MITRE | SOLVING PROBLEMS FOR A SAFER WORLD®

Source: MITRE

Software Engineering With Generative Artificial Intelligence Tools

MITRE FINDINGS AND RECOMMENDATIONS – FALL 2023

By Rock Sabetto, Sujay Kandwal, and Devesh Agarwal

GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS AND SOFTWARE ENGINEERING

The rapid rise of Generative Artificial Intelligence (AI) tools is altering the software engineering profession. These tools interpret user requests, referred to as prompts, and analyze them against large volumes of training data to generate responses. Software engineers are using Generative AI tools today to support work across the lifecycle, for tasks such as requirements generation, software development, and test planning. MITRE understands the growing power of these tools and believes that they are an important part of the future.

It is incumbent upon software engineers to learn how to use Generative AI tools effectively and safely. In July 2023, the MITRE Software Engineering Innovation Center released a Technical Advisory detailing the risks associated with these tools.[1] This paper contains initial findings and recommendations from a series of Generative AI software development tests MITRE conducted in the Fall of 2023.

TEST OVERVIEW

MITRE software engineers tested three Generative AI tools: OpenAI ChatGPT, Google Bard, and Microsoft BingAI.[2]

Tests included a combination of MITRE Software Engineer interview questions[3] and a problem sourced from the LeetCode[4] developer learning platform.

MITRE TESTED GENERATIVE AI TOOLS AGAINST A HUMAN SOFTWARE ENGINEER

Tests varied in complexity, and included basic questions on software engineering fundamentals, simple Python and Java functions, and more intricate C++ code development.[5] MITRE evalutated tool performance by measuring speed and accuracy of the developer and the generative AI tool to complete the task. To measure speed, MITRE captured wall-clock time-to-complete for the human software developer and each AI tool. To measure accuracy, MITRE used a combination of vetted problem solutions from the team's GitLab repository, code testing in Python and Java, and C++ testing in the LeetCode Integrated Development Environment (IDE).

These test results provide a preliminary basis that informed the findings and recommendations presented in this paper and can inspire more rigorous testing.

MITRE plans to conduct future testing using code completion tools (e.g., GitHub Copilot).

[1] MITRE Technical Advisory "On the use of ChatGPT, GitHub Copilot, and other Generative Large Language Models for AI Assisted Software Engineering", July 2023. Case 23-2798. Digital copy available upon request.

[2] Tool versions used for testing: ChatGPT (Used MITRE ChatGPT, an internal, secure Generative AI interface based upon OpenAI ChatGPT, Version 4); Google Bard (Public Version, updated 07/13/2023); BingAI (Public Version, updated 03/14/2023), which leverages the GPT Version 4 model.

[3] MITRE, Software Engineering Interview Repository, https://gitlab.mitre.org/t853. Accessed September 2023.

[4] LeetCode, "Strong Password Checker", https://leetcode.com/problems/strong-password-checker/. Accessed September 2023.

5 MITRE executed four test cases: 1) Python 3-part test, 2) Java Fibonacci Sequence, 3) Software Engineering Knowledge Check and 4) LeetCode Strong Password Checker. The volume of tests is not sufficient to draw widespread conclusions or statistically significant findings but provides insights for future evaluations.

FINDINGS

The capabilities of Generative AI tools to support software developers are by no means perfect, but they do offer the potential to assist developers and reduce time to complete for some development activities.

MITRE's testing shows that in some cases, Generative AI tools can solve low and medium complexity software problems faster than a professional software engineer.

While these results are exciting, they are based upon a small sample of tests and reflect the capabilities of Generative AI tools as of Fall 2023. Results will vary based upon developer skill level and rapidly evolving Generative AI tool capabilities.

Finding One: Generative AI Tools Vary.

The tools tested in this report did not perform equally. Generally, OpenAI's ChatGPT and Microsoft's BingAI performed well in tests using both Python and C++. Google's Bard was able to perform simple Python tasks but when prompted to create C++ code did not generate accurate responses. Additionally, in multiple cases with each tool, responses to identical prompts varied, sometimes significantly. Given the limited scope of testing and continuous tool evolution, MITRE is unable to definitively recommend specific tools.

Tool creators. MITRE-observed variance in tool capability that may reflect differences in model training focus by OpenAI, Microsoft, and Google. Microsoft and OpenAI operate a partnership to develop solutions collaboratively, but commercialize them independently.[6]

Tool architecture. In a general sense, the Generative AI tools use a similar large language model (LLM) architecture underpinned by a huge corpus of data. However, beyond this generalization it is difficult to assess not only the data sources (only BingAI provides citations natively for each prompt), but also the inner workings of the tools. When coupled with the fact that the models are updated at varying rates, developers should evaluate the utility of each tool for the immediate task at hand.

Tool code translation. Bard lacked the ability to create a C++ solution for the LeetCode Password Checker test case. For example, when prompted multiple times for a C++ solution, Bard responses ranged from code that would not compile to statements that Bard was unable to create C++ code. Conversely, BingAI cited a Python solution when it provided the C++ Password Checker code response. When MITRE developers researched the citation, they discovered the BingAI C++ solution used virtually identical function names and code structures as those in the Python solution. This suggests BingAI can identify solutions in one programming language and translate them to another.

Finding Two: Increasing Problem Complexity Decreases Solution Accuracy.

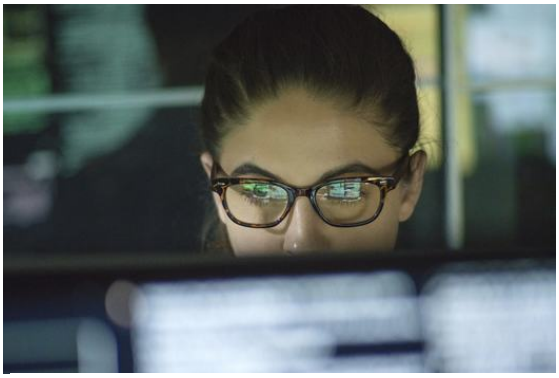
Generative AI tool performance decreases as problem complexity increases.

Decomposition works. For more complex challenges, decomposing the problem enables the creation of discrete building blocks that can be tested individually and then assembled into a complete solution.

Prompt quality matters. In general, the more precise the prompt, the more accurate the tool response. Using carefully crafted pseudocode, MITRE evaluators were able to create prompts that enabled the tools to create accurate solutions to the LeetCode problem. In this case, MITRE created an

[6] OpenAI, "OpenAI and Microsoft Extend Partnership". https://openai.com/blog/openai-and-microsoft-extend-partnership. Accessed September 2023.

800 word block of pseudocode and split it into five discrete subtasks. Only after careful review was it fed to the Generative AI tools. This aligns with the systems engineering principle of writing clear requirements.



Source: MITRE

Decomposition and debugging. Another benefit of creating discrete subtasks is that the resulting code blocks are generally shorter and easier to troubleshoot. When contrasted with the effort required to manually debug a much larger block of code containing a high number of dependencies, development time, which included prompt development, was cut by approximately 50% when compared with a human-only development effort.

Finding Three: Generative AI Tools Lack Creativity.

MITRE testing shows that in their current state, Generative AI tools struggle to produce innovative solutions.

Model source limitations. While these tools are capable of leveraging patterns in existing information to create new information, they do not currently show the ability to create novel software solutions. In the case of BingAI, the tool explicitly cites the sources it is using to create responses.

The innovation is in the prompt. While this is likely to evolve in the coming months, MITRE believes that the best method to get innovative results from the Generative AI tools is to build innovative thought into prompts. For example, if developers create truly unique pseudocode, a tool may be able to leverage its neural network to create code that implements the pseudocode and produces a truly new solution. For now, the software engineer remains the primary source for innovation.

RECOMMENDATIONS

While MITRE testing found variance in capability across well-known tools, there is little argument that they are here to stay, and that they offer potential benefits in terms of accuracy and speed. The following recommendations are based upon the experiences gained during MITRE testing.

Recommendation One: Use Generative AI Tools as Assistants.

Software engineers should use Generative AI tools to help reduce the time required to solve problems, exercising diligence to ensure safe and secure use.

Syntax can be a strength. Generative AI tools can quickly provide syntactically accurate code in a variety of languages. In cases where a developer has a complete understanding of the problem and creates pseudocode, Generative AI tools may be able to quickly convert the pseudocode into properly formatted code. For simpler, discrete coding problems, the tools may be able to provide accurate solutions in seconds, where developers may take several minutes to craft accurate code. When scaled over time and across teams, significant productivity gains may be realized.

Verify everything. As of this writing, it is evident that the human must remain in the loop in the software development process. Efforts are underway at all of the major AI development organizations to create automated interfaces, but the underpinnings of the tools are not of sufficient quality to be programmatically trusted without human

verification. Developers may turn to Generative AI tools to create test cases, but the tools are not of sufficient maturity to be used as a complete replacement for large portions of any development pipeline linked to production operations.

Comparing Generative AI tools to junior developers. New software engineers coming directly from university computer science programs have current and relevant knowledge and skills. These new professionals can produce outstanding work, but the content must often be checked by seasoned developers. The junior developer often needs additional problem context to create software that can be used a production setting. The Generative AI tool, like the new hire, must be treated with patience and provided with feedback. When work is inaccurate, the experienced developer should take the time to explain (via prompt) why the work is inaccurate to facilitate learning and growth.

Recommendation Two: Reduce Complexity to Increase Accuracy.

To solve problems of higher complexity, simply feeding the problem statement to the Generative AI tool is unlikely to work. Much like human software engineers, the tools perform more effectively when problems are decomposed into discrete elements.

Use a decomposition process to handle complexity. MITRE used a basic problem-solving approach for complex problems, decomposing the challenge and creating a series of prompts.

Don't wait to break it down. With Generative AI tools, it can be tempting to simply paste the entire problem into a prompt and hope for the best. While there is little harm in trying this at the outset, this method should be used with caution as it can lead to a lengthy cycle of bug fixes. Developers should recognize this situation and begin to break the problem down.

Create pseudocode to increase accuracy. MITRE found that the creation of pseudocode is beneficial to the Generative AI tool and leads to higher quality responses. MITRE found that the tools, in most cases, were able to rapidly convert pseudocode into working software. This forces the developer to focus on the logic needed to solve the problem. Finally, pseudocode provides an easy to understand description of the working solution that can be used in many ways: to create documentation, as a basis for software language conversion, and as a way to share the solution concept with other developers.

Recommendation Three: Use Generative AI Tools to Reduce Search Time.

The tools leverage a large corpus of training data to perform their work, which theoretically is "beyond search." While the solution is imperfect, these tools can reduce time spent searching using traditional engines (e.g., Google, Bing).

Source: MITRE

Generate initial ideas. Developers can spawn innovative thinking by prompting Generative AI tools for solution concepts when starting a new task. Initial concepts help the software engineer avoid the "cold start problem" when beginning the work. As stated earlier, the tools in the current state

are not innovative, but software engineers can use them in innovative ways.

Create draft explanations. Asking Generative AI tools to explain complex concepts using simple language can help reduce time to create supporting documentation and can help increase the overall quality of the documentation. In this way, Generative AI tools provide an advantage over many developers, who may struggle to write effective documentation or may despise the time required to create written descriptions of software.

Users must verify accuracy prior to using results. In some cases, tools do not provide citations to source information. In others, citations are incorrect. Users cannot blindly trust Generative AI tool outputs.

Recommendation Four: Create Overall Time Savings by Writing Clear Prompts and Selecting the Best Generative AI Tool for Your Project.

MITRE demonstrated through a small set of tests what has been shown via multiple industry studies and surveys: Generative AI tools offer time saving potential.

Take time to create prompts. MITRE developers were able to cut the time required to solve the LeetCode Password Checker test in half by decomposing the problem and investing time in prompt creation. Attempt to achieve similar improvements through prompt engineering. Our findings show that the clarity and accuracy of the prompt generally correlates to tool output accuracy.

Use multiple tools. Software developers must actively maintain an understanding of the strengths of various Generative AI tools. MITRE testing represents a snapshot in time with a limited sample size. Results may vary using larger test volumes. Developers should explore emergent Generative AI tools and revisit tools periodically given the speed of tool modernization. More generally, a comprehensivse assessment of Generative AI software engineering capabilities will benefit the community and should be performed in the near future.

Recommendation Five: Safety Must be Considered.

MITRE believes in the potential of Generative AI tools for software engineering, but risks cannot be ignored. The professional software engineer is responsible for the safe use of the tool.

Readers should review MITRE's Software Engineering Innovation Center Technical Advisory[7] to understand the risks associated with use of Generative AI tools for software engineering. The advisory provides developers with clear advice on the safe use of Generative AI tools.

CONCLUSION

This paper presents preliminary findings and recommendations from a limited set of test cases executed in Fall 2023. MITRE believes that software engineers can save time and improve accuracy by using Generative AI tools as assistants. In their current state, the tools require developers to completely understand problems, decompose complex problems, write clear prompts, and diligently review tool-generated code.

The tools offer clear benefits but require human expertise and oversight to safely develop production quality software.

[7] MITRE Technical Advisory, see Note 1.