Taking Flight with Copilot

https://dl.acm.org/doi/pdf/10.1145/3582083

Brînză Alina-Elena, Apăvăloaei Alexandru-Teodor, Astăluș Adrian-Claudiu

At the present time, pair programming is a technique used in software development where two programmers work together. One of them, the *driver*, writes the code, while the other one, the *navigator*, reviews the code written by the *driver*, the two of them often switching roles. Despite requiring more person-hours in comparison to individual work, the resulting code is of higher quality. There are fewer defects, the code is more readable and the time it takes to write it is reduced. The emergence of AI and AI-powered tools in recent years has led to a new understanding of pair programming. GitHub's Copilot is one of the most relevant tools that are at the forefront of this shift. With Copilot, the AI takes the role of the *driver*, Copilot having the capability of generating lines of code much faster than a peer. Thus, the developer's sole role is the one of the *navigator*.

At its core, Copilot represents an IDE extension that generates code suggestions for the user. While many models offer text completion, either when writing documents or text messages, Copilot differentiates itself from these by being trained on source code (using Codex), rather than plain text. The way in which Copilot works is that the model recognizes similarities between the code written and code seen by Copilot in the training process, and offers new suggestions by synthesizing what it has learned during training. The role of this tool is to speed up the pair programming process and to improve the productivity of the developers. The experience developers encounter is similar to the existing autocompletion that appears in modern IDEs. Different from autocompletion, Copilot gives the developers longer suggestions, sometimes consisting of multiple code lines, these suggestions often being more relevant and helpful. Due to the fact that this tool is still in its early days, there are also a few downsides when it comes to its use. According to user reports, Copilot sometimes does not write 'defensive code'. This means that in some cases, Copilot does not check whether a pointer is null, or indices are negative. Moreover, the suggestions offered by the tool take into context only the current file, when many projects consist of more files that 'communicate' with one another. Besides this, one issue highlighted by users is that in some cases, the suggestions offered by Copilot have proved to be rather distracting.

In order to validate the tool, an in-depth case study has been conducted. This study gathered professional Python developers that had no previous interactions with Copilot. After a brief walkthrough of the tool's usage, these developers were asked to complete a series of tasks with the aid of Copilot. After completing the tasks, the developers were asked to review the tool, with the overall response being that Copilot is an efficient and useful tool that saves time. Nevertheless, some expressed their concerns, as this tool can lead to laziness and over-dependance. A core highlight of this study is that developers that use AI-assisted tools in their process offer increased attention to code reading and reviewing compared to code writing. In general, the majority of participants felt that Copilot contributed to their productivity.