Developer Roles and Ethical Responsibilities in LLM-Assisted Software Development

Kimberly Davis, Qusay H. Mahmoud
Department of Electrical, Computer, and Software Engineering, Ontario Tech University, Oshawa, ON, Canada

Interested in this research?

Objective

- To analyze how advancements in Large Language Models (LLMs) are transforming the roles and responsibilities of software developers.
- To explore the shift in developer focus from code writing to curation, review, and ethical oversight.
- To assess the ethical, professional, and practical implications of integrating LLMs into software development practices.

Methodology

This research examines how LLM integration is reshaping software development through an ongoing review of recent academic and industry literature (2022–present). The findings highlight emerging ethical, professional, and practical implications for software developers and development practices.

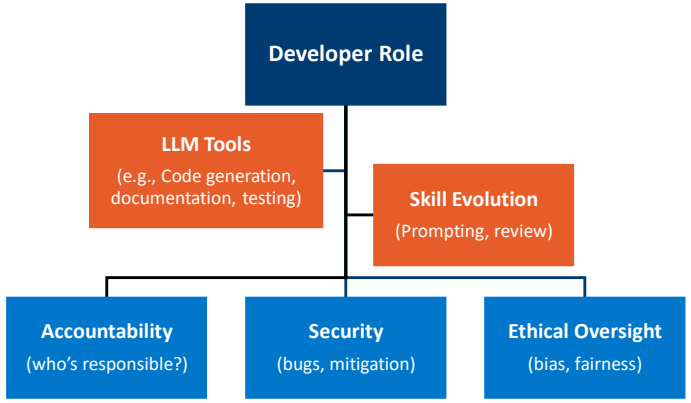
Key Research Findings

This research reveals a transformation in the developer role and software practices driven by LLMs:

- **Bias & Fairness:** LLMs can reproduce social and vendor biases in code. Developers must detect, document, and mitigate bias to ensure fairness in generated code.
- **Productivity vs. Risk:** Efficiency gains can hide potential ethical risks (e.g., insecure, biased code). Balancing productivity with moral responsibility is essential for professional integrity.
- **Skill Development & AI Literacy:** LLMs can make coding more accessible, but over-reliance may weaken critical thinking and technical understanding. Developers need skills in AI evaluation, verification, and responsible use.
- **Oversight & Governance:** Continuous human oversight is essential for accountability. This requires new ethical guidelines and legal frameworks to manage AI risks.
- **Evolving Standards:** The industry needs to establish new standards for accountability, bias testing, and code provenance as part of professional best practices.

Evolution of Developer Roles

- Developers are shifting from code creators to reviewers and validators
- Productivity improves, but ethical and quality oversight intensifies



Core dimensions of evolving developer responsibilities in LLM-assisted coding.

Skill Evolution

- Developers gain new skills in prompt engineering, AI evaluation, and bias testing
- Developers require ongoing AI literacy and ethical awareness in software practice

Code Ownership and Accountability

- Authorship is blurred between model, developer, and dataset
- Generated code may include copyrighted or unlicensed segments
- Transparency, provenance logging, and usage disclosure are needed

Security and Reliability

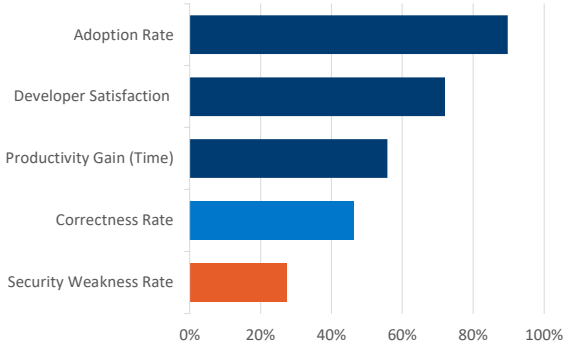
- LLM-generated code may contain security flaws
- Human review is essential, especially for novice developers
- Developers play an important role in mitigating hidden risks in generated code

Ethical Oversight

- LLMs can reproduce social biases (e.g., gender, socioeconomic, or regional)
- Domain-specific testing and bias mitigation are needed before deployment
- Developers are responsible for bias checks and inclusive design

Impact of LLMs

Selected Findings from Prior Studies



Metric	Results	Study Type	Source
Adoption Rate	89.7%	Survey	Stalnaker et al. [1]
Developer Satisfaction	72%	Survey	Bakal et al. [2]
Productivity Gain	55.8%	Controlled Experiment	Peng et al. [3]
Correctness Rate	46.3%	Benchmark Evaluation	Kharma et al. [4]
Security Weakness Rate	27.3%	Static Security Analysis	Fu et al. [5]

Conclusion

This research highlights key directions to strengthen ethical and responsible LLM-assisted software development:

- Strengthen understanding of changing developer roles and responsibilities.
- Integrate bias checks, security review, and human oversight into standard development workflows.
- Support AI literacy and responsible decision-making as LLMs become more integrated into software practice.

References

[1] T. Stalnaker et al., "Developer Perspectives on Licensing and Copyright Issues Arising from Generative AI for Software Development," ACM Transactions on Software Engineering and Methodology, vol. 35, no. 2, 2026, doi: 10.1145/3743133.
[2] G. Bakal, A. Dasdan, Y. Katz, M. Kaufman, and G. Levin, "Experience with GitHub Copilot for Developer Productivity at ZoomInfo," arXiv:2501.13282, 2025.
[3] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," arXiv:2302.06590, 2023.
[4] M. Kharma, S. Choi, M. AlKhanafseh, and D. Mohaisen, "Security and Quality in LLM-Generated Code: A Multi-Language, Multi-Model Analysis," IEEE Transactions on Dependable and Secure Computing, 2026.
[5] Y. Fu et al., "Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study," ACM Transactions on Software Engineering and Methodology, vol. 34, no. 8, Art. 218, 2025, doi: 10.1145/3716848.

Contact: Kimberly.davis@ontariotechu.net