3 Steps to adopt AI for coding

Among 30 million software developers around the world, the level of interest in AI-assisted development remains unclear. The decision-making process on the adoption of AI tools is surrounded by uncertainty, posing questions about preferences and needs. Deep dive into this context to learn about adoption strategies and understand how to test use cases of AI-assisted software development.

Step 1
Choose your strategy

Accelerate: Have the code in mind and need to start writing ideas? Use small prompts to ask the AI assistant to avoid spending time on lengthy answers.

Explorer: If you have insights about the code, let the AI assistant pilot take control and explore possibilities. Be mindful of potential extra time spent analyzing multiple AI suggestions.

Step 2
Follow VEC technically

Validity: Seek AI-generated code with minimal syntax errors (e.g., GitHub Copilot delivers 91.5% error-free code).

Efficiency: Consider computational complexity to evaluate how well the AI answer aligns with canonical code.

Correctness: Assess how effectively the AI-generated code solves the problem (around 21% of AI-generated unit tests were compilable in a recent study).
For more details on VEC, refer to the research on assessing GitHub Copilot's code generation quality.

Step 3
Focus on Well-being

Individuals: Optimize coding time and reduce reliance on documentation and coworker assistance.

Team & Companies: Avoid using personal experiences to accelerate adoption initiatives. Validated benefits indicate AI's power to reduce cognitive load in coding experiences only.

Talk to our team
for more details

STACKSPOT by ZUP

Sources: 1) Grounded Copilot: How Programmers Interact with Code-Generating Models. **2)** Assessing the quality of GitHub copilot’s code generation. **3)** An Empirical Study of Using Large Language Models for Unit Test Generation. **4)** Large Language Models for Software Engineering Survey and Open Problems.