Generative AI and Its Impact on Software Quality: Opportunities and Challenges

Anjan Kumar Ojha
Apple Inc.

Generative AI and Its Impact on Software Quality: Opportunities and Challenges

Author Name(s) Anjan Kumar Ojha
Email(s) anzaan.kumar@gmail.com

Abstract

Generative Artificial Intelligence (GenAI) is rapidly transforming software engineering practices, from code generation to automated testing. This white paper examines the implications of GenAI on software quality, exploring its opportunities and risks. The analysis spans quality attributes such as reliability, maintainability, security, and performance, highlighting both the potential benefits and the inherent challenges. Future research and governance strategies are discussed to ensure sustainable adoption of GenAI in software engineering.

Keywords

Generative AI, Software Quality, Code Generation, Software Testing, Reliability, Maintainability

1. Introduction

The advent of Generative Artificial Intelligence (GenAI) has opened new frontiers in software engineering. Large language models (LLMs), such as GPT-4 and Code Llama, have demonstrated remarkable capabilities in generating human-like text and source code. Their integration into development workflows has introduced opportunities for accelerating software production while raising new concerns regarding software quality. IEEE defines software quality as the degree to which software satisfies stated and implied needs under specified conditions. This paper explores how GenAI affects the key dimensions of software quality, balancing opportunities with challenges.

2. Literature Review

Research on GenAI's impact on software engineering is growing rapidly. Early studies show improvements in developer productivity through AI-assisted coding tools (Vaithilingam et al., 2022). Other works have raised concerns about correctness and maintainability of AI-generated code (Pearce et al., 2023). Scholars also emphasize the role of AI in testing automation (Zhang et al., 2023) and defect prediction. However, systematic assessments of software quality implications remain limited.

3. Opportunities for Software Quality

GenAI contributes positively to software quality in several domains:

• Code Generation: Accelerates development cycles by automating boilerplate code creation.
• Automated Testing: Generates unit tests and regression tests, improving test coverage.
• Knowledge Capture: Documents code, enhancing maintainability.
• Defect Prediction: Learns from historical defect data to predict vulnerabilities.

4. Challenges and Risks

The adoption of GenAI also introduces critical risks:

• Reliability: AI-generated code may contain subtle bugs.
• Security: AI may reproduce vulnerable patterns from training data.
• Maintainability: Poorly structured code can reduce long-term maintainability.
• Explainability: Lack of transparency complicates debugging and verification.
• Ethical Risks: Intellectual property and plagiarism concerns persist.

5. Case Studies and Industrial Adoption

Several companies, including Microsoft (GitHub Copilot) and Amazon (CodeWhisperer), have deployed GenAI-powered tools in development workflows. Initial reports suggest productivity gains of up to 30% but mixed results on long-term maintainability. Case studies highlight the importance of human-in-the-loop approaches to ensure quality assurance.

6. Future Research and Directions

Future research should focus on formal verification of AI-generated code, hybrid AI-human coding models, and standards for responsible AI use in software engineering. Governance frameworks and benchmarking systems are essential to guide sustainable adoption.

7. Conclusion

Generative AI presents both opportunities and risks for software quality. While it can accelerate development and testing, it introduces challenges in reliability, security, and maintainability. The path forward requires balanced integration of AI with traditional engineering practices, guided by ethical and technical safeguards.

References

[1] Vaithilingam, P., et al. (2022). 'Expectations vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models.' ACM CHI.
[2] Pearce, H., et al. (2023). 'Examining the Quality of AI-Generated Code: An Empirical Study.' IEEE Transactions on Software Engineering.
[3] Zhang, Y., et al. (2023). 'AI for Software Testing: Opportunities and Challenges.' IEEE Software.

[4] IEEE Standard Glossary of Software Engineering Terminology (1990). IEEE Std 610.12-1990.