

Sustaining Software Quality in the Era of AI-Assisted Development

Glaucia Melo*, Jessica Pourleyli*, Genevieve Caumartin[†], Corey Yang-Smith[‡], Diego Costa[†], Ahmad Abdellatif[‡]

* Toronto Metropolitan University, Toronto, ON, Canada

{glaucia, jessica.pourleyli}@torontomu.ca

[†] Concordia University, Montreal, QC, Canada

{genevieve.caumartin,diego.costa}@concordia.ca

[‡] University of Calgary, Calgary, AB, Canada

{corey.yangsmith,ahmad.abdellatif@ucalgary.ca}

Abstract—Large language models (LLMs) now write a substantial share of new code, but the open questions for software engineering (SE) research have moved past generation: how do agentic workflows localize and repair defects, how do explanations help developers decide whether to trust an AI-produced change, and how do we evaluate all of this fairly? This 90-minute, lecture-based tutorial gives researchers and students a guided, vendor-neutral introduction to using agents and explainable AI (XAI) in software engineering, with an expanded focus on bias and fairness. Grounded in the presenters’ peer-reviewed work, it follows four questions: is LLM-generated code maintainable, reliable, and secure; how can agents effectively localize and fix what we find; will a developer actually trust the fix; and is our generated code itself unbiased? No hands-on work or laptops are required. Attendees leave with a clear map of the area, the key methods and empirical findings, and pointers to openly available, reusable artifacts they can build on in their own research.

Index Terms—AI-assisted software engineering, agentic systems, explainable AI, bias, LLM-generated code, responsible AI,

basic software engineering knowledge. By the end, participants will have a map of how agents and explainable AI are used in software engineering, a working understanding of bias-aware evaluation, and pointers to reusable artifacts and open research directions they can build on.

2. TUTORIAL FORMAT

Duration: 1.5 hours (90 minutes). **Format:** lecture-based, with lectures and narrated examples drawn from the presenters’ studies. We will also include live polls and guided discussions. No hands-on component, laptops, or accounts are required. The team co-presents by segment: Glaucia Melo leads the context, foundations, and synthesis; Jessica Pourleyli leads agents in SE, explainability, bias and fairness.

A. Target Audience and Prerequisites

The tutorial targets SE researchers, graduate students, and early-career researchers entering AI-assisted software development, as well as practitioners who want a structured overview. It assumes basic familiarity with software development (code quality, testing, the development lifecycle). No machine learning or LLM expertise is required; all AI concepts are introduced from an SE perspective and motivated through empirical findings.

B. Logistics and Materials

The tutorial needs only a projector and screen with standard lecture seating; attendees need no software or accounts. We expect roughly 30 participants, and the format scales to the room. Slides and pointers to the team’s openly shared Colab notebooks and datasets are provided as optional take-home resources so attendees can go deeper on their own code after the session.

3. LEARNING OBJECTIVES AND TUTORIAL OUTLINE

The outline is presented in Table I. After this tutorial, participants will be able to:

- 1) Describe how LLMs are used across the SE lifecycle and the main quality, security, and fairness risks of LLM-generated code;

1. MOTIVATION AND RELEVANCE TO CASCON

Large language models are now embedded across the software development lifecycle, and the question of whether they *can* produce code is largely settled. The questions that matter for research have moved downstream: once an assistant proposes a change, how do agentic workflows find and repair defects, how do explanations help a developer judge the change, and how do we evaluate any of this without baking in bias? A change can compile and read well while still carrying maintainability debt, exploitable vulnerabilities, or hidden disparities [1]–[3].

These threads, agentic SE, explainable AI, and fairness, are advancing quickly but in separate subcommunities, which makes the area hard to enter. This tutorial offers a coherent, citable map. It follows four questions: is LLM-generated code maintainable, reliable, and secure; can agents localize and fix what we find; will a developer trust the fix; and is our evaluation itself fair? The first is treated as a foundation; the middle two introduce agents and XAI for SE; and the fourth, fairness and bias, receives the deepest treatment.

This tutorial targets researchers, graduate students, and practitioners working in AI-assisted development and requires only

- 2) Explain how agentic workflows are applied to SE tasks such as bug localization and repair, and how to read an agent’s trajectory to spot errors, retries, and data-contamination risks;
- 3) Compare explainability strategies (e.g., explain-while-patch vs. patch-then-explain) and articulate why explanation quality does not by itself produce developer trust;
- 4) Apply bias-aware and intersectional evaluation concepts to AI-assisted SE, and identify where bias can enter agentic pipelines;
- 5) Identify reusable, openly available artifacts and open research directions in agents, XAI, and fairness for SE.

TABLE I: 90-Minute Outline

Min	Topic
10	Context and motivation. LLMs across the SE lifecycle; from “Can LLMs write code?” to trusting, explaining, and fairly evaluating what they write; the four questions that organize the session. <i>Evidence:</i> [1]–[3].
10	Foundations: evaluating LLM-generated code (Q1). What static analysis (e.g., SonarQube, Bandit) reveals about maintainability, reliability, and security; secure-looking but incorrect code; reading metrics against human baselines rather than at face value. <i>Evidence:</i> [4].
20	Agents in software engineering (Q2). Agentic bug localization and repair; query reformulation and retrieval; the agentic loop (reformulate, retrieve, refine, self-evaluate); reading agent trajectories to surface errors, retries, and data-contamination risks; context management. <i>Evidence:</i> [5].
15	Explainable AI for developer trust (Q3). Explanation strategies—single-shot, explain-while-patch (ante-hoc), and patch-then-explain (post-hoc); evaluating explanation quality (correctness, completeness, context); why strong explanations do not by themselves yield merge confidence. <i>Evidence:</i> [6].
25	Bias and fairness (Q4) How bias enters quality scoring, agentic pipelines, and explanations; bias-aware and intersectional evaluation; subgroup disparity analysis; and emerging <i>agentic-bias</i> directions (presented as an open, in-development research frontier). <i>Evidence:</i> [7], [8].
10	Synthesis and discussion. Human-in-the-loop; fairness by design; reproducibility; open research directions; Q&A.

4. SIGNIFICANCE AND TAKEAWAYS

The area is moving quickly, and this tutorial gives researchers and students a single, grounded pass through it: the key methods, the empirical findings behind them, and pointers to reusable, openly available artifacts. The fairness segment is the distinctive contribution; most introductions to AI-assisted SE treat bias as an afterthought, whereas here it receives the most time and is connected to the agentic and explainability material the audience has just seen. Attendees leave able to read agent trajectories critically, reason about when an explanation should (and should not) earn trust, and bring an intersectional, bias-aware lens to their own evaluations.

5. OVERLAP STATEMENT

A related, more extensive tutorial by an overlapping set of presenters, *Evaluating and Improving the Quality of LLM-Generated Code*, was accepted as a 180-minute, hands-on

tutorial at the Foundations of Software Engineering (FSE) conference. Both draw on a similar body of peer-reviewed work, but this CASCON tutorial differs deliberately in scope, format, and emphasis. The FSE tutorial is hands-on: participants run a full evaluation pipeline (SonarQube, Bandit, agentic remediation) themselves in Colab over 180 minutes. This CASCON tutorial is a 90-minute, lecture-based introduction for researchers and students entering the area; it requires no laptops or hands-on work, surveys the same methods at a conceptual level, and places its greatest emphasis on fairness and bias, including emerging agentic-bias considerations. The proposal and its materials are original to this submission, and neither tutorial presumes attendance at the other.

6. TUTORIAL HISTORY

This is a new tutorial. Its content is based on a cohesive body of peer-reviewed research conducted by the presenters and their collaborators across three institutions, condensed and reframed for a lecture-based, research-onboarding format.

7. SPEAKERS INFORMATION

Glauca Melo (lead presenter; corresponding tutor) is an Assistant Professor at Toronto Metropolitan University working at the intersection of SE and AI, empirical evaluation of LLM-generated code, software security, agentic systems, explainable AI, and fairness-aware AI, with publications at ICSE, ESEM, and CASCON. **Jessica Pourleyli** (presenter) is an incoming master’s student and researcher at Toronto Metropolitan University studying security in LLM-generated code, vulnerability detection, and automated repair. **Genevieve Caumartin** (preparation contributor) is a PhD student at Concordia University working on context engineering and agentic systems for SE, including bug localization and repair, with work at venues co-located with ICSE, ASE, and SANER. **Corey Yang-Smith** (preparation contributor) is a PhD student at the University of Calgary and a Machine Learning Scientist at SOVRA, studying LLMs for automated software engineering. **Diego Costa** (preparation contributor) is a faculty member at Concordia University in empirical software engineering and software performance/quality. **Ahmad Abdellatif** (preparation contributor) is a faculty member at the University of Calgary working on software analytics and AI for software engineering.

ACKNOWLEDGEMENTS

We used Grammarly AI for spelling correction, as well as ChatGPT to generate some of the LaTeX formatting.

REFERENCES

- [1] A. S. Molison, M. Moraes, G. Melo, F. Santos, and F. Assuncao, “Is LLM-generated code more maintainable & reliable than human-written code?” in *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2025, pp. 151–162.
- [2] M. L. Siddiq, L. Roney, J. Zhang, and J. C. D. S. Santos, “Quality assessment of ChatGPT generated code and their use by developers,” in *Proceedings of the 21st International Conference on Mining Software Repositories (MSR)*, 2024, pp. 152–156.

- [3] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 21 558–21 572.
- [4] J. Pourleyli, G. Melo *et al.*, "When ai codes: Understanding the security of llm-generated code," 2026, under review.
- [5] G. Caumartin and G. Melo, "Reformulate, retrieve, localize: Agents for repository-level bug localization," in *Proceedings of the 2026 7th International Workshop on Bots and Agents in Software Engineering*, ser. BoatSE '26. New York, NY, USA: Association for Computing Machinery, 2026, p. 39–45. [Online]. Available: <https://doi.org/10.1145/3786161.3788460>
- [6] C. Yang-Smith, G. Melo *et al.*, "A user study on explainable artificial intelligence in software engineering," 2026, in preparation.
- [7] N. Abule and G. Melo, "Improving fairness by debiasing intersectional biases in contextual AI models," in *Proceedings of CASCON*, 2025.
- [8] J. Pourleyli, G. Melo *et al.*, "Neutralizing intersectional bias in software engineering with ai," 2026, in preparation.