Generative AI Tools in an Undergraduate Computer Science Program

Sebastian Simon, Raquel Coelho, Iza Marfisi-Schottman, Roy Pea

International Society of the Learning Sciences

Generative AI Tools in an Undergraduate Computer Science Program

Sebastian Simon, Le Mans Université, sebastian.simon@univ-lemans.fr
Raquel Coelho, Stanford University, rcoelho@stanford.edu
Iza Marfisi-Schottman, Le Mans Université, iza.marfisi@univ-lemans.fr
Roy Pea, Stanford University, roypea@stanford.edu

Abstract: We surveyed all undergraduate students (N = 121) enrolled in a three-year Computer Science (CS) program in France to learn about their use of generative AI tools. Our findings reveal widespread use of these tools across all academic years, with a higher proportion of later-year students reporting using these tools for a wider range of coding tasks compared to first-year students.

As the accessibility of generative Artificial Intelligence (AI) tools increases, the relevance of traditional Computer Science (CS) education is being reevaluated. Generative AI tools built on large language models like *ChatGPT* and *Github Copilot* can not only "read" and "write" code in multiple programming languages, but also provide detailed explanations for each piece of code based on natural language queries. Given these advances, the CS education research community is contemplating what CS education should look like now (e.g., Becker et al., 2023). Without concurrence on a clear path for integrating generative AI into CS education, studies highlight how generative AI tools might support learning to code—alleviating programmer's writer's block, explaining algorithmic concepts, clarifying error messages; studies showcase challenges including over-reliance, ethical concerns, harmful bias, sustainability, and hallucinations (Becker et al., 2023). There is emerging evidence on roles for these tools to support learning to code (e.g., Liu et al., 2024; Jacques et al., 2023; Wang et al., 2024). Few studies investigate how students integrate these tools in their learning practices. Insights into how students at various learning stages use these tools can inform novel, participatory design studies (Gomez et al., 2018) focusing on specific needs of students and teachers. Such insights could even contribute to envisioning a generative AI-supported CS undergraduate curriculum (Becker et al., 2023). This paper explores uses of these tools by CS students across three academic years. We surveyed first to third-year undergraduate CS students at a French university to address the following questions: (1) Which generative AI tools are favored by CS students in different academic years? (2) For what coding tasks are students using these tools across academic years?

Method

We employed a convenience sample, recruiting students enrolled in an Institut Universitaire Technologique (IUT) in France. IUTs are selective institutions attached to universities designed primarily to prepare students for the job market. We targeted the entire population of CS students at the IUT (N = 121): 52 first-year undergraduates (of whom 6 were females), 41 second-year undergraduates (3 females), and 36 third-year undergraduates (2 females). With instructors' agreement, students were invited by the first author to anonymously answer questions using a Padlet virtual post board. Students used their own devices to access the survey. Each of the following questions opened a post column in which students could add their answers or upvote other answers: *(1) What AI tools are you using? (2) For what tasks have you sought assistance from AI to enhance your undergraduate education?* Students were given 15 minutes to answer these questions. Each year level was provided with an empty canvas. To analyze the data, we exported it from Padlet to Excel and then merged similar answers (e.g., chatGPT & ChatGPT), filtered out usages unrelated to stereotypical CS tasks (e.g., "image generation" or "tell a story"), and used descriptive statistics to report the findings.

Findings & Discussion

Globally, ChatGPT (59%) was the most reported tool across all years, followed by Dall-E (18%) and Midjourney (16%). GitHub Copilot came in fourth (15%), followed by Deepl (7%), Google Bard (now Gemini) (5%), and Bing (2%). Students mentioned 16 other tools (e.g., You.com, Chatsonic, Canva, Writesonic, Whisper, TomeAI). Focusing on the most reported text tools (ChatGPT for text and GitHub Copilot for coding assistance), use of ChatGPT was particularly high in the third year, with 81% of students reporting its use. This is in contrast to the second and first years, where reported usage was 56% and 37%, respectively. Similarly, students in the third year reported a higher usage of GitHub Copilot (33%) compared to those in their second year (15%). Notably, there was no reported usage of GitHub Copilot among first-year students.

First-year students did not report using generative AI for traditionally recognized coding tasks. Only one student reported using generative AI for debugging code. For second years, the most commonly reported application was code generation (27%), followed by brainstorming architecture design (22%), debugging (17%), and explaining existing code (10%). Development and code help applications were each reported at 5%. Least reported uses were for optimizing code and generating code and documentation, each at 2%. In the third year, analyzing programming errors (33%) was the most common reported use, followed by code generation (22%), improving coding efficiency (19%), and understanding new concepts (17%). Generating complex functions, explaining existing code, and automating repetitive coding tasks were each at 3%.

Three key findings stand out: (1) Students are actively using generative AI for a number of coding-related tasks, (2) Second and third-year students reported higher usage of generative AI tools, including more specialized tools; (3) Second and third-year students reported a wider range of coding-related uses for these tools than first year students. These variations may be driven by a combination of factors, including student needs, growing awareness of tools and capabilities, development environment preferences, learning curves, and potential access restrictions. More experienced coders tackling more complex projects might seek out these tools for both low level (e.g., debugging) and higher level implementation tasks (e.g., brainstorming architecture design). In contrast, the traditional first-year CS curriculum, with its focus on basic programming principles and writing low-level code, may not provide opportunities for students to leverage these tools beyond a limited set of purposes. A potentially greater awareness of these tools' capabilities and limitations could lead more advanced students to explore specialized coding assistants. Similarly, a preference for tools that integrate seamlessly with their preferred integrated development environments (IDEs), such as Copilot, could explain variations in the reported rates of its use. First-year students may also face a steeper learning curve when using IDE-embedded tools rather than conversational interfaces like ChatGPT. Accessibility (e.g., varying costs) could also play a role.

Overall, the adoption of generative AI tools across academic years reflects a student-led shift towards more interactive and assisted learning environments in undergraduate CS. Although our sampling strategy limits the generalizability of these findings, we conjecture that pedagogical approaches in first-year CS miss opportunities for students to deeply explore these tools for coding. This oversight is concerning given the research community's calls for reevaluating teaching strategies to integrate these tools early in the curriculum. Such integration is crucial for maintaining the relevance and effectiveness of pedagogical approaches, especially considering students' active use of these tools and the evolving role of programmers. As generative AI tools handle simpler tasks, growing emphasis on aspects such as code quality assurance underscore the importance of early tool use. Vital consideration should be given to how coding assistants built on LLMs can transcend mere utility, not only aiding in faster, error-free code writing but also acting as cognitive tools to foster coding skills. Drawing on Pea and Kurkland's (1987) work on cognitive technologies for writing, we'll need to design and research tools powered by LLMs more closely linking coding activities with thinking activities. Good developmental writing/coding environments will serve the needs not only of skilled writers/coders, but of *becoming* writers/coders (Pea & Kurland, 1987, p. 280). Adopting this developmental technologies approach, cinching up awareness of available tools early serves a more important goal: their introduction precisely when they become useful for the developmental trajectories of CS learners. This timing could accelerate their overall competency development and potentially attenuate interindividual variability in CS programming success. As Pea and Kurland predicted, "the goal of symbiosis—how to best create the productive union of humans and machines to serve our developmental advancements —will become our critical research topic" (p. 314). With Generative AI, we are here now.

References

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023, March). Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 500-506).

Gomez, K., Kyza, E. A., & Mancevice, N. (2018). Participatory design and the learning sciences. In *International handbook of the learning sciences* (pp. 401-409). Routledge.

Jacques, L. (2023). Teaching CS-101 at the Dawn of ChatGPT. *ACM Inroads*, *14*(2), 40-46.

Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., & Malan, D. J. (2024). Teaching CS50 with AI.

Pea, R. D., & Kurland, D. M. (1987). Cognitive technologies for writing development. In L. Fraser (Ed.), *Review of Research in Education*, 14, 71- 120. Washington DC: AERA Press.

Wang, S., Mitchell, J., & Piech, C. (2024, March). A large scale RCT on effective error messages in CS1. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1395-1401).