# Educating Augmented Programmers

**Mary Sánchez-Gordón,**
**Edmundo Tovar,**
**Ricardo Colomo Palacios,**
**Nelson Piedra,**
**Manuel Castro**

**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 (AI) and large language models (LLMs) that impact programming education.**

Over the last three years, venture capital companies have invested around $1.7 billion in generative AI (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 the human workflow. Popular tools like ChatGPT and Copilot can assist programmers in aspects like code generation, code competition and code optimization, while there are also specific tools in the market like TabNine or Replit Ghostwriter helping programmers in other tasks like refactoring or documenting as well. These AI-based tools are envisaged to increase the amount of work performed by programmers, providing a way to combat the shortage of IT talent. In this article, authors review the impact 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.

## 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 certain programming tasks and increase efficiency, they do not replace the need for human programmers. Regardless the experience level of programmers, they need to understand the code, tasks, and programming concepts [2]. They spend more than half of their time on program comprehension [3]. As software systems continue to evolve and increase in complexity and magnitude, there will always be a need for skilled programmers to create, maintain, and improve them. So far, the cognitive load on the human programmer persist [4]. However, upskilling is an appealing option because the landscape for programming-related occupations will change as they appear to be more susceptible to being influenced by AI-based tools [5], and job markets are currently changing due to mass tech layoffs.

In short, while there may be changes and advancements in the field of programming, programming is not going away anytime soon. 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 (4GL) aimed to make programming easier by using Computer-Assisted Software Engineering (CASE) tools. Therefore, code generation automation has been a longstanding goal. However, most of these approaches are semi-automated approaches which require programming skills and often need experts to be involved. Although there is still not a best tool for programming, programming languages and tools have evolved over the years to address capabilities like code completion, code translation between programming languages, software documentation, debugging and

testing code. In fact, remarkable progress has done in the field of GAI and LLMs [5], but lately, Generative Pre-trained Transformers (GPT) have taken the spotlight [1]. LLMs are commonly related to GPT, but they are not limited to transformer-based models. They can be trained using a range of architectures to go beyond natural language uses and brings 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 take advantage of 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 not only on what programmers type but also considering the rest of the code. The goal of these tools is 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 either programs 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 unclear.

## EDUCATING PROGRAMMERS

Computing Curricula 2020 (CC2020) is a global initiative that focuses on competencies [6]. CC2020 emerged as a response to the changing dynamics of computing and changes in the workplace. This led to develop a competency framework that includes three competency dimensions: knowledge, skills, and dispositions.

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 U.S. [7].

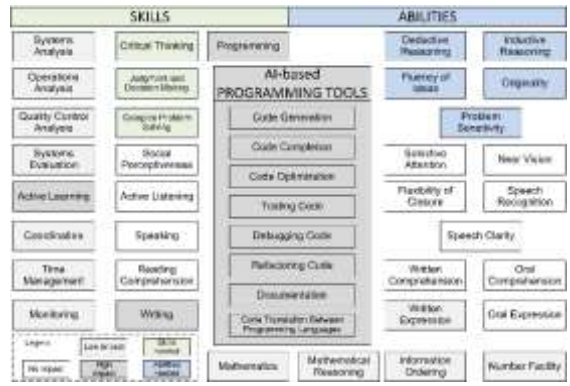


**Figure 1. Overview of the impact of AI-based Programming Tools capabilities on professional computer programmers' skills and abilities.**

It seems that, by their nature, AI-based Programming Tools can influence more on hard skills than soft skills. We envision a low impact over analytical skills —systems analysis, operations analysis, quality control analysis— and management skills —coordination, time management, and monitoring. Although mathematics and reading comprehension are hard skills, we believe that only the last is not affected 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. 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 for 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 make sure that 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 all together in a way that fulfils the requirements. In other words, Copilot generates code that provides some options that could be the right fit, but the programmer still has to decide which snippets to use and how to use them — program comprehension. It entails devising a plan that calls for critical thinking, problem-solving, and decision-making as well as

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.

Additionally, although Copilot automatically suggests the code it thinks the programmer might want, the more specific code comments are, the better Copilot can create code that matches the programmer's intentions. Thus, a valuable skill is to communicate 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 needs of the audience has always been key but now new motivations emerge. Students can write a test title in natural language so that Copilot can use it for unit testing. However, they must ensure proper functioning by using their analytical skills and knowledge while gaining application domain expertise.

The underlying features of these tools are very appealing, especially to non-expert programmers because they can overcome barriers related to hard skills. However, the use of 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 & Answer sites like stack overflow that connect programmers with each other 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 journey learning 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 more 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 they should use them. In the case of AI-based tools, they must learn when to ask for assistance and when to make decisions for themselves. 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.

## REFERENCES


1. Jackie Wiles Beyond ChatGPT: The Future of Generative AI for Enterprises Available online: https://www.gartner.com/en/articles/beyond-chatgpt-the-future-of-generative-ai-for-enterprises (accessed on 29 March 2023).
2. Heinonen, A.; Lehtelä, B.; Hellas, A.; Fagerholm, F. Synthesizing Research on Programmers' Mental Models of Programs, Tasks and Concepts -- a Systematic Literature Review 2022.
3. Xia, X.; Bao, L.; Lo, D.; Xing, Z.; Hassan, A.E.; Li, S. Measuring Program Comprehension: A Large-Scale Field Study with Professionals. In Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE); May 2018; pp. 584–584.
4. Nikita Povarov AI for Software Developers: A Future or a New Reality? Available online: https://www.infoq.com/articles/ai-for-software-developers/ (accessed on 3 April 2023).
5. Eloundou, T.; Manning, S.; Mishkin, P.; Rock, D. GPTs Are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models 2023.
6. CC2020 Task Force *Computing Curricula 2020: Paradigms for Global Computing Education*; ACM: New York, NY, USA, 2020;
7. O*net Online 15-1251.00 - Computer Programmers Available online: https://www.onetonline.org/link/summary/15-1251.00?redir=15-1131.00 (accessed on 31 March 2023).
8. Fixie.Ai — Build on LLMs Available online: https://www.fixie.ai/ (accessed on 2 April 2023).
9. Jalil, S.; Rafi, S.; LaToza, T.D.; Moran, K.; Lam, W. ChatGPT and Software Testing Education: Promises & Perils 2023.
10. Sok, S.; Heng, K. ChatGPT for Education and Research: A Review of Benefits and Risks 2023.
11. AI for Programming Education. *Microsoft Research*.