Leveraging KOALA for Programming Data Collection: A Half-Day Tutorial for Research Application

Daniil Karol, Ilya Vlasov, Katsiaryna Dzialets, Anna Potriasaeva, Anastasiia Birillo
JetBrains Research
{daniil.karol, ilya.vlasov, katsiaryna.dzialets, anna.potriasaeva, anastasia.birillo}@jetbrains.com

ABSTRACT

In the context of teaching programming, it is crucial for educators to have access to data regarding the learners' behavior within coding tools. Such data provides valuable insights into how students approach problem-solving and develop their programming skills, enabling improvements to teaching practices. Moreover, it plays a critical role in conducting research experiments, testing prototypes, and validating new ideas, providing a foundation for advancing educational tools and methods. However, many existing tools tend to be limited in terms of configurability and the types of data that can be collected, often presenting a significant barrier to entry for educators. In order to address these challenges, the KOALA tool was developed as a robust and configurable solution for collecting programming data within JetBrains integrated development environments (IDEs) developed by JetBrains. The aim of this tutorial is to present the KOALA tool, a plugin compatible with all JetBrains IDEs. This tool is designed to facilitate the collection of detailed student interaction data during the task-solving process, including step-by-step code changes and IDE events. Furthermore, it provides the capability to control and customize IDE settings, such as disabling features like auto-completion, enabling researchers to create controlled experimental environments tailored to specific studies. The tutorial will provide a practical demonstration of the tool's capabilities by utilizing it in an in-IDE programming course.

Keywords

programming education, in-IDE learning, activity tracking, code tracking

1. INTRODUCTION

The collection of programming data within development environments has recently become increasingly significant [4, 3]. Such data serves a variety of purposes, including improving development tools [14], analyzing user behavior and interactions with system features [11], and advancing programming education [8]. In educational contexts, data collection enables researchers and educators to explore how students approach problem-solving tasks, how they interact with programming tools, and how their programming skills develop over time. These insights are critical for improving programming education and creating better learning experiences.

While tools and datasets for collecting programming data in educational contexts are becoming increasingly available [12, 7, 3], significant gaps and limitations remain. Existing tools often lack flexibility in their configuration, making it difficult to customize the learning environment or track specific types of student behavior during experiments. For example, they may not allow educators to adjust settings such as code completion, syntax checking, or collection granularity.

To address these challenges, the KOALA tool [6] was recently developed as a powerful and flexible extension [10] for collecting programming data within JetBrains IDEs.[1] **The tutorial will focus primarily on the practical aspects, providing participants with a comprehensive overview of the tool, guidance on configuration options, and hands-on experience to customize the tool for their specific research needs.** To demonstrate the functionality of the configured tool and the nature of the data it collects, participants will engage in solving an in-IDE course designed in the recently introduced in-IDE learning format [2]. This hands-on experience will provide an in-depth understanding of one potential application of the KOALA tool, as well as highlight the opportunities offered by in-IDE courses for programming education. More importantly, the tutorial will equip participants with the knowledge required to leverage the tool not only for in-IDE course development but also for a wide range of research studies conducted within JetBrains IDEs.

2. BACKGROUND

This section provides an overview of the KOALA tool, which will be used in the tutorial. Additionally, it introduces the in-IDE learning format, which will serve as a practical example to demonstrate the data collection and the capabilities of the tool.

Daniil Karol, Ilya Vlasov, Katsiaryna Dzialets, Anna Potriasaeva, and Anastasiia Birillo. Leveraging KOALA for Programming Data Collection: A Half-Day Tutorial for Research Application. In Caitlin Mills, Giora Alexandron, Davide Taibi, Giosuè Lo Bosco, and Luc Paquette (eds.) Proceedings of the 18th International Conference on Educational Data Mining, Palermo, Italy, July, 2025, pp. 712–715. International Educational Data Mining Society (2025).

https://doi.org/10.5281/zenodo.15870316

[1] IntelliJ IDEA: https://www.jetbrains.com/idea/, PyCharm: https://www.jetbrains.com/pycharm/, CLion: https://www.jetbrains.com/clion/

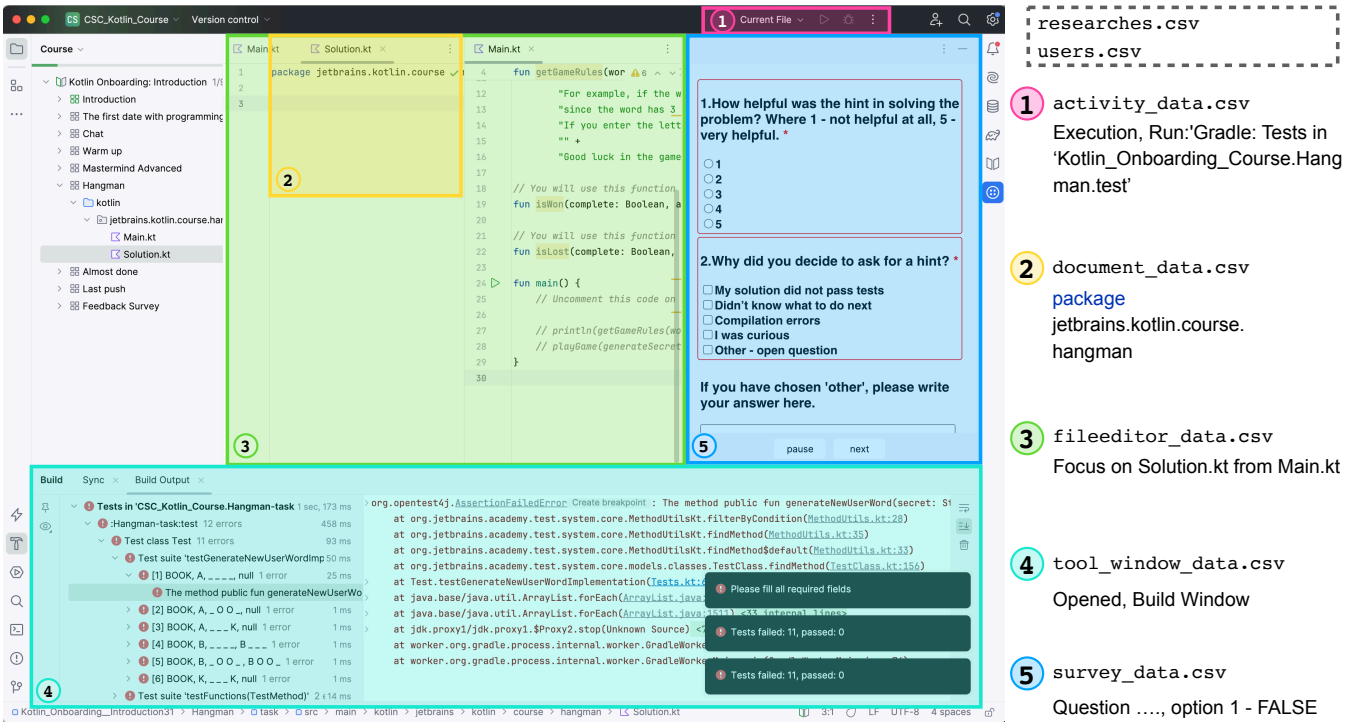

Figure 1: Examples of data collected using KOALA: (1) activities performed in the IDE, such as running, debugging, etc., (2) current student code, (3) opening, closing, and refocusing of files, (4) opened tool windows, and (5) survey data.

2.1 KOALA tool

The TaskTracker tool was previously introduced as a solution for tracking and collecting detailed data while studying inside the IDE [10]. Specifically designed for use with JetBrains IDEs, TaskTracker captures step-by-step code modifications and records a wide range of student actions, including debugging, running programs, and copying or pasting code snippets. By collecting this detailed data, TaskTracker enables researchers to gain deeper insights into how students approach problem-solving and programming tasks. This makes the tool a valuable resource for conducting educational research, analyzing programming behavior, and understanding learning patterns. The insights gathered from TaskTracker data contribute to the design of more effective teaching strategies, curricula, and educational tools, thereby enhancing programming education [13, 1, 9].

This tutorial presents an enhanced version of the TaskTracker tool called KOALA [6], which incorporates significant improvements tailored for research purposes. The updated version allows conducting controlled experiments directly within the IDE. For example, it enables researchers to dynamically modify IDE settings while participants solve specific tasks, allowing for precise experimental conditions. Additionally, the new version introduces a flexible configuration-based approach instead of source-code modification, where a set of configuration files define the tool's behavior and the experimental scenarios.

Finally, the data collected by the new version of the tool is stored in a structured tabular format that ensures clarity and consistency. An example of the collected data can be found in Figure 1. Additionally, the tool supports data conversion into the widely-used *ProgSnap2* format [5], enhancing compatibility with existing research workflows.

2.2 In-IDE learning

The in-IDE learning format was recently introduced as an innovative approach to teaching programming directly within professional IDEs [2]. This format integrates all components of the learning process, including theoretical content, quizzes, and programming exercises, into the IDE. Currently, there are over 50 courses available in this format across various programming languages, including Java, JavaScript, Python, and C++. The format's emphasis on extensive student interaction with the IDE and its diverse features present a valuable opportunity to demonstrate the capabilities of the KOALA tool.

3. TUTORIAL GOALS

The tutorial introduces the KOALA tool that allows tracking the code that the students are writing in the editor and the IDE features they are using, serving both as a data collection tool and a platform for controlled experiments within the IDE. This tool can be useful for researchers who want to study the educational process in granular detail. We will show how to set up and deploy the tool, collect the data in real time and then analyze it to find valuable insights by letting participants solve programming problems in a predefined in-IDE course.

3.1 Prior to the Conference

Ahead of the conference, all necessary materials and resources will be gathered, organized, and made available through a dedicated website. This approach will provide participants with convenient access to the content both during the tutorial and afterward. Furthermore, a pre-survey will be conducted among all registered participants. This survey is designed to collect information about participants' prior knowledge of the topics and their specific expectations for the session. The insights gained from this process will enable the tutorial to be adapted to better suit the unique needs of the audience, ensuring an interactive and engaging experience for all.

3.2 During the Conference

Our tutorial session will be an interactive session split into four parts, outlined below:

1. In the first part, we will provide participants with an overview of the in-IDE learning format and the KOALA tool to establish a better context for the audience. By the end of this section, participants will be familiar with the framework for the upcoming practical part.

2. In the second part of the tutorial, we will set up the KOALA tool together with the audience. We will show all available features of the tool and how to customize it for the research needs. As the output of this part, the participants will have a set of configuration files that can be used for future research or adapted for new studies.

3. In the third part of the tutorial, we will gather learner data while navigating through a pre-made in-IDE course within the pre-configured environment described earlier. As we complete course assignments, we will engage with various LLM-powered features designed to assist learners in overcoming challenges. As a result, participants will obtain data that is ready for analysis.

4. In the final part of the workshop, we will download the KOALA data, process it, and conduct a simple analysis. Participants will emerge from this section with a compiled dataset that includes all step-by-step changes and IDE interactions during their problem-solving sessions, along with the insights gathered. An example of such data can be found in Figure 1. We will also show a way how to convert the gathered data to the popular *ProgSnap2* [5] format to use for the future analysis.

We will have a short break after each part.

4. ORGANIZERS AND PRESENTERS

Daniil Karol is a researcher at the Education Research group at JetBrains Research.[2] His main research interests focus on exploring new approaches to in-IDE learning and methods for data collection during studies. He has over five years of programming experience, one and a half years of research experience, and one year of teaching experience.

Ilya Vlasov[3] is a researcher at the Education Research group at JetBrains Research.[2] His main research interests are in-IDE learning, data visualization, and statistics. He has over five years of programming experience, three years of research experience, and one year of teaching experience.

Katsiaryna Dzialets[4] is a researcher at the Education Research group at JetBrains Research.[2] Her main research interests are generative AI in Education and in-IDE learning. She has four years of product management and product research experience and more than five years of experience in building EdTech products and projects.

Anna Potriasaeva[5] is a software developer at the Education Research group at JetBrains Research.[2] Her main research interests include in-IDE learning, generative AI in education, and intelligent tutoring systems. She has over five years of programming experience and one year of research experience.

Anastasiia Birillo[6] is the head of the Education Research group at JetBrains Research[2] and an external PhD student in Computer Science at Utrecht University. Her main research interests are in-IDE learning, intelligent tutoring systems, and generative AI. She has more than eight years of programming experience, more than five years of teaching experience, and about five years of research experience.

5. REFERENCES

[1] A. Birillo, E. Artser, A. Potriasaeva, I. Vlasov, K. Dzialets, Y. Golubev, I. Gerasimov, H. Keuning, and T. Bryksin. One step at a time: Combining llms and static analysis to generate next-step hints for programming tasks. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, pages 1–12, 2024.

[2] A. Birillo, M. Tigina, Z. Kurbatova, A. Potriasaeva, I. Vlasov, V. Ovchinnikov, and I. Gerasimov. Bridging education and development: Ides as interactive learning platforms. In *Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments*, pages 53–58, 2024.

[3] Y. Cao, L. Wang, Z. Zheng, and X. Tao. A tool for non-intrusive and privacy-preserving developers' programming activity data collection. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 953–962. IEEE, 2020.

[4] J. Edwards, K. Hart, R. Shrestha, et al. Review of csedm data and introduction of two public cs1 keystroke datasets. *Journal of Educational Data Mining*, 15(1):1–31, 2023.

[2]Education Research at JetBrains Research: `https://lp.jetbrains.com/research/education/`

[3]Ilya Vlasov's Google Scholar: `https://scholar.google.com/citations?user=FEHw5dAAAAAJ&hl=en`

[4]Katsiaryna Dzialets's Google Scholar: `https://scholar.google.com/citations?user=Cgk-_5MAAAAJ&hl=en`

[5]Anna Potriasaeva's Google Scholar: `https://scholar.google.com/citations?hl=en&user=Dn2kmtAAAAAJ`

[6]Anastasiia Birillo's Google Scholar: `https://scholar.google.com/citations?user=V6PGOnwAAAAJ&hl=en`

[5] D. Hovemeyer and K. Rivers. Progsnap 2: Towards a standard representation for programming process data.

[6] D. Karol, E. Artser, I. Vlasov, Y. Golubev, H. Keuning, and A. Birillo. Koala: Customizable ide data collection tool. 2025.

[7] A. M. Kazerouni, S. H. Edwards, T. S. Hall, and C. A. Shaffer. Deveventtracker: Tracking development events to assess incremental development and procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 104–109, 2017.

[8] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg. The bluej system and its pedagogy. *Computer Science Education*, 13(4):249–268, 2003.

[9] D. Lohr, H. Keuning, and N. Kiesler. You're (not) my type-can llms generate feedback of specific types for introductory programming tasks? *Journal of Computer Assisted Learning*, 41(1):e13107, 2025.

[10] E. Lyulina, A. Birillo, V. Kovalenko, and T. Bryksin. Tasktracker-tool: A toolkit for tracking of code snapshots and activity data during solution of programming tasks. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 495–501, 2021.

[11] R. Minelli, A. Mocci, R. Robbes, and M. Lanza. Taming the ide with fine-grained interaction data. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pages 1–10. IEEE, 2016.

[12] M. Pärtel, M. Luukkainen, A. Vihavainen, and T. Vikberg. Test my code. *International Journal of Technology Enhanced Learning 2*, 5(3-4):271–283, 2013.

[13] L. Roest, H. Keuning, and J. Jeuring. Next-step hint generation for introductory programming using large language models. In *Proceedings of the 26th Australasian Computing Education Conference*, pages 144–153, 2024.

[14] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. In *Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007)*, pages 9–9. IEEE, 2007.