Evaluating and Improving the Quality of LLM-Generated Code

Glaucia Melo
Toronto Metropolitan University
Toronto, Ontario, Canada
glaucia@torontomu.ca

Jessica Pourleyli
Toronto Metropolitan University
Toronto, Ontario, Canada
jessica.pourleyli@torontomu.ca

Genevieve Caumartin
Concordia University
Montreal, Quebec, Canada
genevieve.caumartin@mail.concordia.ca

Corey Yang-Smith
University of Calgary
Calgary, Alberta, Canada
corey.yangsmith@ucalgary.ca

Diego Costa
Concordia University
Montreal, Quebec, Canada
diego.costa@concordia.ca

Ahmad Abdellatif
University of Calgary
Calgary, Alberta, Canada
ahmad.abdellatif@ucalgary.ca

Abstract

Large Language Models (LLMs) are increasingly used to generate production code, yet systematic methods for evaluating their quality and security remain underdeveloped. This tutorial introduces a reusable, end-to-end evaluation pipeline grounded in empirical software engineering practices, focusing on post-generation validation rather than prompt design. Participants will apply static analysis tools to assess maintainability, reliability, and security, and compare results across models, prompts, and human-written baselines. The pipeline supports structured aggregation and interpretation of outputs, enabling reproducible and defensible assessments. Extensions include agentic remediation, explainability for trust calibration, and bias-aware evaluation. Attendees will leave with practical evaluation artifacts and a principled framework for validating AI-generated code in modern development workflows.

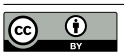
CCS Concepts

• **Security and privacy** → **Software security engineering**; • **Software and its engineering** → *Software verification and validation*; *Software defect analysis*; • **Computing methodologies** → Artificial intelligence.

Keywords

Large Language Models, LLM-generated code, code quality, software security, static analysis, AI-assisted software engineering, human-in-the-loop, agentic systems, explainable AI

ACM Reference Format:
Glaucia Melo, Jessica Pourleyli, Genevieve Caumartin, Corey Yang-Smith, Diego Costa, and Ahmad Abdellatif. 2026. Evaluating and Improving the Quality of LLM-Generated Code. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26), July 05–09, 2026, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3803437.3804900



FSE Companion '26, Montreal, QC, Canada

ACM ISBN 979-8-4007-2636-1/2026/07
https://doi.org/10.1145/3803437.3804900

1 Introduction

Large Language Models (LLMs) are increasingly embedded in software development workflows, yet the quality and security of AI-generated code remain difficult to assess. Empirical studies show that such code often contains subtle defects, maintainability issues, and security vulnerabilities that are not immediately apparent, raising concerns about trust and responsible integration into production systems.

This tutorial addresses this gap by introducing a reusable, end-to-end evaluation pipeline for LLM-generated code, grounded in empirical software engineering practices. The pipeline integrates static analysis, quantitative metrics, visualization, and iterative verification to support systematic comparison across models, prompts, and human-written baselines. Unlike prior work focused on code generation or prompting strategies, this tutorial treats post-generation evaluation as a first-class software engineering problem.

Participants will learn how to critically interpret analysis outputs, going beyond raw metrics to perform comparative and contextual evaluation. The tutorial also introduces extensions to agentic remediation, explainability for trust calibration, and bias-aware evaluation, all within a unified and reproducible workflow.

The tutorial targets researchers and practitioners working with AI-assisted development, requiring only basic software engineering knowledge. By the end, participants will be equipped with practical artifacts and a principled framework for evaluating and improving AI-generated code in real-world settings.

Tutorial Format and Preferred Duration

This is a 180-minute tutorial, structured as two 90-minute blocks separated by a short break. The format is designed to be interactive and engaging, combining concise lectures to introduce core concepts and research findings, live demonstrations that provide step-by-step walkthroughs of analysis techniques, and interactive Q&A.

2 Outline and Covered Topics

2.1 Part 1: Foundations of LLM-Generated Code Quality (90 minutes)

(15 min) Context and Setup: LLMs in Modern SE. A brief introduction to the role of LLMs in software development, common quality and security risks, and the motivation for systematic evaluation. Participants will open the Google Colab notebook, explore the dataset, and familiarize themselves with the analysis pipeline.

(30 min) Evaluation of Maintainability and Reliability. Presenters will demonstrate SonarQube-based static analysis to assess maintainability and reliability of LLM-generated and human-written code. Participants will run the analysis, extract key metrics (e.g., code smells and bugs), and compare results across sources. The discussion will emphasize critical interpretation of metrics, supported by recent empirical studies [2–4], and include basic visualizations within the notebook. SonarQube and Bandit are used due to their industry adoption and support for reproducible, large-scale analysis.

(30 min) Detection of Security Vulnerabilities. Participants will use Bandit to identify common security vulnerabilities in the same code artifacts. They will classify findings, distinguish likely true and false positives, and compare vulnerability profiles across code sources. Results will be aggregated and visualized to support comparative analysis.

(15 min) Integrated Analysis and Discussion. The session concludes with a walkthrough of combining quality and security metrics into a unified pipeline. Participants will reflect on trade-offs, tool limitations, and implications for AI-assisted development, followed by an interactive Q&A.

2.2 Part 2: Post-Generation Remediation, Explainability, and Human-in-the-Loop Validation (90 minutes)

Building on Part 1, this section extends the same evaluation pipeline to advanced tasks, including automated remediation, explainability, and fairness-aware evaluation. Using the Google Colab notebook, participants will experiment with agentic workflows, analyze model explanations, and explore ethical considerations in AI-assisted SE.

(30 min) Agentic Systems for Automated Remediation. Participants will explore a simplified agentic workflow for bug localization and vulnerability remediation [1]. They will run an iterative loop combining generation, verification, and feedback, and examine the limitations and failure modes of automated remediation.

(30 min) Explainability for Developer Trust. Participants will compare LLM-generated explanations for code changes and remediation. They will evaluate explanation quality (e.g., clarity, alignment with tool outputs, and usefulness for decision-making) and explore how explainability techniques support trust in AI-assisted development.

(15 min) Bias-Aware Evaluation. Participants will analyze a curated dataset of code-generation scenarios to examine potential bias in model outputs. They will perform simple group-based comparisons and qualitative analysis, applying lightweight fairness assessment methods such as distribution comparisons and subgroup performance gaps.

(5 min) Synthesis: Human-in-the-Loop and the Future of AI-Assisted Development. The tutorial concludes with a reflection on the full pipeline, followed by an open discussion and Q&A.

3 Key Learning Outcomes

Upon completing this tutorial, participants will be able to:

- Apply a reusable, end-to-end evaluation pipeline for assessing the quality and security of LLM-generated code, and adapt it to new datasets, models, and software artifacts.
- Interpret static analysis outputs comparatively and critically evaluate quantitative metrics, distinguishing meaningful signals from tool-specific noise and false positives.
- Apply and evaluate a human-in-the-loop, agentic remediation workflow in which LLM-generated fixes are iteratively verified through automated re-analysis.
- Perform basic fairness- and bias-aware analysis of contextual AI outputs in SE tasks, and reason about ethical implications in technical evaluation pipelines.

4 Speakers Information and Biographies

Dr. Glaucia Melo is an Assistant Professor at Toronto Metropolitan University, working at the intersection of software engineering and AI, with a focus on LLM-generated code, software security, explainable AI, and fairness-aware systems. She has published in leading SE venues including ICSE, ESEM, and CASCON.

Jessica Pourleyli is an undergraduate researcher at Toronto Metropolitan University focusing on security risks in LLM-generated code, including vulnerability detection and automated repair.

Corey Yang-Smith is a doctoral researcher at the University of Calgary and a full-time Machine Learning Scientist at SOVRA. His research explores the use of large language models in automated software engineering to improve software quality and security.

Genevieve Caumartin is a doctoral researcher at Concordia University specializing in agentic systems and automated bug localization and repair, with publications in venues associated with ICSE, ASE, and SANER.

5 Tutorial History

This is a new tutorial. The content is based on a mature, cohesive body of research conducted by the presenters and their collaborators, published across multiple peer-reviewed venues. The material has been structured for this tutorial to provide a hands-on pedagogical experience.

6 Technical Requirements

The tutorial requires internet access, a projector, and participant laptops. Attendees will follow hands-on exercises using a provided codebase to run static analysis tools and validate remediation results.

References

[1] G. Caumartin and G. Melo. 2026. Reformulate, Retrieve, Localize: Agents for Repository-Level Bug Localization. In *2026 IEEE/ACM International Workshop on Bots and Agents in Software Engineering (BoatSE)*.

[2] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems* 36 (2023), 21558–21572.

[3] Alfred Santa Molison, Marcia Moraes, Glaucia Melo, Fabio Santos, and Fabio Assuncao. 2025. Is LLM-Generated Code More Maintainable & Reliable Than Human-Written Code? . In *2025 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 151–162. doi:10.1109/ESEM64174.2025.00036

[4] Mohammed Latif Siddiq, Lindsay Roney, Jiahao Zhang, and Joanna Cecilia Da Silva Santos. 2024. Quality assessment of chatgpt generated code and their use by developers. In *Proceedings of the 21st international conference on mining software repositories*. 152–156.