JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205

Volume 15 Issue 10, October 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

WHY CLEAN CODE STILL MATTERS IN THE AGE OF AI

J.J. Munirov
"ASIA INTERNATIONAL UNIVERSITY"

Teacher of "General technical sciences" department

Annotation: This article examines the enduring importance of clean code practices in an era increasingly influenced by artificial intelligence (AI) and automated coding tools. It explores the historical roots of clean coding, its principles, and how AI-assisted development challenges and complements these values. The discussion highlights why human readability, maintainability, and ethical programming remain crucial even when AI can generate functional code. The paper also identifies potential risks of neglecting code quality in automated environments and proposes strategies for preserving clean code culture in the future of software engineering.

Keywords: Clean Code, Artificial Intelligence, Software Engineering, Code Quality, Maintainability, Readability, Automation, Ethics

Introduction

The field of software development has entered a transformative phase with the emergence of artificial intelligence tools capable of generating, refactoring, and even documenting code. Systems like GitHub Copilot, ChatGPT, and Tabnine have revolutionized how programmers approach coding tasks by accelerating productivity and reducing the need for repetitive manual work. However, as these tools become more prevalent, an essential question arises: does clean code still matter when machines can write it for us?

Clean code—a concept popularized by Robert C. Martin ("Uncle Bob")—emphasizes clarity, simplicity, and maintainability in software design. It is not just about making code work but making it understandable and sustainable for humans. In the age of AI, while machines can generate syntactically correct code, the responsibility for structure, design decisions, and ethical implications still rests with human developers. Therefore, understanding the relevance of clean code principles has never been more critical.

Defining Clean Code. Clean code is a philosophy that values readability, simplicity, and purpose-driven design. At its core, clean code ensures that programs are easy to understand, modify, and extend. The principles often include meaningful naming conventions, consistent formatting, modularity, and avoidance of duplication. Robert C. Martin's book Clean Code: A Handbook of Agile Software Craftsmanship outlines that good code should "read like well-written prose." This means any developer—regardless of who originally wrote it—should be able to comprehend the logic quickly. Clean code promotes collaboration, reduces technical debt, and prevents bugs caused by unclear or redundant logic.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 10, October 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

Even before AI, clean code was the foundation of professional software engineering. It acted as both a communication tool among developers and a safeguard for the longevity of projects. In the modern era, these values are being tested by automation and machine-generated code. Artificial intelligence has introduced tools that can analyze existing repositories, predict coding patterns, and automatically suggest or generate code snippets. For example, AI models trained on large datasets can autocomplete functions, detect bugs, or optimize performance. While these capabilities enhance productivity, they also risk creating code that is technically correct but semantically opaque. AI does not inherently understand the intent behind a software design. It can produce thousands of lines of working code, yet that code might lack the clarity or logical flow that humans rely on for comprehension and maintenance. Developers using AI-generated code often find themselves debugging or refactoring machine-written logic to align it with project standards. Thus, clean code principles remain essential as a framework for reviewing, integrating, and maintaining AI-assisted outputs. Clean code continues to matter because software development remains a fundamentally human-centered activity. Teams collaborate, review, and evolve codebases over years—sometimes decades. Even if AI contributes to the initial creation, humans will still maintain and expand the system. AI-generated code often prioritizes correctness over clarity. Without human intervention, variable names, structure, or comments may not align with project conventions, making collaboration difficult. Future developers need to understand code logic to fix bugs or implement new features. Clean code ensures that such tasks are efficient and error-free. AI models can inadvertently generate insecure or biased code patterns. Clean code practices—such as proper documentation, validation, and transparency—help mitigate these risks. Codebases often outlive the developers who wrote them. Clean, well-documented code ensures continuity, even when team members change or AI tools evolve.

Overreliance on Automation: Developers may depend too heavily on AI, neglecting to review or understand generated code.

Loss of Craftsmanship: The art of writing clean, elegant code may fade as developers shift from creators to curators.

Inconsistency in Code Quality: AI models trained on diverse data may produce inconsistent styles, violating established conventions.

Security and Ethical Risks: Machine-generated code might unknowingly introduce vulnerabilities or use copyrighted patterns without awareness. To address these challenges, software teams must establish clear coding standards, use code review processes, and train AI systems on curated datasets that reflect best practices. As AI evolves, the relationship between automation and craftsmanship will continue to shape the software industry. Future AI coding assistants are likely to become more context-aware, capable of understanding project-specific patterns and enforcing coding standards automatically. This could enhance, rather than replace, clean code practices. Moreover, the collaboration between human intuition and AI speed could lead to “augmented clean code”—a model where developers use AI as a partner to enforce readability, test coverage, and architectural coherence. Universities and coding bootcamps are

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 10, October 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

also expected to integrate AI literacy with traditional clean coding education, ensuring that future developers appreciate both efficiency and elegance in software design.

Conclusion

Clean code remains the cornerstone of software quality, even in the age of artificial intelligence. While AI accelerates development and enhances efficiency, it cannot replace the human understanding of clarity, structure, and intent. The future of programming depends not on abandoning clean code principles but on integrating them with intelligent automation.

As AI continues to reshape how code is written, developers must uphold their role as craftsmen—ensuring that every line of code, whether human- or machine-generated, communicates purpose, responsibility, and clarity. Clean code, therefore, is not an outdated ideal but a timeless standard guiding the ethical and sustainable evolution of software engineering.

Resources

1. Мунировв, Д. Д. О. (2024). КАК ОБЛАЧНЫЕ ТЕХНОЛОГИИ СПОСОБСТВУЮТ ЦИФРОВОЙ ТРАНСФОРМАЦИИ. *MASTERS*, *2*(8), 44-51.

2. Мунировв, Д. Д. О. (2024). РОЛЬ СЕТЕЙ В СОВРЕМЕННОЙ ИТ-ИНФРАСТРУКТУРЕ. *WORLD OF SCIENCE*, *7*(8), 27-34.

3. Мунировв, Д. Д. О. (2024). ВАЖНОСТЬ КИБЕРБЕЗОПАСНОСТИ В ЦИФРОВУЮ ЭПОХУ. *PSIXOLOGIYA VA SOTSIOLOGIYA ILMIY JURNALI*, *2*(7), 35-42.

4. MUNIROV, J. (2024). THE FUTURE OF CLOUD TECHNOLOGY: DRIVING INNOVATION AND EFFICIENCY IN THE DIGITAL ERA. *Medicine, pedagogy and technology: theory and practice*, *2*(9), 193-201.

5. MUNIROV, J. (2025). REVOLUTIONIZING REMOTE WORK WITH REAL-TIME COLLABORATION TOOLS. *PEDAGOGIK TADQIQOTLAR JURNALI*, *2*(2), 27-31.

6. MUNIROV, J. (2025). VIRTUAL REALLIK TEXNOLOGIYALARIDAN FOYDALANIB AMALIY O ‘QUV JARAYONLARINI TASHKIL QILISH. *PEDAGOGIK TADQIQOTLAR JURNALI*, *3*(1), 100-103.

7. Jalolov T. S. & Munirov J. J. (2025). TA’LIM JARAYONIDA VIRTUAL REALLIK ASOSIDA INTERAKTIV DARSLARNI TASHKIL ETISHNING SAMARADORLIGI. Development Of Science, 5(1), pp. 104-111. https://doi.org/0

8. MUNIROV, J. (2025). TRANSFORMING SOFTWARE DEVELOPMENT WITH AI-POWERED CODE GENERATION TOOLS. *ИКРО журнал*, *15*(01), 230-232.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 10, October 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

9. MUNIROV, J. (2025). ORGANIZING PRACTICAL LEARNING PROCESSES USING VIRTUAL REALITY TECHNOLOGIES. *PEDAGOGIK TADQIQOTLAR JURNALI*, *3*(2), 74-77.

10. Munirov, J. J. (2025). CREATING TELEGRAM BOTS WITH JAVASCRIPT. Recent scientific discoveries and methodological research, 2(6), 8-13.