, upskilling is an appealing option because the landscape for programming-related occupations will change as they appear more susceptible to being influenced by AI-based tools.[5] Job markets are currently changing due to mass tech layoffs.

In short, while there may be changes and advancements in the programming field, programming will be around for a while. If anything, the ways of working will change, and programming skills will become even more valuable and essential as technology continues to play an increasingly significant role in our lives.

TOOLS TO AUGMENT PROGRAMMERS' POTENTIAL

Since the 1970s, fourth-generation languages have aimed to make programming easier using computer-assisted software engineering tools. Therefore, code generation automation has been a longstanding goal. However, most of these approaches are semi-automated, requiring programming skills and often requiring experts to be involved. Although there is still no best tool for programming, programming languages and tools have evolved over the years to address capabilities like code completion, code translation among programming languages, software documentation, debugging, and testing. The so-called AI coders are reaching a level of intelligence that increasingly enables them to rival human software developers. Remarkable progress has been made in the field of GAI and large language models (LLMs),[5] but lately, GPTs have taken the spotlight.[1] LLMs are commonly related to GPTs but are not limited to transformer-based models. They can be trained using a range of architectures to go beyond natural language uses and bring code-generating abilities.[5]

Utilizing this cutting-edge technology, tools such as Copilot, TabNine, and Replit Ghostwriter attempt to overcome the shortcomings of their forerunners. They use natural language queries and the ability to program by example, a technique called *few-shot learning* in the research literature. For instance, it allows them to suggest real-time code completions based on what programmers type and the rest of the code. These tools aim to help programmers improve their productivity by assisting them with tasks like the ones mentioned above and augmenting processes like programming rather than becoming the programmer itself. Programmers can ask for recommendations on libraries, convert a program from one language to another or data from one format to another, generate filler content for something like an SQL database, and receive support for the debugging process of a program.

No wonder programmers want to learn how to use AI-based programming tools to their advantage, but how future programming education can address these tools remains to be seen.

EDUCATING PROGRAMMERS

CC2020 is a global initiative that focuses on competencies[6] and outlines curricular guidelines for educational programs in computing. CC2020 emerged as a response to the changing dynamics of computing and changes in the workplace. This led to the development of a framework that includes three competency dimensions: knowledge, skills, and dispositions. These guidelines can be tailored to accommodate the emergence and prevalence of new technologies, such as GAI and LLMs.

The CC2020 can be vital in preparing the next generation of programmers to work in a world where LLMs and similar technologies are ubiquitous. By adopting a proactive and adaptive approach, the curriculum can ensure that programmers benefit from these tools and understand, critique, and positively contribute to their development and application across various fields.

From this perspective, the evolution of AI-based programming tools is changing the scenario since they facilitate access to knowledge and pose a human-centered partnership model of programmers and AI working together to enhance programming, learning, and writing skills. Figure 1 illustrates the potential impact of AI-based programming tools on the set of computer programmers' skills and abilities proposed by the occupational information network from the United States.[7]

It seems that by their nature, AI-based programming tools can have more influence on hard skills than soft skills. We envision a low impact on analytical skills (systems analysis, operations analysis, and quality control analysis) and management skills (coordination, time management, and monitoring). Although mathematics and reading comprehension are hard skills, we believe only the latter is unaffected by these new tools. In line with this, mathematical reasoning and other related abilities (number facility and information ordering) are also influenced. The influence these tools can have on written expression is also not surprising. However, soft skills like critical thinking, problem-solving, and decision-making rise as necessary to maximize the benefits of using these tools.

It is vital to understand the limitations of GAI and LLMs and critically evaluate their outputs. Likewise, five relevant abilities also seem to come into play (fluency of ideas, originality, problem sensitivity, and deductive/inductive reasoning), whereas the remaining skills and abilities are still needed.

In this panorama, raising requirements for degrees related to computer science is a valid mechanism of "natural selection." However, we advocate helping students develop a growth mindset by adapting initial professional education beyond fundamental programming.

One way is to integrate market tools like Copilot into courses related to programming fundamentals. Although it can improve proficiency with syntaxes and semantics of programming languages, students still need to ensure the code is functional. These AI-based programming tools can give students all the pieces they might need, but it falls on the student to put the pieces together in a way that fulfills the

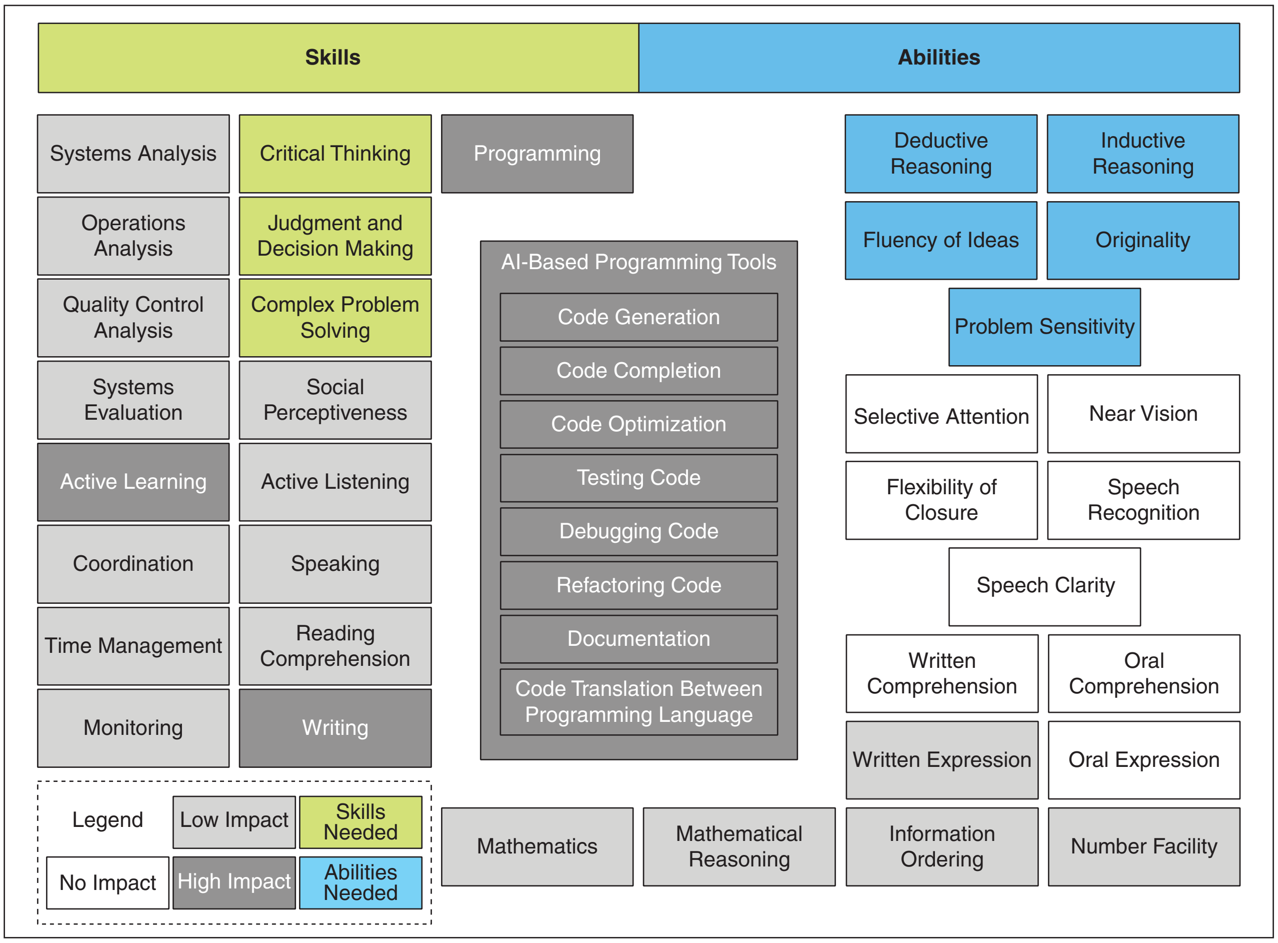

FIGURE 1. An overview of the impacts of AI-based programming tools capabilities on professional computer programmers' skills and abilities.

requirements. In other words, Copilot generates code that provides some options that could be the right fit, but the programmer still must decide which snippets to use and how to use them: program comprehension. This entails devising a plan that calls for critical thinking, problem-solving, decision-making, and abilities related to problem sensitivity, fluency of ideas, and originality. Therefore, course design should embrace this technology while cultivating the necessary soft skills for future professionals.

In this regard, it is essential to emphasize the education of programmers in ethics and responsibility. The GAI and LLMs raise ethical considerations, ranging from bias removal to privacy concerns and adapting to the specific challenges posed by AI. CC2020 can potentially reinforce these areas, preparing students to make informed and ethical decisions in their professional endeavors.

Additionally, although Copilot automatically suggests the code it decides the programmer might want, the more specific the code comments are, the better Copilot can create code that matches the programmer's intentions. Thus, a valuable skill is communicating effectively by writing comments in the code. In this way, AI-based programming tools, and others, can understand the pieces of code. Writing according to the audience's needs has always been crucial, but new motivations have emerged. Students can write a test title in natural language so that Copilot can use it for unit testing. However, they must ensure proper functioning using their analytical skills and knowledge while gaining expertise in application domains.

The underlying features of these tools are very appealing, especially to nonexpert programmers, because they can overcome barriers related to hard skills. However, using third-party tools or libraries also requires considering the potential impact on aspects like security risks and control over the piece of software. For instance, a piece of code that uses an AI-generated library or ready-to-use agents from a free marketplace like Fixie[8] is threatened if, subsequently, it appears that the library or agent has flaws or defects. Thus, students must be knowledgeable about the limitations of AI-based programming tools.

In practice, AI-based programming tools also impact the effort required to perform some programming tasks. In the best scenario, these tools can increase the amount of work performed, and therefore, future programmers should gain expertise using these tools. However, we also note that professional programmers, at all levels of experience, rarely work alone and code in a vacuum, so other soft skills not directly impacted by AI-based programming tools should be cultivated. In this new scenario, it is also expected that question and answer sites like Stack Overflow that connect programmers to help solve problems also change.

Another way to implement the human-centered partnership model is to carefully design in-class activities or labs that take students through a set of exercises or tasks guided by an autonomous intelligent teaching assistant (an AI tutor) rather than an instructor or teacher assistant that improves students' understanding of the material.[9] In this case, the focus is on the learning journey and may empower students to become self-directed and autonomous learners.[10] In addition, an AI tutor has the potential to adapt to goals desired by the students, their speed of learning, and their level of knowledge to aim to ensure they are getting the most out of their education.[11] This online teaching can support students, particularly from minority groups, and decrease dropout rates.

AI-based programming tools are gaining popularity and use due to the promise of a faster, less manual programming process. Thus, educating programmers on the limitations of these and other tools is needed to let them decide when to use them. In the case of AI-based tools, they must learn when to ask for assistance and when to make decisions for themselves. Through a multifaceted approach encompassing ethics, practical application, and interdisciplinary collaboration, CC2020 can be poised to define the trajectory of computer science education in the age of GAI. The course design also must adapt to introduce new tools that boost hard skills like programming and develop soft skills like critical thinking, complex problem-solving, and decision-making as never before. Finally, education must be directly connected to real-world situations and prepare students for trend technologies that respond to industry needs. C

MARY SÁNCHEZ-GORDÓN is an associate professor in the Computer Science Department, Østfold University College, NO-1757 Halden, Norway. Contact her at mary.sanchez-gordon@hiof.no.

EDMUNDO TOVAR is a professor in the Computer Languages and Systems and Software Engineering Department, Universidad Politécnica de Madrid, 28660 Madrid, Spain. He is a Senior Member of IEEE. Contact him at edmundo.tovar@upm.es.

RICARDO COLOMO-PALACIOS is a professor at Universidad Politécnica of Madrid, 28660 Madrid, Spain. He is a Senior Member of IEEE. Contact him at ricardo.colomo-palacios@hiof.no.

NELSON PIEDRA is a professor at Universidad Técnica Particular de Loja, 110107 Loja, Ecuador. He is a Member of IEEE. Contact him at nopiedra@utpl.edu.ec.

MANUEL CASTRO is an electrical and computer engineering professor at the Spanish University for Distance Education, 28040 Madrid, Spain. He is a Life Fellow of IEEE. Contact him at mcastro@ieec.uned.es.

REFERENCES

1. "Jackie Wiles beyond ChatGPT: The future of generative AI for enterprises." Gartner. Accessed: Mar. 29, 2023. [Online]. Available: https://www.gartner.com/en/articles/beyond-chatgpt-the-future-of-generative-ai-for-enterprises
2. A. Heinonen, B. Lehtelä, A. Hellas, and F. Fagerholm, "Synthesizing research on programmers' mental models of programs, tasks and concepts: A systematic literature review," *Inf. Softw. Technol.*, vol. 164, Dec. 2022, Art. no. 107300, doi: 10.1016/j.infsof.2023.107300.
3. X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: A large-scale field study with professionals," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, p. 584, doi: 10.1145/3180155.3182538.
4. N. Povarov. "AI for software developers: A future or a new reality?" InfoQ. Accessed: Apr. 3, 2023. [Online]. Available: https://www.infoq.com/articles/ai-for-software-developers/
5. T. Eloundou, S. Manning, P. Mishkin, and D. Rock, "GPTs are GPTs: An early look at the labor market impact potential of large language models," 2023, *arXiv:2303.10130*.
6. CC2020 Task Force, *Computing Curricula 2020: Paradigms for Global Computing Education*. New York, NY, USA: ACM, 2020.
7. "15-1251.00 – Computer programmers." O*NET OnLine. Accessed: Mar. 31, 2023. [Online]. Available: https://www.onetonline.org/link/summary/15-1251.00?redir=15-1131.00
8. "FIXIE.AI — Build on LLMs." FIXIE. Accessed: Apr. 2, 2023. [Online]. Available: https://www.fixie.ai/
9. S. Jalil, S. Rafi, T. D. LaToza, K. Moran, and W. Lam, "ChatGPT and software testing education: Promises and perils," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW)*, 2023, pp. 4130–4137, doi: 10.1109/ICSTW58534.2023.00078.
10. S. Sok and K. Heng. *ChatGPT for Education and Research: A Review of Benefits and Risks*. (Mar. 2023). SSRN. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4378735
11. "AI for programming education." Microsoft Research. Accessed: Sep. 14, 2023. [Online]. Available: https://www.microsoft.com/en-us/research/project/ai-for-programming-education/

Digital Object Identifier 10.1109/MC.2023.3326622