Teaching IT Software Fundamentals: Strategies and Techniques for Inclusion of Large Language Models

Strategies and Techniques for Inclusion of Large Language Models

Sharon Gumina*
Department of Integrated Information Technology, College of Engineering and Computing, University of South Carolina
gumina@cec.sc.edu

Travis Dalton
Department of Integrated Information Technology, College of Engineering and Computing, University of South Carolina
daltont@email.sc.edu

John H. Gerdes Jr.
Department of Integrated Information Technology, College of Engineering and Computing, University of South Carolina
gerdes@mailbox.sc.edu

ABSTRACT

This paper argues for the inclusion of tools that utilize Artificial Intelligence (AI) Large Language Models (LLMs) in information technology (IT) undergraduate courses that teach the fundamentals of software. LLM tools have become widely available and disrupt traditional methods for teaching software concepts. Learning objectives are compromised when students submit AI-generated code for a classroom assignment without comprehending or validating the code. Since LLM tools including OpenAI Codex, Copilot by GitHub, and ChatGPT are being used in industry for software development, students need to be familiar with their use without compromising student learning. Incorporating LLM tools into the curriculum prepares students for real-world software development. However, students still need to understand software fundamentals including how to write and debug code. There are many challenges associated with the inclusion of AI tools into the IT curriculum that need to be addressed and mitigated. This paper presents strategies and techniques to integrate student use of LLM tools, assist students' interaction with the tools, and help prepare students for careers that increasingly use AI tools to design, develop, and maintain software.

CCS CONCEPTS

• **Social and professional topics** → Professional topics; Computing education; Computing education programs; Information technology; • **Computing methodologies** → Artificial intelligence.

KEYWORDS

AI, Artificial intelligence, ChatGPT, Code generation, Copilot, GitHub, GPT-4, Information technology, IT, Large language model, LLM, OpenAI, Programming, Software development, Software fundamentals,

*Corresponding author.

SIGITE '23, October 11–14, 2023, Marietta, GA, USA

ACM ISBN 979-8-4007-0130-6/23/10.
https://doi.org/10.1145/3585059.3611409

ACM Reference Format:
Sharon Gumina, Travis Dalton, and John H. Gerdes Jr.. 2023. Teaching IT Software Fundamentals: Strategies and Techniques for Inclusion of Large Language Models: Strategies and Techniques for Inclusion of Large Language Models. In *The 24th Annual Conference on Information Technology Education (SIGITE '23), October 11–14, 2023, Marietta, GA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3585059.3611409

1 INTRODUCTION

Over the years, software developers have used various tools for code generation, versioning, analysis, debugging, and testing. Recently, the widespread emergence and integration of AI LLMs into these tools further enhances the automation of these tasks for the developer [1]. LLMs are a subset of AI that are trained in an unsupervised way on exceptionally large datasets [2]. LLMs use this training to improve their processing of natural language inputs including user prompts to produce responses including software artifacts such as test cases, documentation, code, and more. Tasks such as generating code, debugging, documentation, and software testing can be facilitated by the LLM tool [1]. A recent study finds that the model-generated code is comparable in complexity and readability to that written by human programmers [3]. While LLM tools do not eliminate the need for experienced programmers, they can speed development and make the process of developing code easier [4]. Since LLM tools are being used in industry it is both appropriate and necessary to cover their use in the IT curriculum.

The use of LLM tools is attractive to students because students view software development as challenging and often spend more time debugging their applications than developing them [5]. AI language models are sophisticated and provide various ways to support the concepts presented in a software fundamentals course including concept explanations, code examples, error handling, and debugging, algorithmic thinking, pseudocode and flowchart development, tool guidance, best practices, and resources and references [5]. Modern Integrated Development Environments (IDEs) such as Visual Studio Code and PyCharm include LLM extensions that provide AI-assisted coding suggestions, contextual documentation, and real-time feedback on programming errors. The ChatGPT tool uses the Codex model which can quickly generate, debug, and describe appropriate code for traditional first-year programming assignments [5–7]. As the LLM technology matures, these tools should be able to correctly handle even more difficult and complex problems.

Table 1: Software Fundamentals - Competencies for IT Undergraduate Degree Programs

Software Fundamentals - Competencies [8]	Potential Applications for LLM Tools [7, 9]
Use multiple levels of abstraction and select appropriate data structures to create a new program that is socially relevant and requires teamwork. (Program development)	Clarification of requirements for data, operations, and performance Explanation of data structures Comparison of data structures Suggestions for object-oriented design Responses including examples and use cases
Evaluate how to write a program in terms of program style, intended behavior on specific inputs, correctness of program components, and descriptions of program functionality. (App development practices)	Evaluation of program style
Develop algorithms to solve a computational problem and explain how programs implement algorithms in terms of instruction processing, program execution, and running processes. (Algorithm development)	Algorithm generation Explanation of instruction processing, program execution and running processes from an algorithm or code
Collaborate in the creation of an interesting and relevant app (mobile or web) based on user experience design, functionality, and security analysis and build the app's program using standard libraries, unit testing tools, and collaborative version control. (App development practices)	Code generation to solve a particular problem in supported languages. Includes most popular languages such as Python, Java, C++, C#, Swift, R, PHP, HTML/CSS, Perl

It is reasonable to assume that students will use tools such as ChatGPT to generate solutions for their assignments. Unfortunately, when students rely on an LLM tool and submit the generated solutions without personally doing any of the validation and verification, it circumvents the learning objectives of the assignment. As student use of LLM tools increases, it is necessary to change how software fundamentals are taught and assessed to achieve the desired learning objectives. Responding to this disruptive technology warrants discussion of strategies and techniques that might be used to promote student learning and help mitigate over-reliance on AI that erodes learner engagement and student knowledge of core software principles.

2 BACKGROUND

The Association for Computing Machinery (ACM) identifies curriculum guidelines for IT undergraduate degree programs [8]. In the guidelines, essential competencies for IT undergraduate degree programs are identified. Table 1 Software Fundamentals – Competencies for IT Undergraduate Degree Programs lists the essential IT competencies for software fundamentals and potential applications for LLM tool assistance [7, 9]. These competencies are impacted by the emergence of AI language models which generate real-time documentation, explanations, summaries, and code in response to user prompts and queries

3 CHALLENGES OF AI IN IT EDUCATION

The United Nationals, Educational, Scientific, and Cultural Organization (UNESCO) recommends a human-centered approach for the inclusion of AI within education. [10, 11]. A human-centered approach emphasizes the ethical and societal dimensions of AI, and promotes its use to prioritize key principles, including inclusivity and equality, respect for human rights and values, transparency and explainability, learner empowerment, and well-being, and collaboration and multi-stakeholder engagement [10]. In the following sections the challenges of AI inclusion within software coursework are mapped to the key principles of UNESCO's human-centered approach.

3.1 Inclusivity and Equality

Over the past year, access to LLM tools has steadily improved. Pricing impacts inclusivity and equality in education. OpenAI, the developer of ChatGPT, offers different pricing models and options including a free access version, subscription plans, and pay-as-you-go options for its products, including ChatGPT. According to OpenAI, their GPT-4 tool surpasses ChatGPT in its reasoning capabilities [12], but, at the time of this paper, GPT-4 is not free to users. GitHub's Copilot is available to programmers at a monthly fee and available to verified students for free [13].

3.2 Respect for Human Rights and Values

The learner's data and privacy are linked to respect for human rights and values, particularly in the context of data protection and privacy rights. Due to the rapid emergence of AI language models, they may not fully safeguard the learner's data and privacy as they interact with the LLM tool. Students may forget to safeguard their personal information while using these tools. It is important that users of AI-powered systems do not share personal information about themselves or others. Data that contains personally identifiable information (PII) and personal health information (PHI) should not be shared with an AI tool. AI systems may perpetuate bias or discrimination in their responses and recommendations [14].

3.3 Transparency and Explainability

ChatGPT lacks the context in which students are learning software fundamentals. AI-generated explanations of a software concept

may be given at a more advanced level, and hinder student learning [5, 14]. AI language models are trained on code repositories that may not be suitable for student learning. For example, in response to an introductory coding problem, the AI's generated code might use advanced concepts such as pointers, or recursion. Also, the styles and approaches to coding may not fit with the style and approach of the course [5, 7]. Students who are unfamiliar with the tool may have difficulty understanding different forms of AI code suggestions which would hinder the learner's ability to understand how the code works and why it was selected [5]. One study determined that AI-generated code was more difficult for users to debug than code they had written [15].

Publicly available code often requires any user to cite the original source, but code sources are not transparent in code generated by an AI language model [5]. While the LLM model might provide documentation of the code explaining its functionality, there is a lack of transparency in the training data used in its explanation.

3.4 Accountability and Responsibility

Research findings suggest that students can easily generate working code in a programming language of their choice using ChatGPT [16], but this code may not meet assignment requirements [6]. Students may not feel responsible for validating and verifying that the code runs properly, is logically correct, and meets the assignment's rubric. When code is automatically generated by an LLM tool used by students, learners may not feel personally accountable for the quality of the generated code.

ChatGPT generates its answers by synthesizing its training data and often produces responses similar to existing sources which may lead to plagiarism [14]. Similar responses submitted by students when they use AI language models may undermine student accountability and responsibility. There are concerns about online assessment security and cheating which erodes academic integrity [14].

3.5 Learner Empowerment and Well-Being

The focus of this principle is developing the student's skill and knowledge to the point that they can generate correct code. The LLM tool should be utilized as a programming aid that assists the student's development of a software application. The student must understand software development principles, be able to read and understand the logic of the code generated by the tool and have the skills to make needed adjustments.

One challenge to learner empowerment and well-being is over-reliance on AI models for lab and assignment completion. Lack of contextual understanding within AI models may lead to generated responses that do not meet the assignment requirements [6, 17]. Solutions generated with LLM tools may contain logic or syntax errors [5, 6]. One study determined that ChatGPT generated correct answers to 55.6% of questions and was not able to assess the correctness of its responses [6]. Generated code may contain vulnerabilities and/or bias introduced by the AI model training data [5]. Learner over-reliance on tools such as ChatGPT may cause a decline in student creativity, critical thinking, reasoning, and problem-solving skills [18], and students may not be motivated to analyze and solve the problem on their own [14].

3.6 Collaboration and Multi-Stakeholder Engagement

Student use of LLM tools enhances their understanding of human-to-machine interaction but does not substitute for human-to-human interaction. Software development, especially large-scale software development, requires teamwork – the interactions between different people including the users, project managers, system analysts, system developers, programmers, and testers. A course in software fundamentals needs to include learning outcomes associated with the collaborative nature of software development and educate students on how multi-stakeholder engagement influences the development process.

4 BENEFITS

AI LLM tools pose challenges for an existing IT curriculum. Despite these challenges, the models are widely used [19], and banning their usage is not a practical approach when courses are available asynchronously and online in higher education. These tools are also being used in industry [1], so it is important that students are familiar with them. A more practical approach is to embrace these technologies, thereby providing opportunities for teaching and learning software fundamentals with the assistance of AI.

4.1 Real-World Relevance

AI language models, including ChatGPT, are technologies disrupting real-world software development. Including these AI tools in software coursework exposes students to emerging software technologies and better prepares them for a dynamic workplace [15, 20]. No-code/low-code platforms used in software development offer a 'what you see is what you get (WYSIWYG) environment with a drag-and-drop functionality. These platforms increasingly include AI models such as AI Builder in Microsoft Power Apps [21].

4.2 More focus on Software Theory and Problem-Solving

A key component of IT software coursework includes programming which is not easy for most students to learn [5, 18]. AI models such as GitHub's Copilot assist novice programmers with code suggestions, examples, and explanations that reduce the cognitive load on students learning to program [18]. In one study, students using AI models for their programming assignments were successful if they provided accurate prompt descriptions and validated the code before submission [18]. AI models may reduce task completion time on programming assignments with reduced learner stress and discouragement [18]. Reduced task completion time writing and debugging applications allows educators to focus more on software theory and problem-solving rather than the syntax of a particular language.

4.3 Self-Directed Learning

Self-directed learning empowers students and provides autonomy as they learn and complete assignments. Students may approach interaction with an AI model in a myriad of self-directed approaches. The student may interact with AI language models by prompting the tool for question answers, code generation, code debugging,

and code explanations [22]. Learners design the questions for the language model. Students may discover that their prompts need to be augmented or decomposed into multiple sub-prompts [23].

Early in the process, a student may use a tool such as Copilot or ChatGPT to explore options for developing code. Later in the process, a student may paste code that contains syntax errors and prompt the AI model to identify the errors and correct them [15]. Learner interaction with an AI model during software debugging has tremendous potential to reduce frustration and task completion time.

4.4 Exposure to AI

Exposure to AI tools allows students and educators to obtain new competencies and ways of addressing human-computer interaction. Simply asking a query of an AI language model does not guarantee a correct answer. Language model users need to be aware of context limitations when they use these models. Queries may need to be decomposed and information on the context provided to the AI model to receive quality responses. Any response needs to be validated and tested for correctness due to the limitations of the AI training data [19].

5 STRATEGIES AND TECHNIQUES

Educators may include strategies and techniques that incorporate group projects, hands-on activities, and demonstrations to foster the benefits of LLM models and mitigate the pitfalls. AI language tools can create quick solutions that contribute to student learning if they are used appropriately. Learners can achieve better results if they query effectively, validate, and verify the results, and reflect on the outcomes. Below are suggested strategies and techniques that can help to effectively integrate AI LLM tools into a software fundamentals class.

5.1 Develop a Policy for the Inclusion of AI

Presenting a policy for AI model usage is important in both the course syllabus and assignments. This sets the guidelines for the use of LLM technologies in the course. This paper suggests the addition of phrases such as the following:

- It is the student's responsibility to examine and evaluate the information generated by AI tools for relevance and accuracy. It is the student's responsibility to validate generated code to verify that it is free of errors, including syntactic and semantic errors
- It is the student's responsibility to verify that the software meets rubric requirements for items such as file types, input/output, data types, program style, and documentation
- If an AI tool is used in an assignment, the student should explicitly acknowledge the use of the tool and describe how it was used for the assignment.
- If an AI tool is used in an assignment, the audit trail of queries/prompts and responses should be included as part of the assignment.
- It is the student's responsibility to safeguard their personal information and the personal information of others when they use an AI tool.
- Students should be able to comprehend and reflect on any AI-generated information as part of their assignment.

5.2 Ensure Equal Access to The Tools

Ensuring equal access to LLM tools requires that adequate resources are provided to all the students. For example, software development tools currently installed on lab machines can be updated with an AI tool extension such as adding GitHub Copilot software to Visual Studio Code. The syllabus should describe the technology requirements and resources available for the use of these AI model tools.

5.3 Explain and Demonstrate the Benefits and Pitfalls of AI models in Software Development

There are strategies that can be designed into a course to better educate learners on the benefits and pitfalls of using AI for their assignments. Learners should understand that an AI response might be incorrect due to limited context understanding or suboptimal prompting. One study identified incorrect solutions generated by Copilot due to the tool's lack of contextual understanding [7]. This lack of contextual understanding stemmed from the details of the assignment which included parsing a text file that was custom to the assignment. AI Tools such as Copilot can be a frustrating experience because they often do not understand learner instructions to fix or improve the generated code unless the user prompts are extremely specific [24].

One strategy might be the inclusion of a series of class labs where students use LLM tools to foster ideas and explore a particular concept such as encapsulation. These results of the exploration can be discussed in the class and suggestions gathered for how to better interact with the LLM tool.

Designing assignments that explore the limitations of the LLM tools can educate learners about the pitfalls of using AI. Assignments, such as creating a static website that requires multiple file types such as HTML files, CSS files, and JavaScript files, challenge students to link the files successfully. This assignment can be used to demonstrate the limitations of AI models when the assignment becomes more complex.

Successful techniques for interacting with LLM tools may be demonstrated by the Instructor during class activities and exercises to better prepare students for their use. Demonstrations can be conducted in class or attached as a video demo as assignments are made available.

5.4 Require a 3-Step Approach to Assignments

One of the cited studies recommends a three-step approach to developing programming applications in conjunction with an AI tool. These steps include system decomposition, functional programming, and program refinement [23].

- **System Decomposition** breaks the problem into a sequential list of tasks. This breakdown is done repeatedly until the algorithm is decomposed into sub-tasks using control structures such as iteration and decision making. System decomposition is more useful for AI prompting and produces increased quality of code [23].

- **Functional Programming** determines the function attributes including name, argument list, and description. Functional programming is a widely accepted programming paradigm where programs can be broken down into functions.
- **Program Refinement** includes the execution and testing of the programming application to determine the syntax and semantic errors. AI language models are more prone to semantic errors [23] so learners should validate the results of the program using the debugger.

5.5 Include Projects with Unique Contexts and Personalization

Over-reliance of the learner on tools can result in a decrease in their creativity and problem-solving skills [18]. One of the IT competencies identified by ACM for IT software fundamentals includes student collaboration in the creation of an interesting and relevant mobile or web app [8]. Including projects with unique contexts and learner personalization fosters student creativity and problem-solving. For example, an assignment that requires a Python program that draws a triangle from three integer inputs is easily generated by a tool such as ChatGPT. This type of assignment does not foster creativity because the generated code may be the same for students. Including a group project where students design the program requirements and develop the appropriate algorithm, data structures, modules, and user interface gives them opportunities to explore tasks required in developing software, not the least of which is teamwork. They make decisions that are unique to their project and foster creativity.

5.6 Require Validation and Verification of Code

There are multiple methods of validation and verification of code that can be used in a course. Students that use LLM-generated code should place the generated code into an IDE, such as Visual Studio Code, and use the integrated debugger to toggle a breakpoint on the first line of the main routine and step through the code to make sure that all the code is utilized and is semantically and syntactically correct.

In a web systems course, students can use the debugger within VS Code or external tools such as the W3C Markup Validation Service to validate code, such as HTML and CSS. By doing so, students assume responsibility for the code and recognize that it may have deficiencies such as logic errors, syntax errors or other important issues such as accessibility and others.

5.7 Require Student Demonstration and Explanation

Student reflection is critical in understanding the software they have developed including the data structures selected, the program style, the control structures, and how the program solves a computation problem. Requiring learners to demonstrate and explain the software they have developed allows them to reflect on what they have learned. One option is for the students to record a video showing their running code. These demonstrations and explanations can be used in the assessment to determine if the student has full comprehension of the software they have created. Full comprehension is required in a student demonstration and explanation of the resulting software. This explanation might include the intended behavior, specific inputs, program components, and functionality.

6 CONCLUSION

This paper advocates for the integration of AI LLM tools into software fundamental courses for the simple reason that they are being used in industry, and that they will only become more powerful with time. Employers will expect that students not only know how to code, but also that they know how to effectively use various tools, including AI LLM tools, so it is important to cover these tools in the IT curriculum.

While AI tools can generate code, they cannot yet replace an experienced software developer. They do represent a useful tool that can facilitate code development, but the user of any tool needs to know how to use the tool and its limitations. Since the generated code may not meet the requirements, the user must be able to fall back on fundamental concepts and experiential knowledge to validate and verify the generated code.

The use of AI is disrupting how IT software fundamental courses are taught. This paper advocates for the proactive consideration and use of strategies and techniques to encourage student learning in an evolving software development environment where LLM tools are easily accessed by both students and educators.

REFERENCES

[1] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé Bissyandé. 2023. Is ChatGPT the Ultimate Programming Assistant–How far is it? arXiv:2304.11938. Retrieved from https://arxiv.org/abs/2304.11938

[2] Abeba Birhane, Atoosa Kasirzadeh, David Leslie, and Sandra Wachter. 2023. Science in the age of large language models. *Nature Reviews Physics*, 5 (April 2023), 277-280. DOI: https://doi.org/10.1038/s42254-023-00581-4

[3] Naser Al Madi. 2022, October. How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot. In 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10 – 14, 2022, Rochester, Michigan. ACM Inc., New York, NY, 1-5. https://doi.org/10.1145/3551349.3560438

[4] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv:2303.07839. Retrieved from https://arxiv.org/abs/2303.07839

[5] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. (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 (SIGCSE 2023), March 15-18, 2023, Toronto ON Canada. ACM Inc., New York, NY, 500-506. https://doi.org/10.1145/3545945.3569759

[6] Sajed Jalil, Suzzana Rafi, Thomas D. LaToza, Kevin Moran, and Wing Lam. 2023. Chatgpt and software testing education: Promises & perils. arXiv:2302.03287. Retrieved from https://arxiv.org/abs/2302.03287

[7] Ben Puryear and Gina Sprint. 2022. Github copilot in the classroom: learning to code with AI assistance. Journal of Computing Sciences in Colleges, 38(1), (Nov 1, 2022), 37-47. Retrieved from https://dl.acm.org/doi/abs/10.5555/3575618.3575622

[8] Computing Curriculum. 2020 Paradigms for Global Computing Education. (June 2021). Retrieved May 28, 2023 from https://dl.acm.org/doi/book/10.1145/3467967

[9] Som Biswas. 2023. Role of ChatGPT in Computer Programming.: ChatGPT in Computer Programming. Mesopotamian Journal of Computer Science (February 2023), 8-16. DOI: https://doi.org/10.58496/MJCSC/2023/002

[10] Fengchun Miao, Wayne Holmes, Ronghuai Huang and Hui Zhang. 2021. AI and education: A guidance for policymakers. UNESCO Publishing.

[11] Renate Andersen, Anders I. Mørch, and Kristina Torine Litherland. 2022. Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education. Behaviour & Information Technology, 41, 9 (July 2022), 1830-1847. DOI: https://doi.org/10.1080/0144929X.2022.2083981

[12] OpenAI. GPT-4 is OpenAI's most advanced system, producing safer and more useful responses. Retrieved May 28, 2023, from https://openai.com/product/gpt-4

[13] Github. GitHub Copilot is generally available to all developers. Retrieved May 28, 2023, from https://github.blog/2022-06-21-github-copilot-is-generally-available-

to-all-developers/

[14] Mohammadreza Farrokhnia, Seyyed Kazem Banihashem, Omid Noroozi, and Arjen Wals. 2023. A SWOT analysis of ChatGPT: Implications for educational practice and research. Innovations in Education and Teaching International, (April 2023), 1-15. DOI: https://doi.org/10.1080/14703297.2023.2195846

[15] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded copilot: How programmers interact with code-generating models. Proceedings of the ACM on Programming Languages, 7(OOPSLA1), 85-111. DOI: https://doi.org/10.1145/3586030

[16] Basil Qureshi. 2023. Exploring the use of chatgpt as a tool for learning and assessment in undergraduate computer science curriculum: Opportunities and challenges. arXiv:2304.11214. Retrieved from https://arxiv.org/abs/2304.11214

[17] David Baidoo-Anu, and Leticia Owusu Ansah. 2023. Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. Available at SSRN 4337484. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id$=$4337484

[18] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23-28, 2023, Hamburg Germany. ACM Inc., New York, NY, 1-23. https://doi.org/10.1145/3544548.3580919

[19] Ahmed Tlili, Boulus Shehata, Michael Agyemang Adarkwah, Aras Bozkurt, Daniel T. Hickey, Ronghua Huang, and Brighter Agyemang. 2023. What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. Smart Learning Environments, 10, 1, (February 2023), 15. DOI: https://doi.org/10.1186/s40561-023-00237-x

[20] Burak Yetistiren, Isik Ozsoy, and Eray Tuzun, E. 2022. Assessing the quality of GitHub copilot's code generation. In Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE 2022), November 17, 2022, Singapore Singapore. ACM Inc., New York, NY, 62-71. https://doi.org/10.1145/3558489.3559072

[21] William Villegas-Ch, Joselin García-Ortiz, and Santiago Sánchez-Viteri. 2021. Identification of the factors that influence university learning with low-code/no-code artificial intelligence techniques. Electronics, 10, 10 (May 2021), 1192. DOI: https://doi.org/10.3390/electronics10101192

[22] Alin Zamfiroiu, Denisa Vasile, and Daniel Savu. 2023. ChatGPT–A Systematic Review of Published Research Papers. Informatica Economica, 27, 1 (2023), 5-16. DOI: https://doi.org/ 10.24818/issn14531305/27.1.2023.01

[23] Hao Bai. 2022. A Practical Three-phase Approach To Fully Automated Programming Using System Decomposition And Coding Copilots. In Proceedings of the 2022 5th International Conference on Machine Learning and Machine Intelligence (MLMI '22), September 23-25, 2022, Hangzhou China. ACM Inc., New York, NY, 183-189. https://doi.org/10.1145/3568199.3568228

[24] Michel Wermelinger. 2023. Using GitHub Copilot to Solve Simple Programming Problems. In Proceedings of the 54th ACM Technical Symposium on Computing Science Education (SIGCSE 2023), March 15-18, 2023, Toronto Canada. ACM Inc., New York, NY, 172.178. https://doi.org/10.1145/3545945.3569830

,