

Using a Low-Code Environment to Teach Programming in the Era of LLMs

Anna Potriasaeva
JetBrains Research

Belgrade, Serbia
anna.potriasaeva@jetbrains.com

Yaroslav Golubev
JetBrains Research

Belgrade, Serbia
yaroslav.golubev@jetbrains.com

Katsiaryna Dzialets
JetBrains

Munich, Germany
katsiaryna.dzialets@jetbrains.com

Anastasiia Birillo
JetBrains Research

Belgrade, Serbia
anastasia.birillo@jetbrains.com



Figure 1: A possible UI for the proposed approach.

ABSTRACT

LLMs change the landscape of software engineering, and the question arises: *“How can we combine LLMs with traditional teaching approaches in computer science?”* In this work, we propose to teach students in a low-code environment of code generation, developing not only their coding but also decomposition and prompt skills.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ICER '24 Vol. 2, August 13–15, 2024, Melbourne, VIC, Australia

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0476-5/24/08

<https://doi.org/10.1145/3632621.3671429>

CCS CONCEPTS

• Computing methodologies → Artificial intelligence; • Social and professional topics → Software engineering education; • Human-centered computing → Interactive systems and tools.

KEYWORDS

Programming Education, MOOC, LLMs, Generative AI

ACM Reference Format:

Anna Potriasaeva, Katsiaryna Dzialets, Yaroslav Golubev, and Anastasiia Birillo. 2024. Using a Low-Code Environment to Teach Programming in the Era of LLMs. In *ACM Conference on International Computing Education Research V.2 (ICER '24 Vol. 2)*, August 13–15, 2024, Melbourne, VIC, Australia. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3632621.3671429>

1 APPROACH

Research shows that LLMs are crucial for teaching students both higher-level concepts and the practical skills of prompting that are increasingly important [4, 5, 7, 8]. The key idea of our approach is to use *intelligent prompt engineering* to teach algorithmic thinking and decomposition [6], while combining it with code generation and direct coding. Figure 1 shows a potential UI for this.

Firstly, the students learn the basics of task decomposition so that they can work with the program at the level of functions (1). Then, for the given task, the student describes the necessary algorithm in natural language using *intelligent prompt engineering* (2), meaning that the prompt may contain code if the student is already familiar with some programming concepts. The student can RUN a particular *description* block (2) to see the current version of the generated code in the *draft* section (3) and the errors in it. Importantly, the errors are highlighted in both blocks (4). In our vision, the student can only fix errors in the *description*, but we are considering making it possible in the *draft* section too. Finally, the student can CHECK their solution, generating all the code and launching the task's tests (5).

2 TECHNICAL DETAILS

For the pilot, we will focus on Kotlin [2], integrating our solution into the in-IDE learning format [3] of the JetBrains Academy plugin [1]. We will create a special domain-specific language to provide *description* and *draft* editor blocks, and use completion to help with the prompt. The IDE setting allows us to use *static analysis* to:

- (1) check whether students only use defined variables and functions in the code parts of the *description* block;
- (2) analyze the model's output to ensure that the code directly defined in *description* stays the same.

We will also create a grammar to teach the students to write concrete prompts (see (2)) and highlight if their prompt is too vague.

3 POSSIBLE EVALUATION

We plan a pilot evaluation with first-year bachelor students. Half the students will get the traditional MOOC experience, solving a given course with only code. The other half will get more exercises on task decomposition and prompting at the beginning, and then the same course with the proposed approach integrated. After completing the course, we will give both groups a separate task and compare their performance. The planned research questions are:

RQ1: What are the students' perceptions of the proposed approach?

RQ2: How does the proposed approach affect the speed of solving tasks?

RQ3: How does the proposed approach affect code quality?

REFERENCES

- [1] 2024. *JetBrains Academy Plugin*. Retrieved June 6, 2024 from <https://plugins.jetbrains.com/plugin/10081-jetbrains-academy>
- [2] 2024. *Kotlin*. Retrieved June 6, 2024 from <https://kotlinlang.org/>
- [3] Anastasiia Birillo, Maria Tigina, Zarina Kurbatova, Anna Potriasaeva, Ilya Vlasov, Valerii Ovchinnikov, and Igor Gerasimov. 2024. Bridging Education and Development: IDEs as Interactive Learning Platforms. *arXiv preprint arXiv:2401.14284* (2024).
- [4] Adrienne Decker, Mark Allen Weiss, Brett A Becker, John P Dougherty, Stephen H Edwards, Joanna Goode, Amy J Ko, Monica M McGill, Briana B Morrison, Manuel Pérez-Quiñones, et al. 2022. Piecing Together the Next 15 Years of Computing Education Research Workshop Report. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. 1051–1052.
- [5] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A Becker, and Brent N Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 296–302.
- [6] Gerald Futschek. 2006. Algorithmic Thinking: The Key for Understanding Computer Science. In *International conference on informatics in secondary schools-evolution and perspectives*. Springer, 159–168.
- [7] James Prather, Paul Denny, Juho Leinonen, Brett A Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, et al. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. 108–159.
- [8] Matti Tedre and Henriikka Vartiainen. 2023. K-12 Computing Education for the AI Era: From Data Literacy to Data Agency. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 1–2.