Manufacturing Science and Education 2025

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 68, Issue Special III, August, 2025

CHATGPT VS GITHUB COPILOT – A REQUIREMENT-BASED TEST CASES GENERATION CAPABILITIES EVALUATION

Codrina-Victoria LISARU, Claudiu Vasile KIFOR

Abstract: *With the rapid development of AI-based tools, such as ChatGPT and GitHub Copilot, the desire to explore them and evaluate their usability is also developing, to use their capabilities to make human work more efficient. The purpose of this paper is to assess whether ChatGPT and GitHub Copilot can generate precise, correct, and enough test cases based on requirements, to help and encourage the software testing community to use these AI-based tools. This paper compares the answers of both AI-based engines and offers a brief overview of the test cases generation capabilities.*

Keywords: *ChatGPT, GitHub Copilot, Requirements, Test Cases, Software Quality*

1. INTRODUCTION

In the last few years, AI-based assistants have gained popularity, and they are getting even more popular every day, becoming of great help in software engineering [1]. AI has a growing key role in the software quality testing industry, with multiple benefits covering many areas [2]. A special place is occupied by generative artificial intelligence, which is, according to [3], a "branch of AI focused on models that create new content", and has multiple usages, including generating text, ideas, and data used in different areas of interest, being a great opportunity also for the testing of software products quality aspects. Besides generative AI, other types of AI categories can be applied in software testing, with great benefits, as described in [4] and [5]. Comparative studies have shown that AI has proved to be an effective instrument in multiple areas of software development, automation-based testing, test cases prioritization [6] and regression testing [7] but with a less focus on requirements-based testing, as will be presented in the "Related work" section. This paper aims to complement the studies on the usage of AI in requirements-based software testing by addressing the generation of test cases based on requirements.

The application field of this study mainly involves testing the quality of software products, in requirements-based testing environments, which are present in a variety of industries. The paper also evaluates the capability of AI tools to understand a requirement, generate a reasonable number of test cases based on a problem, and categorize answers when requested. These topics can be applied and used in other fields than software testing.

This paper is organized as follows. Section 2 presents related work and discusses the problems that exist associated with the generation of code and test cases using AI assistants. Section 3 presents the research questions, the design of the experiment, and the construction of the dataset. Section 4 presents the results of the experiment and discussion on the topic. Section 5 presents the study limitations, encountered problems, and future research, followed by the conclusion of the study, in section 6.

2. RELATED WORK

Agnia S. et al. conducted in [1] a study based on opinions collected from an impressive number of programmers, regarding multiple topics, including: "(a) implementing new features, (b) writing tests, (c) bug triaging, (d) refactoring, and (e) writing natural-language artifacts, as well as their individual stages.", and providing an overview of the LLMs' popularity and usage.

Davide T. conducted in [8] an empirical evaluation ("through systematic observation, experimentation, and experience in real-world coding problems") of 3 different Large Language Model (LLM) Engines, to compare the generated source code for 3 complex coding problems, by evaluating its correctness and quality. A similar approach is described in [9], in which it is evaluated the "ChatGPT's usability as a code generation tool".

Sajed J. et al. conducted in [10] a study based on the ability of ChatGPT to answer questions correctly and to provide correct explanations, but also evaluating how correct the generated answer-explanation pairs are.

Yihao L. et al. [11] conducted an evaluation of Large Language Models (LLMs) in the software testing field, with a focus on "test case generation, error tracing, and bug localization across twelve open-source projects", concluding that LLMs can be effective in detecting errors, but their answers need to be carefully reviewed by a human. The study also provides a generous description of the used methods and offers important information and conclusions regarding the LLMs in software testing.

A. Millam and C. Bakke offer, in [12], an interesting overview regarding the use of AI by programmers, how the AI technology can be improved, and considers ethical issues when it comes to AI. The AI used in the study are ChatGPT and Bing Chat, which "were asked the same question, their responses were recorded and the percentage of each chatbot's code that was extraneous was calculated.". Based on this methodology, qualitative factors were calculated and presented in the paper.

M. Staron et al. provide in [13] an overview of the recent research on testing, debugging, and log analysis using AI tools by analyzing the papers on these topics presented at two conferences, and concluding how useful can be the generative AI tools, such as Stable Diffusion and ChatGPT.

Shreya B. et al., in [14], investigate the effectiveness of Large Language Models in the context of unit test script generation, and "how the generated test cases compare with those generated by an existing unit test generator", concluding that both tools demonstrate very similar results, with no significant differences. In a similar manner, *Jacob H. and Demian F.,* in [15], and *Maurizio L. et al.* in [16], focus on the task of generating test cases for mobile applications [15] and web end-to-end testing in [16], using artificial intelligence tools, providing measurements for their feasibility and effectiveness.

3. METHODOLOGY

3.1 Research questions

This paper aims to answer 3 research questions:

RQ1. Which one offers better test cases, ChatGPT or GitHub Copilot?

RQ2. To what extent can testers rely on AI assistants to help in creating test cases based on requirements?

RQ3. Which one is more suitable for the software testing activity? ChatGPT or GitHub Copilot?

3.2 Design of the experiment and construction of the dataset

This paper aims to offer an overview of the ChatGPT [17] and GitHub Copilot [18]'s capabilities to generate test cases based on requirements, which consist of a set of problems, separated into distinct chapters, as follows: Working with lists, Arithmetic, Logic and codes. The source of the problems is Werner Hett's set of 99 Prolog problems [19], a well-known set of programming problems, more precisely the first 3 sections, with small adaptations to make the problems more general.

A total of 43 problems were selected for this experiment (P01 – P50), and only one problem was excluded (P13) due to redundancy. Problems P29-30 and P42-45 don't exist in the set of problems. Then, 2 popular AI engines were selected to conduct the experiment: ChatGPT and GitHub Copilot, as they have been evaluated as invaluable assets and achieved great rates of success in other studies [1][8]. For each of the problems, ChatGPT and GitHub Copilot were asked 3 times each to: "*Generate the most representative test cases for the* *following requirement, mentioning the used testing technique: …*" followed by the problem's text and its correspondent number (P01, P…), and then asked to evaluate its confidence level: "*Rate the confidence level of your response, from 1 (being the lowest) to 10 (being the highest).*", after the AI assistant answered to the first issue. The confidence query was applied in similarly in [10] and was adopted in this study to discover if there is any correlation between the correctness of the answer with the self-evaluation grade. A total number of 258 interrogations resulted from this (129 for ChatGPT, 129 for GitHub Copilot). Each query was made in a separate chat, in order not to influence the response for a problem, using the previous responses. More problems were interrogated in the same chat only when they were dependent on each other, using the same interrogation structure presented earlier (question – answer – self-evaluation request – answer). The same GPT model was used for the interrogations, on both ChatGPT and GitHub Copilot: GPT-4o. The results of the interrogations were saved and can be provided at demand.

4. RESULTS AND DISCUSSION

The results of the interrogations were analyzed and synthesized and will be briefly presented as follows.

From a total number of 43 problems, 129 interrogations for each AI engine, resulted in 1403 test cases generated by ChatGPT, and 953 test cases generated by GitHub Copilot. These test cases were subjectively evaluated one by one by the author, verifying their correctness and input – expected output consistency, structure of the response, and common test cases for 3 interrogations. A summary of the results is presented in Table 1 – Results of the experiment.

Table 1

Results of the experiment

	ChatGPT	**GitHub Copilot**
Total number of problems	43	43
Total number of interrogations	129	129
Total number of generated test cases	1403	953
Average number of generated test cases per problem	10.88	7.39
Average delta (max no. of generated TCs – min no. of generated TCs, per problem)	4.02	2.21
Correct and complete test cases	1395	952
Specific test cases – input and expected output defined, values	1403	940
General test cases – no specific input and expected output defined, no values	0	13
Total number of redundant test cases (out of the total number of generated test cases)	69	21
Average common test cases (overall, for all the problems)	6	5
Code suggestion (out of the total number of interrogations)	1	50
Self-evaluation average grade	9.00	9.45
Self-evaluation justification (out of the total number of interrogations)	126	114

For ChatGPT, out of 1403 test cases, 99.50% (1396) were correct and complete, meaning that the formulated testing scenario, the example input, and the expected output are correct and complete, and correctly respond to the formulated requirement. The rest of the test cases, which were classified as incorrect or incomplete, had either missing expected output or incorrectly calculated values for the expected output. For GitHub Copilot, out of 953 test cases, 99.9% (952) were correct and complete, only one test case had the expected output wrongly calculated. Overall, ChatGPT generated more test cases than GitHub Copilot, with an average number of generated test cases per problem of approx. 11, in comparison with approx. 7. In terms of redundancy, ChatGPT generated more redundant test cases (repeated tests or input and expected output that test the same scenario), a percentage of 4.92%, in comparison with 2.20% redundant test cases generated by GitHub Copilot. The number of common tests generated by ChatGPT (6) is very close to the number of common tests generated by GitHub Copilot (5). The number of common tests was evaluated by matching the tests that test the same scenario among the 3 interrogations of each problem. Regarding the code suggestion for each interrogation, only for one interrogation, ChatGPT proposed a sequence of test code, in comparison with GitHub Copilot, which offered test code suggestions for 50 of the interrogations, a fact that highlights the code-oriented construction of this AI engine. The self-evaluation average grade and the justification of the grade don't offer much information. Overall, the self-evaluation justification is very well-built and covers valid arguments. Lower grades were given by the AI engine when a greater degree of uncertainty occurred in its answer.

Regarding the mention of the used testing technique, ChatGPT provided a great variety of testing methods for each interrogation, in comparison to GitHub Copilot, which only provided a few, popular testing methods.

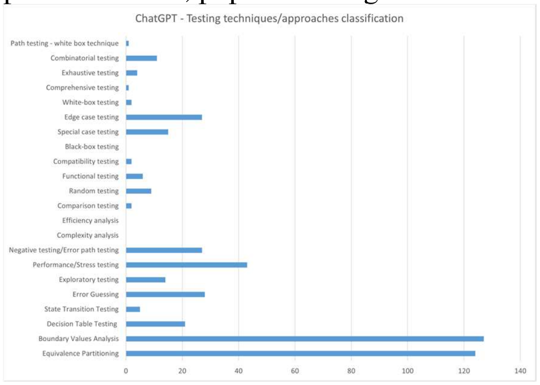

Fig. 1. ChatGPT – Testing techniques/approaches classification

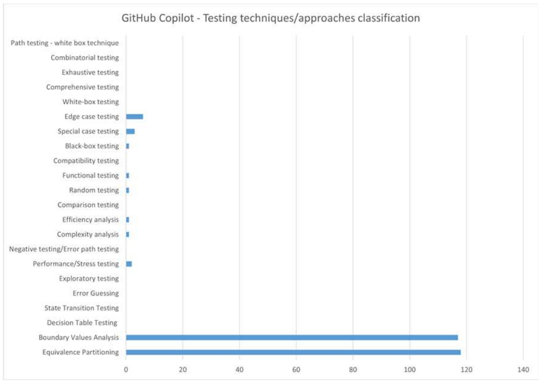

Fig. 2. GitHub Copilot – Testing techniques/approaches classification

The above figures present the number of interrogations that use the specified testing method/approach. Some testing methods are general, such as "Black-box testing", "White-box testing", and more are specific.

The most popular testing techniques used by ChatGPT are Equivalence partitioning, Boundary values analysis, Error guessing, Performance/stress testing, Negative testing, Edge case testing, Decision table testing, and Exploratory testing, followed by other techniques with a lower degree of usage. On the other side, GitHub Copilot classifies the generated test cases using Boundary values analysis and Equivalence partitioning in most of the cases, generalizing, this way, its answers be suitable in more contexts.

Overall, both AI assistants offer comprehensive test cases, with a high degree of correctness and low rates of redundancy, prove a good understanding of the requirements, and generate appropriate test cases, with a high coverage of scenarios, this way being considered very helpful tools in the work of software testers, this way answering the research questions **RQ1** and **RQ2.** Although, a minimum human review of the generated answers is necessary. To answer **RQ3**, considering the code orientation of GitHub Copilot, this tool would be more appropriate for automation software testing, rather than manual testing. But when it comes to approaching general testing scenarios, both AI tools are suitable. Given the average delta (max number of TCs – min number of TCs per problem), it would be useful for a tester to interrogate the AI multiple times for the same issue, to cover as many scenarios as possible.

5. STUDY LIMITATIONS AND FUTURE RESEARCH

The first limitation of the study related to the use of a limited range of problem types. A future improvement would be to include a greater diversity of issues. Another issue might be represented by the generality of the problems provided to both AI tools, if the problems would be more specific, and to be implemented in a certain programming language, the AI tools might offer more specific answers. Also, the chapters referring to binary trees, multiway trees, graphs, and miscellaneous problems, from the same set of problems are intended to be included in a future study, being of a higher difficulty.

6. CONCLUSION

This study offers a summary of the topic of test case generation based on general requirements, with no references to any specific programming language, using 2 popular AI tools, ChatGPT and GitHub Copilot. The results of the analysis are favorable for the software testing area, with high rates of correctness and completeness of the generated test cases and low rates of redundancy, incompleteness, and wrongness, and provide a favorable point of view over the usage of AI in the field of software products testing. AI can successfully contribute to testing the quality of software products.

7. REFERENCES

[1] Agnia S., Yaroslav G., Timofey B., Iftekhar A., *Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward*, Information and Software Technology 178, ISSN 0950-5849, DOI *DOI https://doi.org/10.1016/j.infsof.2024.107610,* 2025

[2] Hussam H., Ahmad H., Mohammad L., The Impact of Artificial intelligence on Software Testing, 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), IEEE, Amman, Jordan, April 2019, pp. 565 – 570, DOI: 1109/JEEIT.2019.8717439.

[3] Lucas L., Ron V., *Generative Artificial Intelligence and the Future of Software Testing,* Computer, Volume 57, Issue 1, IEEE, pp. 27-32, January 2024, DOI https://doi.org/10.1109/MC.2023.3306998 2024

[4] Rui L., Antonio Miguel Rosado da C., Jorge R., Artificial intelligence Applied to Software Testing: A Literature Review, 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, June 2020, pp. 1-6, ISBN: 978-989-54659-0-3,DOI:10.23919/CISTI49556.2020.9141124 2020.

[5] Minimol Anil J., Automating and Optimizing Software Testing Using Artificial Intelligence Techniques, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12 (5), , pp. 594 – 602, 2021

[6] Kohani K. M., Nurezayana Z., Mohd Zanes S., Test Case Prioritization Using Swarm Intelligence Algorithm to Improve Fault Detection and Time for Web Application, Journal of Soft Computing and Data Mining, 4(2), pp. 59-66, October 2023

[7] Shahbaa I. K., Raghda A., A review paper: optimal test cases for regression testing using artificial intelligent techniques, International Journal of Electrical and Computer Engineering (IJECE), Vol. 13, No. 2, April, pp. 1803 – 1816, ISSN: 2088 - 8708, 2023

[8] Davide T., Studying the Quality of Source Code Generated by Different AI Generative Engines: An Empirical Evaluation, Future Internet 16, no 6: 188, DOI https://doi.org/10.3390/fi16060188 2024

[9] Tanha M., Hong Z., User Centric Evaluation of Code Generation Tools, 2024 IEEE International Conference on Artificial Intelligence Testing (AITest), IEEE, Shanghai, China, July 2024, pp 109-119, 2024

[10] Sajed J., Suzzana R., Thomas D. L., Kevin M., Wing L., ChatGPT and Software Testing Education: Promises & Perils, 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, Dublin, Ireland, April 2023, pp. 4130-4137, 2023

[11] Yihao L., Pan L., Haiyang W., Jie C., W. Eric W., Evaluating large language models for software testing, Computer Standards & Interfaces, Volume 93, Elsevier, April 2025, ISSN 0920-5489, 2025

[12] Andrew M., Christiane B., Coding with AI as an assistant: Can AI generate concise computer code?, Journal of Information Technology Education: Innovations in Practice, Volume 23, 2024

[13] Miroslaw S., Silvia A., Gregory G., Alexander S., Testing, Debugging, and Log

Analysis with modern AI tools, IEEE Software, Volume 41(2), March - April 2024, pp. 99-102, 2024
[14] Shreya B., Tarushi G., Dhruv K., Pankaj J., Unit Test Generation using Generative AI: A Comparative Performance Analysis of Autogeneration Tools, in Proceedings of the 1st International Workshop on Large Language Models for Code, ACM, Lisbon, Portugal, April 2024, pp.54-61, 2024
[15] Jacob H., Demian F., Generating Software Tests for Mobile Applications Using Fine-Tuned Large Language Models, 2024 IEEE/ACM International Conference on Automation of Software Test (AST), IEEE, Lisbon, Portugal, April 2024, pp 76-77, ISBN 979-8-4007-0588-5/24/04, 2024
[16] Maurizio L., Filippo R., Hafiz Zeeshan Y., Boni G., AI-Generated Test Scripts for Web E2E Testing with ChatGPT and Copilot: A Preliminary Study, 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), ACM, Salerno, Italy, June 2024, pp. 339-344, ISBN 979-8-4007-1701-7/24/06, 2024
[17] OpenAI, ChatGPT: https://chatgpt.com/ accessed in 2025
[18] GitHub, GitHub Copilot: https://github.com/copilot accessed in 2025
[19] Werner H., P-99: Ninety-Nine Prolog Problems, https://www.ic.unicamp.br/~meidanis/courses/mc336/2009s2/prolog/problemas/ accessed in 2025

ChatGPT vs GitHub Copilot – O evaluare a capacităților de generare a cazurilor de testare bazate pe cerințe

Odată cu dezvoltarea rapidă a instrumentelor bazate pe Inteligența Artificială, cum ar fi ChatGPT și GitHub Copilot, se dezvoltă și dorința de a le explora și de a le evalua gradul de utilizare, cu scopul de a le folosi capacitățile pentru a face munca oamenilor mai eficientă. Scopul acestei lucrări este de a evalua dacă ChatGPT și GitHub Copilot pot genera cazuri de testare precise, corecte și suficiente pe baza cerințelor, pentru a ajuta și încuraja comunitatea de testare software să utilizeze aceste instrumente bazate pe Inteligența Artificială. Această lucrare compară răspunsurile ambelor motoare bazate pe Inteligența Artificială și oferă o scurtă prezentare a capacităților de generare a cazurilor de testare.

Codrina-Victoria LISARU, PhD Student, Lucian Blaga University of Sibiu, Department of Industrial Engineering and Management, codrinavictoria.ignat@ulbsibiu.ro, +40743941716
Claudiu V. KIFOR, Prof. PhD., Lucian Blaga University of Sibiu, Faculty of Engineering, Department of Industrial Engineering and Management, claudiu.kifor@ulbsibiu.ro