Automated Programming Exercise Generation in the Era of Large Language Models

Niklas Meißner
Institute of Software Engineering
University of Stuttgart
Stuttgart, Germany
niklas.meissner@iste.uni-stuttgart.de

Sandro Speth
Institute of Software Engineering
University of Stuttgart
Stuttgart, Germany
sandro.speth@iste.uni-stuttgart.de

Steffen Becker
Institute of Software Engineering
University of Stuttgart
Stuttgart, Germany
steffen.becker@iste.uni-stuttgart.de

2024 36th International Conference on Software Engineering Education and Training (CSEE&amp;T) | 979-8-3503-7897-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/CSEET62301.2024.10662984

***Abstract*—Lecturers are increasingly attempting to use large language models (LLMs) to simplify and make the creation of exercises for students more efficient. Efforts are also being made to automate the exercise creation process in software engineering (SE) education. This study explores the use of advanced LLMs, including *GPT-4* and *LaMDA*, for automated programming exercise creation in higher education and compares the results with related work using *GPT-3.5-turbo*. Utilizing applications such as ChatGPT, Bing AI Chat, and Google Bard, we identify LLMs capable of initiating different exercise designs. However, manual refinement is crucial for accuracy. Common error patterns across LLMs highlight challenges in complex programming concepts, while specific strengths in various topics showcase model distinctions. This research underscores LLMs' value in exercise generation, emphasizing the critical role of human supervision in refining these processes. Our concise insights cater to educators, practitioners, and other researchers seeking to enhance SE education through LLM applications.**

***Index Terms*—AI-Generated Exercises, Large Language Models, Programming Exercises, Software Engineering Education**

I. Introduction

The field of software engineering (SE) education continually seeks innovative approaches to enhance the learning experience and engage students in meaningful ways. Also, lecturers search for ways to improve their teaching and processes. Creating exercise sheets and assignments is often time-consuming for lecturers, and creating new exercises regularly can be cognitively exhausting. To reduce the effort of creating new exercises, the automatic question generation (AQG) concept was explored [1]. However, AQG research in the field of education is still narrow [2]. We started using large language models (LLMs) to generate self-assessment quizzes in software engineering education [3]. Nevertheless, there seems great potential for also creating programming exercises using AQG. In previous work, we introduced automated-generated programming exercises into the educational context using ChatGPT with *GPT-3.5-turbo* [4]. Recently, the emergence of AI has ushered in new possibilities for automating educational content creation [5], [6].

In this study, we explore the potential of LLM applications such as ChatGPT, Bing AI Chat, and Google Bard based on *GPT-4*, *LaMDA* in the context of programming exercise generation for higher education and compare against our previous *GPT-3.5-turbo* results [4] as baseline. Traditionally, creating programming exercises that effectively challenge and stimulate students' problem-solving skills has demanded considerable time and effort from lecturers. Integrating LLMs into this process introduces an intriguing opportunity to streamline exercise creation while maintaining the pedagogical integrity and envision a future where a significant portion of programming exercises can be generated with reduced manual intervention. This leads to our research questions for this study:

RQ1: *"How do ChatGPT with GPT-4, Bing AI Chat, and Google Bard perform in generating programming exercises compared to ChatGPT with GPT-3.5-turbo concerning the quality of the exercises and process simplicity?"*

RQ2: *"Which of the LLM applications is fitted best for lecturers to generate programming exercises?"*

The objective is to investigate the feasibility and effectiveness of using these LLM applications to create programming exercises. We intend to offer educators (e.g., lecturers), practitioners, and researchers a detailed perspective on the role of LLM applications in programming education. By analyzing the quality, diversity, and accuracy of the exercises produced, we aim to uncover the strengths and limitations of each LLM in this specific context and compare them to each other. Our investigation delves beyond the theoretical potential of LLMs and into the practicalities of their implementation. We scrutinize the exercise generation process to identify common patterns of error and challenges associated with automated generation. Furthermore, we explore the extent of manual intervention required to refine LLM-generated exercises, thus providing a comprehensive understanding of the dynamics between automated assistance and human oversight. By presenting empirical insights into the capabilities of these models, we hope to contribute valuable insights that inform decisions regarding integrating LLMs into programming curricula.

II. Generation of Exercises

A. ChatGPT with GPT-3.5-turbo as Baseline

In previous work, we investigated generating exercises for a programming course using ChatGPT with *GPT-3.5-turbo* [4]. We considered a beginner to intermediate-level programming

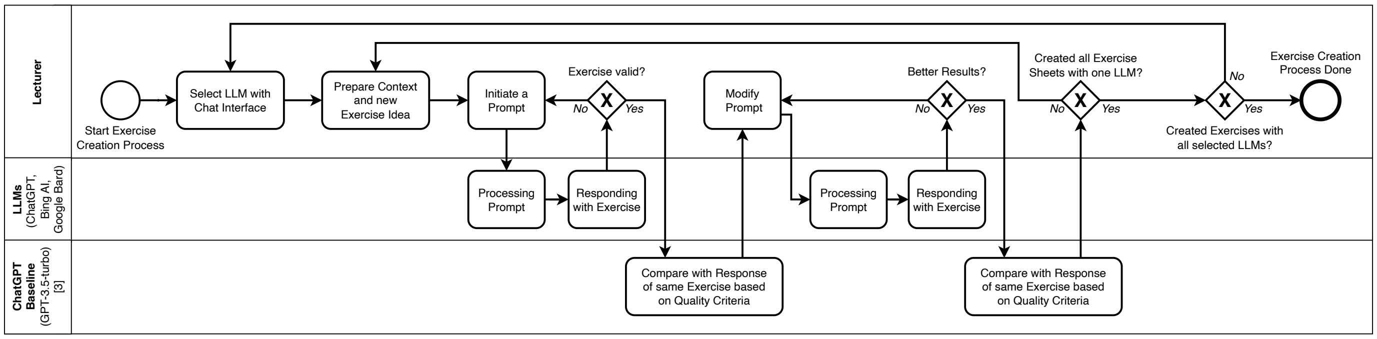
Lecturer
LLMs (ChatGPT, Bing AI, Google Bard)
ChatGPT Baseline (GPT-3.5-turbo) [3]
Start Exercise Creation Process
Select LLM with Chat Interface
Prepare Context and new Exercise Idea
Initiate a Prompt
Exercise valid?
No
Yes
Modify Prompt
Better Results?
No
Yes
Created all Exercise Sheets with one LLM?
No
Yes
Created Exercises with all selected LLMs?
No
Yes
Exercise Creation Process Done
Processing Prompt
Responding with Exercise
Processing Prompt
Responding with Exercise
Compare with Response of same Exercise based on Quality Criteria
Compare with Response of same Exercise based on Quality Criteria

Fig. 1. BPMN of our process to investigate the use of LLM applications for programming exercise creation.

course focusing on the objects-first approach [7], [8], hence we use the same course scenario in this study.

We created 12 exercise sheets[1] with multiple exercises, each for different topics in such a course. Each exercise sheet required minor manual edits to be applicable in an actual course. We evaluated the exercise sheets concerning their quality for student beginners and whether the students saw that the sheets were mostly AI-generated. The results showed that the generated sheets were helpful to the students and that they did not know the sheets were AI-generated [4].

The quality of generated exercises differs depending on the topic of the exercise. While exercises about control flow structures went extremely well, creating exercises that practiced identifying errors and classifying them correctly was very challenging [4]. In this study, we use the previous results with *GPT-3.5-turbo* as a baseline and examine other LLM applications to see if they achieve better results.

Therefore, we selected four different exercise topics for this study and selected the respective exercise sheets and prompts as a baseline against which to compare. The exercise topics we decided on are (i) exercises that provide the students a description of an API and tasks that the students should solve by using the API, (ii) exercises that practice different control flow structures, (iii) exercises with inheritance, generics and the strategy design pattern, and (iv) exercises that practice the understanding of different types of errors. We chose these topics as (i) is complex regarding a good design, (ii) went very well in the baseline, (iii) is very complex, and the baseline exercises had some semantic incorrectness, and (iv) went very bad in the baseline. In another paper, we investigated further exercise generation and comprehension of UML modeling exercises using *GPT-4* and *DALL·E* [9].

B. General Process

We followed a similar process compared to the baseline [4]. The process is depicted with a BPMN process in Figure 1. We started by selecting the LLMs we intended to test. The most important criterion was that the LLM had a chat interface through which lecturers could interact. We began each generation by providing the context and exercise topic in our initial prompt. Based on the first prompt, the LLM creates an initial exercise draft, which we review. We either reject the draft if the exercise is not valid or compare it with the exercise in the baseline. We then modify and specify the prompt with more context, a different scenario, or guide the LLM to what we expected. The LLM then generates an updated draft. If the design does not meet our expectations, i.e., it is a bad design, we start again with a different or better-specified context. If we achieve a better result, we then compare it again with the exercise in the baseline. We repeat the process until all exercise sheets are created with all selected LLMs.

The prompt engineering process was carried out step by step in accordance with Velásquez-Henao et al. [10], starting by defining the goal. The prompt was then designed using best-practice prompt patterns [11], the result was assessed using evaluation criteria [12] and then compared with each other. We used the prompt patterns [11] to design prompts such as:

"Act as a software engineering professor. I want you to create exercises for an advanced beginner programming course in Java about topics I will write in future messages. The exercises themselves should be formatted with markdown. The source code should be in Java. Multiple files are allowed. Please pay attention to good code style, i.e., follow clean code rules. Please create an exercise about inheritance with the strategy design pattern."

All prompts and LLM conversations are stored on Zenodo[2]. To determine the response quality, we evaluated the responses according to Lo's continuous evaluation criteria [12] for (1) accuracy, (2) relevance, and (3) completeness.

C. LLM Selection

As we have already investigated the general applicability of LLMs to generate programming exercise sheets [4], we are interested in how the different applications that use the same or different LLMs in the background perform this task. We only consider such LLM client applications that lecturers can use without programming any additional code, e.g., to conduct API calls. Therefore, we will not investigate the use of OpenAI's GPT API, Azure OpenAI service, Meta's LLAMA, etc., but focus on ChatGPT, Bing AI Chat, and Google Bard. As the baseline used ChatGPT with *GPT-3.5-turbo*, we focus on ChatGPT with *GPT-4* in this work. According to its

[1]https://github.com/spethso/Programmierprojekt/wiki

[2]https://doi.org/10.5281/zenodo.8298489

TABLE I
COMPARISON OF THE LLM APPLICATIONS AND THE BASELINE BASED ON QUALITY CRITERIA.

Exercise topic	Baseline (*GPT-3.5-turbo*)		ChatGPT (*GPT-4*)		Bing AI Chat (*GPT-4*)		Google Bard (*LaMDA*)	
	# of Prompts	*Quality*	*# of Prompts*	*Quality*	*# of Prompts*	*Quality*	*# of Prompts*	*Quality*
Control flow structures	few (1 – 4)	good	few (1 – 3)	good	few (1 – 4)	decent	few (1 – 3)	decent
API understanding and usage	decent (9)	good	few (4)	good	N.A.	N.A.	N.A.	N.A.
OOP - Inheritance, Generics, etc.	few (2 – 4)	decent	few (1 – 3)	good	few (1 – 3)	bad	few (2 – 3)	bad
Error finding and classification	many (?)	bad	few (4)	good	N.A.	N.A.	N.A.	N.A.

documentation, Bing AI Chat uses *GPT-4* as well. However, Google Bard uses Google's *LaMDA* model. As we could not generate exercises with the web version of Bing AI Chat, we used the version available in Skype.

D. Exercise Generation using Different LLMs

As mentioned, we use ChatGPT with *GPT-4*, Bing AI Chat with *GPT-4*, and Google Bard with *LaMDA* to generate exercises for the four topics described in subsection II-A. In the following, we describe our experiences and specific observations when using these applications. We describe the quality of the exercises and usability of the applications without describing every detail. Table I depicts a summary comparison of the three applications and the baseline for the four chosen topics. The table provides information on the number of prompts required to obtain a reasonably acceptable response from the LLMs. We considered up to five prompts to be "*few*", up to ten to be "*decent*", and everything above that to be "*many*". We could not determine the exact number of prompts for the baseline error-finding exercise. However, *GPT-3.5-turbo* required a large number of prompts and achieved poor results. We used the criteria described in subsection II-B to determine the quality of the generated exercises and summarized them as "*bad*", "*decent*" and "*good*". In the following paragraphs, the various exercise topics are analyzed in detail, and their quality is described:

1) Control flow structures: This exercise sheet aims to practice control flow structures. As the course focused on the objects-first approach, the exercises should follow an OO style, i.e., using classes and objects in the main method [4].

In the baseline, exercises for different control-flow structures were created with only a few prompts. While the exercises are of good quality, they often lack an object-oriented focus. Nevertheless, some exercises for if-else and switch-case fulfill this requirement and provide an acceptable OO design.

While ChatGPT with *GPT-4* created some standard control-flow exercises, e.g., Fibonacci numbers or password checking, it could also generate less typical tasks. However, most tasks lack an acceptable OO design and often use Java I/O to retrieve user input. Nevertheless, we could create most exercises within one to four prompts. Notably, ChatGPT with *GPT-4* often provided the control flow structures already in a start code, which had to be excluded by us via an additional prompt. We did not see this behavior in the baseline.

We were able to generate basic control-flow structure exercises with Bing AI Chat. However, the exercises did not follow any OO design and did not strictly vary from standard examples. Furthermore, Bing AI Chat used Java I/O even after explicitly stating not to use Java I/O.

Also, Google Bard primarily generated standard examples, e.g., grades and weekdays. We could not create control-flow structure exercises that follow an OO design.

In summary, all three applications can generate decent control-flow structure exercises without OO design. However, we experienced that ChatGPT outperforms Google Bard and Bing AI Chat in terms of creativity, OO design, scenarios, and exercise quality. Notably, we noticed that the baseline performed better than Google Bard and Bing AI Chat.

2) API understanding and usage: This exercise sheet aims to practice the understanding and usage of Java APIs. The exercise should provide a textual description of an API, i.e., not providing source code. The students should understand this API and use it to implement different scenarios or methods [4].

In the baseline, generating basic and acceptable exercises for this topic was possible with only a few prompts. However, more detailed change requests or extensions often required many prompts and would have been more easily injected into the exercise manually. The exercises were creative in design and API but often followed a similar structure.

Also, ChatGPT with *GPT-4* could create good-quality API exercises. In comparison to the baseline, we required fewer prompts for acceptable exercises. Nevertheless, ChatGPT with *GPT-4* created similar scenarios and APIs as the baseline.

Using Bing AI Chat, we could not create exercises for this topic that satisfy our requirements. The generated exercises provided the source code directly instead of an API description, even though we explicitly stated otherwise. Furthermore, instead of calling Java class APIs, some tasks expected the calling of remote APIs, which is not intended for a beginner to intermediate programming course.

Like Bing AI Chat, Google Bard could not create exercises for this topic. Google Bard did not have a class API to consume and use, but it had tasks to program an entire class based on a given description or tasks to call a remote API.

In summary, only ChatGPT with *GPT-3.5-turbo*, i.e., the baseline, and with *GPT-4*, could generate exercises that fulfill this topic's requirements. Using *GPT-4*, we experienced higher creativity and required fewer prompts.

3) Inheritance, polymorphism, and generics: This exercise sheet aims to practice more complex OO concepts such as inheritance, polymorphism, dynamic binding, and generics [4].

We also try to create an exercise for the strategy design pattern as an example for the composition over inheritance concept.

In the baseline, exercises for the complex OO concepts mentioned above were created successfully.
Depending on the concept, the quality varied. Some exercises looked correct on the first view but showed semantic incorrectness when trying to solve them. Using one prompt, the baseline could also create a strategy design pattern exercise. In general, the baseline required only a few prompts.

ChatGPT with *GPT-4* generated well-designed exercises for this topic's concepts. However, the exercises use standard examples similar to the baseline. In contrast to the baseline, ChatGPT with *GPT-4* did create Java Generics exercises that were not only a wrapper above Java Collections. Also, the strategy design pattern exercise was created using one prompt. We could not find semantic errors as in the baseline.

Bing AI Chat could also create decent exercises with standard examples for this topic. However, the tasks for Java Generics use either a generic pair class or Java Collections and are, thus, not creative. Additionally, the strategy design pattern exercise did not have a clear and straightforward structure.

Google Bard was able to create exercises for this topic. These exercises follow standard examples, and the Generics exercise uses Java Collections instead of interesting new scenarios. Furthermore, the Generics exercise was semantically broken as the methods contained infinite recursions and an attribute of the child class in a parent class. Google Bard could create a strategy design pattern exercise, but the exercise was generic and not creative with a concrete scenario.

In summary, we could create decent to acceptable exercises for this topic with all three applications similar to the baseline. However, all exercises were primarily created based on standard examples. Nevertheless, we experienced that ChatGPT with *GPT-4* offered the highest quality for the exercises.

4) Error finding and classification: This exercise sheet aims to practice identifying errors in existing source code and classifying them, i.e., mapping them to lexical, syntactical, static semantical, and dynamic semantical error classes [4].

In the baseline, there were difficulties creating correct and well-defined exercises on this topic. ChatGPT with *GPT-3.5-turbo* often created correct or nearly correct source code. In case of errors, the baseline often misclassified them. Especially for lexical errors, the baseline seemed not to know the concept.

While ChatGPT with *GPT-4*, in general, could create good exercises for this topic, it still lacks an understanding of lexical errors and misclassified errors of other types as lexical. However, after a short explanation, ChatGPT could apply a correct understanding to the generated exercises. Therefore, we could generate exercises for this topic in only a few prompts.

Bing AI Chat was unable to generate this specific type of exercise. It became apparent that the model was limited in constructing programming exercises that required error identification and correction. This finding underscores the complexity of accurately modeling syntactic and logical errors within code and highlights the challenges inherent in training LLMs to capture such intricacies.

While Google Bard generated exercises with correct source codes, unfortunately, it did not inject intentional errors into the code. This made the generated exercises mostly unusable.

In summary, only ChatGPT with *GPT-4* could create correct exercises with acceptable quality and amount of prompts.

5) Usability for lecturers: While the results were decent for most exercise topics, the applications' usability vastly differs.

Chat history and prompt editing: In ChatGPT, each prompt can be edited, creating a new branch in the chat history. Therefore, the resulting chats are not linear but follow a tree structure. This allows the lecturer to come back to a prompt and refine it to update the exercise's draft differently, which can be very helpful if some prompts lead to worse results than expected. Instead of repairing the draft within the same branch, the lecturer returns to an earlier prompt and continues creating the exercise. In contrast, Bing AI Chat and Google Bard offer only a linear chat history, and Google Bard does not allow any editing of prompts except for the last prompt, making changes in an earlier state difficult or impossible.

Structure: ChatGPT and Google Bard allow the lecturer to create a separate chat for each exercise sheet or single exercise. The lecturer can rename these chats to identify them more easily later. However, Bing AI Chat only offers one chat to work with, creating multiple exercises in the same history.

Sharing: ChatGPT and Google Bard offer links to share a chat history with others, e.g., exercises with other lecturers.

Output format: As in the baseline, we aimed to create our exercises in markdown format. While ChatGPT and Google Bard can generate exercises in markdown and render them within the chat, Bing AI Chat could not provide markdown. However, we have not tried other formats such as *LaTeX*.

Overall, in our opinion, ChatGPT offers the best usability to lecturers as it allows the most fine-grained editing while keeping the entire chat history as a tree-like structure, allowing them to switch between the branches. Using a separate chat for each exercise sheet and formatting is possible and allows renaming the chats. Furthermore, the exercises were formatted in markdown as requested and rendered. Google Bard and Bing AI Chat lack some criteria and do not offer additional features to improve their usability.

III. Lessons Learned and Threats to Validity

A. Lessons Learned

There are some lessons learned for generating programming exercises with LLMs: (1) ChatGPT outperforms the other LLM applications regarding results and usability. (2) Even though ChatGPT and Bing AI Chat both use *GPT-4*, the quality of the results differs a lot. (3) Google Bard offers confident explanations that often are incorrect, e.g., for error identification and classification. (4) The LLM applications often provide a starter code in the control flow structures exercise, already containing the control flow structure to practice. (5) Context-switches and editing of prompts work best in ChatGPT due to the branching of lecturer input and providing multiple chats. (6) Exercise scenarios are very similar in all applications.

B. Threats to Validity

We identified three main threats to validity. (1) In our study, we were investigating the use of the three applications only on four programming exercise topics, which we selected based on the experiences in the baseline study. (2) For each topic, we created around five exercises. Therefore, a larger study might be required to make stronger empirical-based assumptions about the quality of the generated exercises. (3) We did not evaluate the exercises' quality in an actual course but only by manual review. However, despite the stated threats, we believe that the lessons learned and insights presented are valuable and useful for educators, practitioners, and other researchers.

IV. Related Work

The research by Sarsa et al. [13] focused on exploring the capabilities of generating programming exercises with large language models, focusing on OpenAI Codex. Similar to this paper, the goal was to use these capabilities to generate programming exercises with associated elements such as sample solutions, test cases, and code explanations [13]. However, the authors only obtained results with OpenAI's outdated Codex model, where this paper analyzes the differences between the various large language models and presents recommendations for specific programming exercise types.

Nguyen et al. [14] used an approach to generate questions with LLMs. They aimed to generate and assess questions using *GPT-3*. Even though *GPT-3* is an outdated model, they also focused on higher education in the field of data science. They assessed the created questions using automatic scoring by Google *GPT-3* and manual review [14]. However, since we focus on the generation of exercises and differences in using state-of-the-art LLMs, the objectives of the studies differ.

The generation of programming exercises was also investigated by Lu et al. [15]. Their approach was to develop a question-generation system that combined semantic and syntactic elements. The main objective was to compare the quality of questions generated by the machine with those generated by experts [15]. However, compared to these related works, we focus on LLM applications accessible to lecturers and practitioners without having to build own custom generators. Moreover, we focus on more complex programming exercises in higher education, which the other works mostly did not do.

V. Conclusion

In this paper, we investigated the use of ChatGPT with *GPT-4*, Bing AI Chat, and Google Bard to generate exercises for a beginner to intermediate-level programming course scenario. We compared the results against each other and used ChatGPT with *GPT-3.5-turbo* as a baseline from related work regarding the quality of the exercises and usability. Our study focuses on four different exercise topics that went very well or poorly with *GPT-3.5-turbo*, were very challenging in design, or were complex concepts. Our results showed that ChatGPT with *GPT-4* fitted best, followed by ChatGPT with *GPT-3.5-turbo*. Google Bard provides good usability for lecturers but had worse results than ChatGPT with *GPT-3.5-turbo*. Bing AI Chat was cumbersome to use and did not provide particularly good results for the used input prompts. Educators, practitioners, and other researchers can use the results of this study to create own exercises using the LLMs mentioned, taking into account the strengths and weaknesses described here, and thus achieve the desired result more efficiently. In general, ChatGPT and Google Bard could easily generate good programming exercises. They especially help in creating exercise variants and good first drafts. However, in future work, LLMs without a graphical user interface, which may be specialized in creating exercises, could also be investigated. Nevertheless, generated exercises still usually require manual edits and cross-checking.

Acknowledgment

We acknowledge the support of the Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg (MWK, Ministry of Science, Research and the Arts Baden-Württemberg under Az. 33-7533-9-19/54/5) in Künstliche Intelligenz & Gesellschaft: Reflecting Intelligent Systems for Diversity, Demography and Democracy (IRIS3D) and the support by the Interchange Forum for Reflecting on Intelligent Systems (IRIS) at the University of Stuttgart.

References

[1] G. Kurdi, J. Leo *et al.*, "A Systematic Review of Automatic Question Generation for Educational Purposes," *International Journal of Artificial Intelligence in Education*, vol. 30, no. 1, pp. 121–204, Mar 2020.

[2] T. Steuer, L. Bongard *et al.*, "On the Linguistic and Pedagogical Quality of Automatic Question Generation via Neural Machine Translation," in *Technology-Enhanced Learning for a Free, Safe, and Sustainable World*. Cham: Springer International Publishing, 2021, pp. 289–294.

[3] N. Meißner, S. Speth, J. Kieslinger, and S. Becker, "EvalQuiz – LLM-based Automated Generation of Self-Assessment Quizzes in Software Engineering Education," in *Software Engineering im Unterricht der Hochschulen 2024*. Bonn: GI e.V., 2024, pp. 53–64.

[4] S. Speth, N. Meißner, and S. Becker, "Investigating the Use of AI-Generated Exercises for Beginner and Intermediate Programming Courses: A ChatGPT Case Study," in *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)*.

[5] B. du Boulay, "Artificial Intelligence as an Effective Classroom Assistant," *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 76–81, 2016.

[6] G.-J. Hwang, H. Xie *et al.*, "Vision, challenges, roles and research issues of Artificial Intelligence in Education," *Computers and Education: Artificial Intelligence*, vol. 1, p. 100001, 2020.

[7] S. Speth, "Teaching Object-Oriented Programming with the Objects-first Approach: An Experience Report," in *Proceedings der SEUH 2023*. Gesellschaft für Informatik, Bonn, 02 2023, pp. 35–42.

[8] B. Meyer, *Touch of Class*. Springer, 2009, vol. 51.

[9] S. Speth, N. Meißner, and S. Becker, "ChatGPT's Aptitude in Utilizing UML Diagrams for Software Engineering Exercise Generation," in *2024 IEEE 36th International Conference on Software Engineering Education and Training (CSEE&T)*.

[10] J. D. Velásquez-Henao, C. J. Franco-Cardona, and L. Cadavid-Higuita, "Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering," *DYNA*, vol. 90, no. 230, p. 9–17, Nov. 2023.

[11] J. White *et al.*, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT," 2023.

[12] L. S. Lo, "The CLEAR path: A framework for enhancing information literacy through prompt engineering," *The Journal of Academic Librarianship*, vol. 49, no. 4, p. 102720, 2023.

[13] S. Sarsa, P. Denny *et al.*, "Automatic generation of programming exercises and code explanations using large language models," in *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, ser. ICER '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 27–43.

[14] H. A. Nguyen, S. Bhat *et al.*, "Towards Generalized Methods for Automatic Question Generation in Educational Domains," in *Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption*. Cham: Springer International Publishing, 2022.

[15] O. H. T. Lu, A. Y. Q. Huang *et al.*, "Expert-authored and machine-generated short-answer questions for assessing students learning performance," *Educational Technology & Society*, vol. 24, no. 3, 2021.