

How much more productive does AI make developers?

27 August 2025, 14:16

Pascal Pollet

What AI really brings to software development: findings of a Stanford study

One of the most important applications of generative AI in software development is supporting programmers with writing code. Numerous specialist AI tools for coding now exist for this purpose, and some companies even claim that they will soon be able to replace some of their software developers with AI. But what can you realistically expect from AI for programmers?

Stanford University conducted a large-scale study of the impact of AI support on productivity in writing software code. What makes this study stand out is its methodology. Instead of using artificial testing, the researchers took real-world data from more than 600 companies that granted them access to their code repositories.

As a result, they gained insight into the work of over 100,000 developers and were able to analyse billions of lines of code. This allowed them to quantify the frequency of changes (commits) to the repositories, among other things. Productivity was measured not by the number of lines of code

written, but by the functionality of the delivered software.

What are the real benefits of AI for software development?

The study shows that **AI-assisted coding** initially increases output by around 30 to 40%. However, not all of this gain is permanent: the use of AI also introduces errors that programmers subsequently have to track down and correct. This extra work ultimately reduces the average gain to around 15 to 20%.

When does AI work and when doesn't it?

It is clear from the study that the impact of **AI in software development** varies significantly depending on several factors:

1. Complexity of the programming task

AI performs significantly better on simple programming tasks. With more complex tasks the gains are smaller, and in some cases the use of AI can even lead to a decrease in productivity.

2. Maturity of the codebase

AI works more efficiently with new (greenfield) projects than existing (brownfield) codebases. A few figures from the study:

- Simple greenfield tasks: 30–35% gain
- Complex greenfield tasks: 10–15% gain
- Simple brownfield tasks: 15–20% gain
- Complex brownfield tasks: 5–10% gain

3. Popularity of the programming language

AI's effectiveness also depends on the programming language used. For less popular languages such as Cobol, AI offers hardly any benefits, even with simple tasks. In fact, it can negatively impact productivity if a language has low popularity and high complexity.

However, for popular languages like **Python** and **Java** the gains are significant, because AI systems have been much better trained on the large amount of code available in these languages.

4. Size of the codebase

As codebase size increases, the benefits of AI diminish noticeably. This is relevant, as most modern codebases consist of several thousand lines of code or more.

Part of the reason for this is that the performance of **Large Language Models (LLMs)** for coding tasks decreases as the context length increases. Context length is the maximum volume of information (expressed in tokens) that a language model can process simultaneously, both from the input prompt and from the generated output.

AI as a powerful, but not magical solution for software development

There can be no doubt that generative AI undeniably makes it possible to significantly increase the performance of software developers. There are some caveats, however. **Productivity increases** aren't a new phenomenon in the software world. In recent decades, even greater productivity gains have often been achieved by switching to newer programming languages. For example, scripting languages like Python offer a productivity advantage of some 30% to 70% over procedural languages like Fortran or C++. Compared to Assembly, these modern languages are at least five times faster in terms of development time.

As we've seen, generative AI is a powerful accelerator for software development, but not a universal recipe for greater efficiency. The context from the programming language to the nature of the task is the key factor in determining success. It's therefore crucial for organisations to **integrate AI intelligently into development practices**, rather than expecting to see 'magic' leaps in performance.

Sources

- Https://www.youtube.com/watch?v=tbDDYKRFjhk
- Softwareengineeringproductivity.stanford.edu

Authors



Pascal Pollet