Understanding Copilot X and its potential for improved DevOps practices

OLIVER SCHWALBE LEHTIHET MATHIAS NÄREAHO

oliverle | nareaho @kth.se

April 23, 2023

Contents

I/We certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

1 Introduction

One of the fundamentals of DevOps is collaboration, working together with other developers and team members to create an agile and effective environment. A common approach to collaboration is pair programming, where one developer takes on the role of the Driver (focusing on completing small goals at the keyboard) and the other acts as the Navigator (observing the Driver, reviewing code and giving direction) [1]. Pair programming can have positive impacts on development time, in particular when the developers have previous experience with pair programming and when the project is of a larger size [2].

There are some drawbacks to pair programming, one large risk is that the navigator takes on a passive role, leaving it to the driver to carry the load. Another problem is that the Navigator, who gives direction to the developer, would need to be knowledgeable of the operations and infrastructure of the software environment, in order to best make suggestions. A key goal in a DevOps environment is to bridge the gap between developers and IT operations, however this could prove a challenge especially among larger organizations where there is a clearer division between software developers and IT operations.

A solution to this kind of problem is to introduce artificial intelligence into the mix. The role of the driver could be assumed by an AI agent, taking instructions from the human navigator to move the project in the right direction. The AI agent could also take on the role of Navigator, by understanding the operational context and offering code suggestions based on its understanding of the operational requirements and infrastructure. The most recent version of GitHub Copilot has adopted OpenAI's new GPT-4 model, and allows for voice and chat controlled code development, among other things [3]. This allows Copilot to take on the role of driver or navigator, and for the developer to take a step back and allow it to do the bulk of code writing and recommendations.

In this thesis we will give a general description of pair programming techniques in a DevOps setting, and evaluate the possibilities, strengths and weaknesses of using AI for these purposes.

2 Background

In this section the concepts of the essay will be briefly explained and put into context, including what pair programming is and its benefits, the role of pair programming in DevOps, and a state of the art analysis of relevant AI tools, with a special focus on GitHub Copilot X.

2.1 Pair programming

The idea of pair programming is simple: two programmers work on the same problem at the same time, on the same computer. It is not a new concept, but there are many misconceptions about the subject. For example, two people working the same problem is often expected to be twice as expensive [4], when the opposite may be true. The duo of programmers each take on the role of the driver (writing the code) and navigator (suggesting next course of action, strategical thinking) as seen in Figure 1, and frequently swap the roles.

Pair programming has been shown to improve results and speed up development cycles [2], and the positive effects are especially strong in pairs that have previously worked together and in projects with larger scopes.

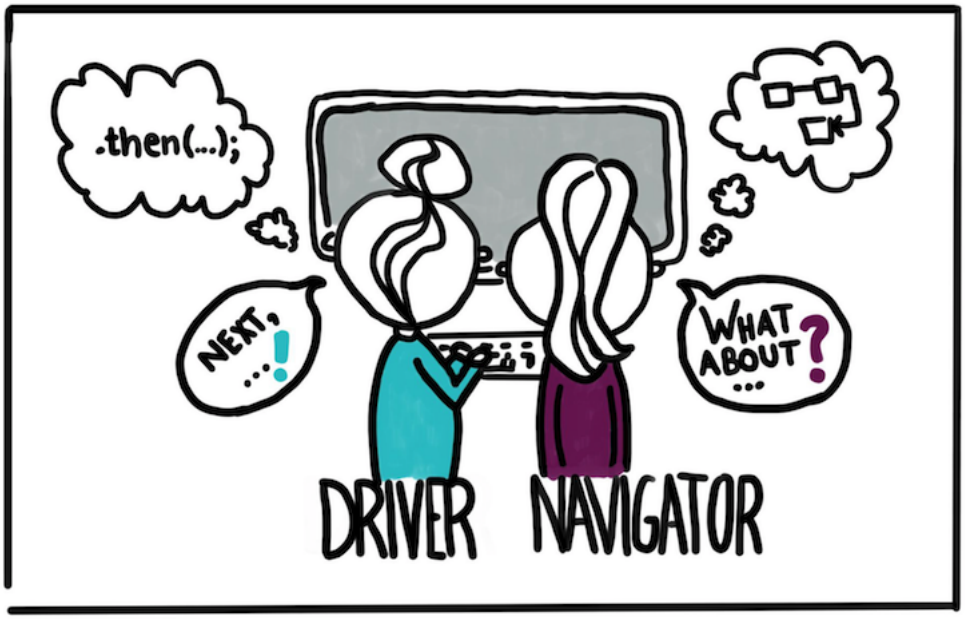

Figure 1: A programming pair consisting of a Driver and a Navigator

2.2 Relevance of pair programming in DevOps

DevOps is the art of collaboration, agile development and automation. Integrating developers of differing expertise into robust and well-rounded teams has been shown to increase productivity, reduce deployment times and minimize downtime [5]. Introducing pair programming into these teams is a natural next step, as this will introduce code review and discussion as soon as the driver starts coding. These discussions helps avoid a siloed developer spend time on a solution that was never going to work.

DevOps faces challenges at times, as everyone may not be on board with the best practices and tools. Pair programming aids in this regard as well, since the driver and navigator can help each other find solutions for the operational needs of the task.

2.3 GitHub Copilot X

GitHub Copilot is an artificial intelligence tool launched in 2021 (public launch in 2022) by GitHub and OpenAI [6]. At it's conception, it was an assistant that helped auto-complete code snippets using AI, but it has grown since. It is seen as an evolution of pair programming, where the intention is to achieve a similar effect but with a human-AI duo.

Copilot is a sort of descendant of the GPT language models that are developed by OpenAI, with the first version of Copilot using GPT-3. With the release of Copilot X, many new features have been unveiled, and many of them are significant improvements to the human-AI interaction pipeline. First off, it will use the new GPT-4 model, which simply put has significantly more parameters, meaning that it has the ability to learn more difficult and advanced patterns [7]. Secondly, Copilot is no longer just a rocket-powered auto-complete, but now supports chat functions, allowing the human to take on the role of navigator by explicitly telling Copilot what the code should do, or driver by having Copilot analyze the context of configuration files and providing appropriate recommendations[8] . Third, Copilot will also be able to help the developer with documentation, including languages, frameworks and other technologies [3].

2.4 Other AI code tools

Copilot is not the only pair programming AI on the market, but it is one of the most utilized and sophisticated ones. Other similar options include:

1. Codeium, which advertises its ability to avoid licensed code, something it claims Copilot struggles with [9].

2. Tabnine, which allows its users to train the model on only their own repositories, thus adapting a style similar to the user [10].

3. CodeGeeX, which is an open source alternative with the ability to generate multiple suggestions from a single prompt to let the developer choose the best one [11].

However Copilot is likely to remain the most widely used one, since it is integrated with the most advanced large language model.

3 Combining pair programming with AI

As was established earlier, pair programming is a welcome addition to DevOps, but there are several questions and considerations when one part of the pair is virtual. In this section we will discuss strengths and weaknesses of Copilot when utilized in a DevOps environment.

3.1 Strengths

For every step of the DevOps lifecycle collaboration is a key concept, and Copilot X would allow the used to have someone to collaborate with, even if no human part of the team is available. While it may not be an ideal tool for a novice, an experienced user of tools like Terraform or other DevOps toolkits can bring along Copilot for auto-completion and code suggestions.

One of the most important tools of a developer is searching. Searching for code, libraries, tools or any other part of the development process. After something suitable has been found, it then has to be shaped to fit the problem (or code) at hand, which can be time consuming and tedious. If using Copilot for pair programming, a lot of time can be saved at this step, since Copilot can suggest and implement suitable code blocks, suggestions and documentation.

GitHub Copilot X can be used to automatically review code changes and offer recommendations not only on technical issues, but also on proper coding conventions. This has been shown effective in automated review of pull requests as seen in Figure 2, where Copilot makes a suggestion for improved coding conventions. The ability to automatically review code changes could prove very useful in CI/CD pipelines, since these rely on extensive automation, including automated code testing [12]. In particular, Copilot could expand upon the existing method of automated testing – which typically test small pieces of code to ensure their correct functionality via unit tests – by creating dynamic coding convention tests. Copilot, integrated into a CI/CD pipeline, could help ensure that new code written fits into the coding standards of the code that has already been written to the repository. That is, instead of using a clearly defined rule set to ensure compliance with a coding convention, Copilot could learn dynamically from the code in the repository to ensure compliance with a coding convention that may be hard to define before the software project begins, and bring this functionality to an automated CI/CD pipeline.

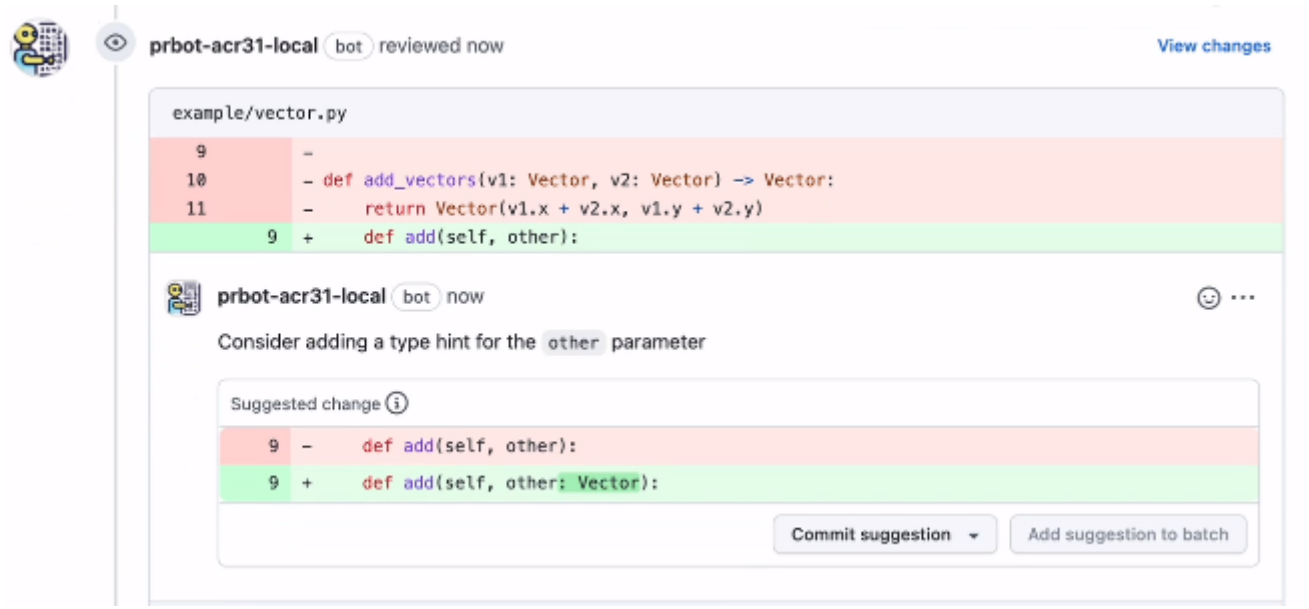

Figure 2: Copilot reviews a code change and suggests an improvement to follow proper coding conventions

3.2 Challenges

Because Copilot currently understands the context of e.g. infrastructure and operational requirements by analyzing files in the repository, a potential issue is that this is insufficient to properly understand the operational context and convey accurate recommendations or code for the developer. Because DevOps methodology includes creating configuration files, and writing infrastructure as code, these challenges become most notable in organizations where DevOps methodology has not been fully implemented. If developers work on software and have fully integrated DevOps methodology in their work practices, then files in the software repository may include information like which operating system the code will run on, and which version of libraries are used. This would allow Copilot to see this information and make recommendations or write code that takes the operational context in mind, by e.g. writing operating system specific code, and making sure methods called are not from later library versions. In contrast, if DevOps was not used and the software repository does not include infrastructure information, then Copilot would not know the best type of recommendation or code to make. A potential fix for this issue in organizations where DevOps may not be appropriate, could be to allow Copilot to read context not only from other files, but from a manual prompt input. Then the IT operations department could write a prompt specifying the infrastructure used, and Copilot could start each instruction it gets, by first looking at the prompt.

4 Conclusion

GitHub Copilot X is a viable alternative and supplement to traditional pair programming, but may prove to be insufficient if there is not enough infrastructure data available to it. To write code that is operational and infrastructure smart, Copilot should be used by developers in a DevOps environment, where it would have access to infrastructure information. As a result, if Copilot does shows promising results in speeding up development, it may push more organizations to a DevOps methodology. Because of it's ability to understand and review code automatically, Copilot has the potential to be integrated with a number of DevOps tools and technologies, such as CI/CD pipelines.

References

[1] B. Böckeler and N. Siessegger, *On Pair Programming*. [Online]. Available: https://martinfowler.com/articles/on-pair-programming.html (visited on Apr. 21, 2023).

[2] M. I. Aguirre-Urreta, G. Marakas, and W. Sun, "Effectiveness of Pair and Solo Programming Methods: A Survey and an Analytical Approach", 2015. [Online]. Available: https://www.semanticscholar.org/paper/Effectiveness-of-Pair-and-Solo-Programming-Methods%3A-Aguirre-Urreta-Marakas/d29d3b5db846f17ab757abdbfe9 (visited on Apr. 21, 2023).

[3] T. Dohmke, *GitHub Copilot X: The AI-powered developer experience*, en-US, Mar. 2023. [Online]. Available: https://github.blog/2023-03-22-github-copilot-x-the-ai-powered-developer-experience/ (visited on Apr. 21, 2023).

[4] J. Moore, *Advance Your DevOps with Pair Programming – Even Remotely*. [Online]. Available: https://thenewstack.io/advance-your-devops-with-pair-programming-even-remotely/ (visited on Apr. 21, 2023).

[5] Atlassian, *Benefits of DevOps*, en. [Online]. Available: https://www.atlassian.com/devops/what-is-devops/benefits-of-devops (visited on Apr. 21, 2023).

[6] D. Gershgorn, *GitHub and OpenAI launch a new AI tool that generates its own code*, en-US, Jun. 2021. [Online]. Available: https://www.theverge.com/2021/6/29/22555777/github-openai-ai-tool-autocomplete-code (visited on Apr. 21, 2023).

[7] OpenAI, "GPT-4 Technical Report", 2023, Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.2303.08774. [Online]. Available: https://arxiv.org/abs/2303.08774 (visited on Apr. 23, 2023).

[8] *About GitHub Copilot for Individuals*, en. [Online]. Available: https://ghdocs-prod.azurewebsites.net/en/copilot/overview-of-github-copilot/about-github-copilot-for-individuals (visited on Apr. 21, 2023).

[9] C. Team, *GitHub Copilot Emits GPL. Codeium Does Not. — Codeium*. [Online]. Available: https://codeium.com/blog/copilot-trains-on-gpl-codeium-does-not (visited on Apr. 23, 2023).

[10] T. Team, *Tabnine Enterprise vs. GitHub Copilot Business*. [Online]. Available: https://www.tabnine.com/blog/tabnine-enterprise-vs-github-copilot-business/ (visited on Apr. 23, 2023).

[11] codegeex, *CodeGeeX - A Multilingual Code Generation Tool - CodeGeeX*. [Online]. Available: https://codegeex.cn/ (visited on Apr. 23, 2023).

[12] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", *IEEE Access*, vol. PP, Mar. 2017. DOI: 10.1109/ACCESS.2017.2685629.