Comparing Open Source with Software Code Generated by AI Tools from Software Maintainability Quality Factor Perspective

HAMED J. FAWAREH[1], HAZIM M. AL-SHDAIFAT[1], GHASSAN SAMARA[2]
[1]Department of Software Engineering,
Zarqa University,
JORDAN

[2]Department of Computer Science,
Zarqa University,
JORDAN

Abstract: - Artificial Intelligence (AI) has made great strides in various industries, including software development, with tools such as ChatGPT transforming the way code is written, maintained, and optimized. This study examines the impact of AI-generated code on software quality, with a focus on maintainability, code complexity, and documentation quality. Comparing AI-generated code with open-source code from GitHub for three tasks of varying difficulty (easy, medium, and hard), we evaluated key metrics, including the maintainability index (MI), lines of code (LOC), cyclic complexity (CC), Halstead volume (V), and comment ratio. The findings indicate that AI-generated code is usually more verbose its cyclical complexity tends to drop on easier tasks, reducing error rates. In a complex task maintainability prefers to support programmers with AI-generated code significantly, and better documentation according to comments. These results show that AI tools can support and enhance code quality, especially, in an industry where maintainability and simplicity are critical.

Key-Words: - AI Tools, Software Maintainability, Software Quality, ChatGPT, Code Generated, Software Metrics.

Received: March 9, 2025. Revised: June 11, 2025. Accepted: July 15, 2025. Published: November 4, 2025.

1 Introduction

Artificial intelligence (AI) has been one of the fastest-growing fields; its inception is traced back to the mid-20th century, with early applications in problem-solving and logical reasoning. For many years, AI has infiltrated every level of industry, from health and finance to manufacturing, with the introduction of machine learning, natural language processing, and autonomous systems, [1]. The growing ability of AI to imitate cognitive functions and perform tasks that used to be the sole realm of human intelligence has led it to be integrated into many aspects of everyday life, changing both consumer experiences and professional practices, [2].

The area of software development is where AI has had a lot of breakthroughs with tools for making work more productive and code quality better. Prominent examples are OpenAI's ChatGPT and GitHub Copilot, the latter developed together with Open AI. It is in this context that conversational AI, best represented by ChatGPT, supports developers by generating code snippets, making suggestions providing explanations, and debugging tips. As much as it offers real-time code suggestions, GitHub Copilot also generates full functions based on natural language descriptions, [3], [4]. Another tool is Google Gemini, which enriches the user experience in terms of information retrieval and contextual recommendation, [5].

Such tools are revolutionizing software development by automating repetitive work, providing help on demand, and making a cooperative environment between human developers and AI doable. They not only accelerate the development process but also help keep code consistent and of high quality, [6].

Code quality is a primary factor in software development and is often appraised on account of its attributes: readability, maintainability, efficiency, and accuracy. Attributes ensure that the software created is reliable, user-friendly, and easy to update or debug, [7].

Readability: This can be defined as how easily a developer may understand the program for the very important quality of collaboration and future

maintainability, [8]. AI tools such as ChatGPT and GitHub Copilot improve readability to a great extent by providing proper naming conventions that are clear and consistent, besides organizing code in a more humanly readable fashion, [9].

Maintainability: A software system is said to be maintainable when it can be easily modified to correct a deficiency, improve performance, or cope with changed environment requirements. The tools in AI that help maintain code are the automatic pinpointing and suggestion of solutions to popular bugs and potential vulnerabilities, thus reducing errors and bugs, [2], [10].

Efficiency: This case, therefore, relates to the manner in which resources are utilized at the optimal level so that the software can effectively run with respect to memory and processing power. The AI tools through some of the latest optimized refactoring suggestions provided by the tool, lead to the faster execution of the code and consequently better performance, [11].

Accuracy: It means the software correctly performs what it has to do. AI tools help to achieve this through the development of code, intended to conform to some stated requirements, and through automatic tests of code against known situations to screen for potential bugs, [12].

2 Research Background

The research in [12] focused on the study of deep learning technologies for code summarization and search to detail the capturing of semantic relations in contextual information. The main task of this article was to improve efficiency and maintainability in software development. These are tools that involve deep learning to assist in addressing some of the limitations found in earlier traditional methods, which otherwise seem blind to the structural information at the time. When these AI tools were put into general practice for any engineering tasks, this study had massive improvements to show in what was being carried out. This method was set to reduce developers' workload since it could be a source of more accurate and efficient automation tools.

Paper [13] discussed the way of using AI to fully automate the different stages of the software development lifecycle, such as coding processes, requirements gathering, and software testing. Their study proved quite beneficial in terms of increases in efficiency, accuracy, and maintainability by the use of machine learning, natural language processing, and deep learning. The AI technologies provided may not specify the programming languages, but there appears to be a general emphasis on them. Researchers in [13] discussed that the approaches driven by AI understand better than others and "refine" the needs of users for software quality. The comprehensive analysis had details of the deep ability of AI to totally alter the process of software development.

The research in paper [14] investigated the effect of code commenting on software quality by focusing on readability and maintainability. Moreover, this study showed some of the areas that create automatic comments, which are consistency, classification, and quality rating. This research [14] was conclusive documentation that improved collaboration and made future efforts of maintenance manageable. With regard to readability and maintainability, the study established that there were substantial roles played by comments in the total quality of a software application.

Paper [15] experiment series has proven that GPT-4 can be easily utilized to generate code. Although the code's efficiency and accuracy are likely to be significantly improved with GPT-4, it is still necessary to conduct thorough human validation on most codes produced. While many empirical tests were flawed, they did find improvement in refactoring abilities for GPT-4 metrics. The research study highlighted the strength of current AI-assisted coding methods and their inability to achieve perfection without human supervision. Suggested significant information about how AI is being incorporated into coding activities.

The research paper in [16] explored the impact of AI code generators, specifically OpenAI Codex, on novice programmers in introductory programming courses. They conducted a controlled experiment with 69 participants aged 10-17, who worked on Python code-authoring tasks with and without access to Codex. The study found that using Codex significantly improved code-authoring performance and did not negatively impact subsequent manual code-modification tasks participants who used Codex performed slightly better in retention post-tests. The research highlighted the potential of AI tools to enhance learning and code quality for novice programmers.

The approach given in [17] is automatically enhance code readability, based on deep learning and, more specifically, the T5 model. A methodology was developed to identify readability-improving commits, and a dataset of a large volume, 122,000 commits, was produced, available from GitHub. They report that training T5 on emulating developer actions for readability enhancement

yields perfect prediction accuracy in a range of 21% to 28%. The manual analysis concluded that a significant readability improvement could be achieved for most of the cases without introducing any change to the code behavior manually. This paper proposed a comprehensive approach to automate readability improvement in the code.

The research paper [3] compared AI-assisted code generation technologies like GitHub Copilot, Amazon Code Whisperer, and ChatGPT using the HumanEval Dataset. They measured code quality measures such as validity, correctness, security, dependability, and maintainability. The results are such that the code generated by ChatGPT was proper 65.2% of the time, while with GitHub Copilot and Amazon Code Whisperer, it was 46.3% and 31.1%, respectively. This allows practitioners to gain a deeper understanding of the strengths and weaknesses of the tools. The findings emphasized that the proper tool should be used to improve code quality.

Studied AI tools in [18] used in software quality assurance followed by software companies from Pakistan. Such studies emphasized the benefits available from AI with regard to the optimization of QA strategy and business processes. This is despite only 30% of such companies currently applying AI-based tools, a scenario where major challenges were identified and solutions proposed in improving SQA with modern machine learning models. It described the important role played by AI in enhancing software development as well as quality assurance processes.

3 Approach Taken

The approach taken in this paper starts with the initial plan of applying ChatGPT as the primary assistant for programming in the code generation stage. The programming languages selection was based on their prominence in the open-source community and their suitability for the task at hand; the ChatGPT tool and the selection languages are essential for generating a source code and later for evaluation. Following the identification of code generation assignments, the researcher used ChatGPT to develop source code-specified tasks with three levels (Easy, Medium, and Hard). Later they compared the generated AI code for specified tasks to the open-source code produced by human developers which is available as open source in GitHub sources.

A feasible method for comparing is to use open-source code from carefully chosen GitHub sources. This code is chosen for a variety of programming tasks and is classified into three levels based on difficulty level. This range provides a comprehensive assessment of ChatGPT's effectiveness at different levels of difficulty (Easy, Medium, and Hard). Using maintainability quality factors, which included Maintainability Index (MI), Comment Percentage, Lines of Code (LOC), Halstead Volume (V), and Cyclomatic Complexity (CC) as criteria, we were able to compare ChatGPT code with open-source code. The maintainability of code quality was influenced by the comparison with AI tools. The discussion will move on to the outcomes, following two distinct types of case studies maintainability quality factors of the code. Nevertheless, both aspects are discussed.

To evaluate the maintainability of the AI code generated by ChatGPT and compare it with human code used in this study several metrics were applied to evaluate different aspects of the code to ensure that it can be modified and maintained over time. Common metrics used to assess maintainability include the following:

Maintainability Index (MI): This metric provides an overall score indicating the level of maintainability of software, it is calculated using factors such as code complexity, lines of code, and comments, higher MI values indicate better maintainability.

Maintainability Index Formula (MI) = 171 — 5.2 * ln (Hashed Volume) — 0.23 * (CC: Cyclomatic Complexity) −16.2 * (% Comments of Lines Code).

where:

MI: Maintainability Index.

Ln: natural logarithm.

Halstead Volume: measuring the size code and its complexity.

CC: Cyclomatic Complexity: measuring the control flow code complexity.

Cyclomatic Complexity (CC): Using the Cyclomatic Complexity metric to calculate the independent linear paths using the Generated code, the evaluation of his matric depends on the percentage where the higher level of complexity indicates harder-to-maintain and more complex code and vice versa.

Cyclomatic Complexity Formula: $CC=E-N+2*P$

Where:

E: edges number

N: Node number

P: connected components number in the control flow graph.

Halstead metrics: This set of metrics (including Halstead size, difficulty, and effort) measures complexity based on the triggers and processors in the code, these metrics help predict how difficult it will be to maintain the code.

Halstead Metrics Formula:

$$V = N * \log2 (n) + N * \log2 (N)$$

where N represents the total number of operators and operands and n is the number of distinct operators and operands.

Lines of code (LOC): The number of lines in a program is measured using an uncomplicated instructional method. The complexity of conservation is often indicated by an advanced number of lines of code, despite not being directly linked to quality.

Formula: Simply the number of all non-null and non-empty lines of code in a software system.

Percentage of comments: This metric indicates the proportion of the code base that consists of comments that help explain the code, a higher percentage of comments generally enhances readability and maintainability, especially for developers unfamiliar with the details of the project.
Formula
Comments percentage = (number of comment lines/total code lines) *100.

4 Result and Discussion

Two types of code were examined in the study regarding the maintainability factor for code quality, with ChatGPT-generated code comprising three test cases (Easy, Medium, and Complex). Three distinct test cases were utilized in the second type of code, which was open-source code accessible on GitHub. Additionally, the result analysis is based on five metrics: Maintainability Index (MI), Lines of Code (LOC), Cyclomatic Complexity (CC), Halstead Volume (V), and Comment Percentage.

Case Study #1 Basic Calculator (Easy): There are five matrices applied to the code generated by ChatGPT along with the open-source codes as shown in Table 1. The result of our experiment is the follows:

- **Maintainability Index (MI):** The MI for the GitHub code was found to be 83.21, and for the ChatGPT-generated code, it was a little lower at 78.7. This means that, relatively, the GitHub code had a slight edge over the ChatGPT code in terms of being maintainable, with a ratio of 1.06:1 in favor of the former.
- **Lines of Code (LOC):** ChatGPT code contained 45 lines, which is higher by 34 lines over the GitHub code; hence, the ratio would be 1.32:1, which means that ChatGPT solution is more verbose.
- **Cyclomatic Complexity (CC):** ChatGPT code was 3 in CC, which is way lower than that of the GitHub code, which stands at 6. That means if we consider the ratio of 1:2 regarding complexity, ChatGPT gave positive output with simpler code.
- **Halstead Volume (V):** The volume metric came in higher for ChatGPT at 312.75 versus 276.25 for GitHub. This is a ratio of 1.13:1, The result is that the code supplied from ChatGPT was more complex than the GitHub code in terms of operations and operands,
- **Comment Percentage:** Both codebases had 0% comments, showing no difference in documentation.

Table 1. Simple calculator (easy)

Metrics	*GitHub*	*ChatGPT*
Comment Percentage	0%	0%
Maintainability Index (MI)	83.21	78.7
Lines of Code (LOC)	34	45
Cyclomatic Complexity (CC)	6	3
Halstead Volume (V)	276.25	312.75

Case Study #2 Integer to Roman (Medium): For the case study considered, we applied a code of average complexity from case study #1, and then five metrics were applied to code generated through ChatGPT and also the open-source code, the results of which are shown in Table 2. The result of our experiment is the follows:

- **Maintainability Index (MI):** The MI of ChatGPT is slightly more, at 78.418, compared to GitHub's 76.452, giving a ratio of 1.03:1 in favor of ChatGPT, which shows marginally better maintainability.
- **Lines of Code (LOC):** The LOC for the ChatGPT code was 58, while that for the GitHub code was 45, the ratio was 1.29:1, showing that the ChatGPT code was more verbose.

- **Cyclomatic Complexity (CC):** The CC for both codes was the same, which is 10. It indicates the same amount of logical complexity.
- **Halstead Volume (V):** The code for ChatGPT presented a very low volume of 115.234, as opposed to GitHub, which is 362.04. The ratio here was 1:3.14, marking that code in ChatGPT is less complex in operation and operands.
- **Comment Percentage:** The ChatGPT code had 10% comments and the GitHub code had 0%, which obviously documented the ChatGPT code well.

Table 2. Integer to Roman

Metrics	*GitHub*	*Chatgpt*
Comment Percentage	0%	10%
Maintainability Index (MI)	76.452	78.418
Lines of Code (LOC)	45	58
Cyclomatic Complexity (CC)	10	10
Halstead Volume (V)	362.04	115.234

Case Study #3 Median of Two Sorted Arrays method (Complicated): For this case study, we used more complex codes as compared to case studies 1 and 2. We tested the ChatGPT-generated code and open-source code against the five matrices, and the results are presented in Table 3. The result of our experiment is the follows:

- **Maintainability Index:** The ChatGPT MI is considerably higher than that of GitHub: 131.31 to 83.66, in a ratio of about 1.57:1, this indicates much better maintainability for the ChatGPT code.
- **Lines of Code (LOC):** ChatGPT code was 39 lines just slightly higher than GitHub's 36 lines by a ratio of 1.08:1 indicating an insignificant increase in verbosity.
- **Cyclomatic Complexity (CC):** Both the codes had the same CC of 4, showing identical complexity in decision-making paths.
- **Halstead Volume (V):** ChatGPT code = 276.52, GitHub = 237.06, Ratio = 1.17:1. Here, the evidence is greater since it presents a bigger complexity in the number of operations and operands made.
- **Comment Percentage:** ChatGPT code had 25% comments, while those of GitHub had none, showing documented work much better.

Table 3. Median of Two Sorted Arrays method

Metrics	*GitHub*	*Chatgpt*
Comment Percentage	0%	25%
Maintainability Index (MI)	83.66	131.31
Lines of Code (LOC)	36	39
Cyclomatic Complexity (CC)	4	4
Halstead Volume (V)	237.06	276.52

Generally, ChatGPT code Maintainability for all case studies demonstrated better or at least similar maintainability, especially in more complex case studies. For the complicated case study, the MI was significantly higher, which means that the code produced by ChatGPT can be more appropriate for high-maintainability tasks. Code Complexity: ChatGPT solutions were more verbose (more LOC), but most often had less Cyclomatic Complexity, especially for simpler tasks, which could benefit from having fewer potential errors and hence better code simplicity. Documentation: There are comments throughout the code base of ChatGPT, especially in complex tasks where documentation is critical for maintenance and understandability.

5 Conclusion

This paper discussed the comparative performance of AI-generated code using ChatGPT with human-developed code taken from a variety of open-source GitHub repositories on several metrics of code quality and maintainability. It was seen that, while on average more verbose, ChatGPT-written code provides a significant maintainability advantage, especially for more complex tasks. As an example, low cyclomatic complexity in the easier task hints at the fact that AI-generated code can be simple and, therefore, potentially less error-prone, while a high maintainability index in the harder task reveals a good fit for applications that are highly maintainable. Also, comments in the ChatGPT code help to provide better documentation, which is important in future updating and collaboration on the code.

Acknowledgement
This research is funded fully by Zarqa University - Jordan.

References:

[1] Dwivedi, Y., Hughes, L., Ismagilova, E., G. A.-I. J. of, &, U. (2021). Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice, and policy. *International Journal of Information Management,* Vol. 57, April 2021, https://doi.org/10.1016/j.ijinfomgt.2019.08.002.

[2] Hamed Fawareh, Hazim M. Al-Shdaifat "Investigating AI-generated code on the impact on software efficiency code quality factor", *Sustainable Data Management,* Vol. 171, pp. 89–97, 2025, Springer. https://doi.org/10.1007/978-3-031-83911-5_8.

[3] Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. https://doi.org/10.48550/arXiv.2304.10778.

[4] Saeidnia, H. R. (2023). Welcome to the Gemini era: Google DeepMind and the information industry. Library Hi Tech News. https://doi.org/10.1108/LHTN-12-2023-0214.

[5] CG Balaji, Md Imran, S Vamsi Ujwala, Ajinkya P Edlabadkar, Hamed Fawareh, "An Improved Neural Classification Logic to Predict Kidney Disease Identification Using Medical Image Processing Principles", *International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES),* 2024, pp. 1-6. https://doi.org/10.1109/ICSES63760.2024.10910605.

[6] Guo, P. J., & Cooper, R. (2023). Six Opportunities for Scientists and Engineers to Learn Programming Using AI Tools Such as ChatGPT. *Computing in Science and Engineering,* 25(3). https://doi.org/10.1109/MCSE.2023.3308476.

[7] Hamed Fawareh, "Software quality model for maintenance software purposes", *International Journal of Engineering Research and Technology,* 2020, 13(1), pp. 158-162.

[8] Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. https://doi.org/10.48550/arXiv.2302.06590.

[9] Hamed Fawareh, Hazim M Al-Shdaifat, M Al-Refai, Faid AlNoor Fawareh, Mohammed Khouj, "Investigates the Impact of AI-generated Code Tools on Software Readability Code Quality Factor" *2024 25th International Arab Conference on Information Technology (ACIT).* https://doi.org/10.1109/ACIT62805.2024.10876897.

[10] Ajith Abraham, Bineet Kumar Gupta, Satya Bhushan Verma, Archana Sachindeo Maurya, Mohammad Husain, Arshad Ali, Sami Alshmrany, Sanjay Gupta, "Improvement of Translation Accuracy for the Word Sense Disambiguation System using Novel Classifier Approach", *The International Arab Journal of Information Technology (IAJIT),* Vol. 21, Number 06, November 2024.

[11] Nascimento, N., Alencar, P., & Cowan, D. (2023). Comparing Software Developers with ChatGPT: An Empirical Investigation. https://doi.org/10.48550/arXiv.2305.11837.

[12] Zhang, T., Sun, X., Zheng, Z., & Li, G. (2022). Intelligent analysis for software data: research and applications. *In Frontiers of Information Technology and Electronic Engineering* (Vol. 23, Issue 5). https://doi.org/10.1631/FITEE.2230000.

[13] Devalla, S., & Yogix, M. K. (2023). Building Trust in AI -A Simplified Guide to Ensure Software Quality. *Journal of Soft Computing Paradigm,* 5(3). https://doi.org/10.36548/jscp.2023.3.001.

[14] Perera, D. T. W. S., Premathilake, H. T. M., Thathsarani, K. P. H., Nethmini, R. H. T., De Silva, D. I., & Samarasekara, H. M. P. P. K. H. (2023). Analyzing the Impact of Code Commenting on Software Quality. *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT* 2023. https://doi.org/10.1109/ICCCNT56998.2023.10307948.

[15] Poldrack, R. A., Lu, T., & Beguš, G. (2023). AI-assisted coding: Experiments with GPT-4. https://doi.org/10.48550/arXiv.2304.13187.

[16] Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. *Conference on Human Factors in Computing Systems - Proceedings.* https://doi.org/10.1145/3544548.3580919.

[17] Vitale, A., Piantadosi, V., Scalabrino, S., & Oliveto, R. (2023). Using Deep Learning to Automatically Improve Code Readability. *Proceedings - 2023 38th IEEE/ACM International Conference on Automated Software Engineering,* ASE 2023. https://doi.org/10.1109/ASE56229.2023.00112.

[18] Qazi, S., Memon, M. S., Ali, A., & Nizamani, S. (2022). Role of Artificial Intelligence (AI) Tools for Assuring Quality in Software. *Journal of Southwest Jiaotong University,* 57(2). https://doi.org/10.35741/issn.0258-2724.57.2.5.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US