Paper Metadata
Pages: 23 • Venue: null • Citations: null • Access: null • Fields: None

Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools

Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk | (2022)

1. Executive Summary

This empirical study examines early developer experiences with GitHub Copilot, finding it shifts programming from writing to reviewing code, increases perceived productivity, and reveals new challenges around trust and code provenance.

2. Purpose & Research Question

- The purpose was to investigate developers' initial experiences with GitHub Copilot during its technical preview to understand how AI pair programming tools are used, their challenges, and their impact on development workflows. *(Explicitly Stated)*
- The rationale was that Copilot represents a significant shift in programming tools, moving from synchronous human pairing to AI assistance, yet actual user experiences beyond public hype were largely unknown and needed empirical investigation. *(Explicitly Stated)*
- RQ: How do developers use and experience GitHub Copilot in practice, what challenges do they encounter, and how does it impact their productivity and workflow? *(Inferred)*

3. Theoretical Framework

- No explicit theoretical framework was mentioned, though the analysis implicitly draws on concepts from human-computer interaction and software engineering practices around pair programming and tool adoption. *(Inferred)*

4. Methodology

- The research employed a multi-method approach combining qualitative and quantitative methods: forum analysis, a case study with professional developers, and a large-scale survey. *(Explicitly Stated)*
- Key techniques included analyzing 279 forum posts for themes, conducting structured interviews with screen sharing during coding tasks, and administering surveys with rating-scale questions about Copilot usage and perceived productivity. *(Explicitly Stated)*
- The sample included early Copilot technical preview users: forum contributors, five professional Python developers with no prior Copilot experience, and 2,047 survey respondents from 17,420

contacted users. *(Explicitly Stated)*

• Analysis methods included thematic analysis of qualitative data from forums and interviews, and statistical correlation analysis between survey responses and usage metrics like acceptance rate and persistence. *(Explicitly Stated)*

5. Major Findings & Contributions

• Finding 1: Developers using Copilot shift from writing code to reviewing AI-suggested code, spending more time understanding and assessing suggestions than generating code themselves. *(Explicitly Stated)*

• Finding 2: Acceptance rate of Copilot suggestions showed the highest positive correlation with perceived productivity (ρ=0.24), indicating users feel more productive when accepting suggestions, even if editing is required. *(Explicitly Stated)*

• Contribution: Provides first empirical evidence of how AI pair programming tools are used in practice, revealing new workflow patterns, productivity impacts, and emerging challenges around code quality, trust, and legal concerns that shape future tool design and research. *(Explicitly Stated)*

6. Study Limitations & Gaps

• Limitations include focus on early adopters during technical preview, small case study sample (n=5), and self-reported productivity measures rather than objective performance metrics. *(Inferred)*

• Gaps remain in understanding long-term effects on code quality, security implications, developer skill evolution, and how to effectively track AI-generated code provenance throughout the software development lifecycle. *(Explicitly Stated)*

7. Study Implications

• For Research: Future work should investigate developer trust in AI tools, code provenance tracking, security implications, and how AI assistance changes required developer skills and team collaboration dynamics. *(Explicitly Stated)*

• For Practice/Policy: Organizations should develop guidelines for AI tool usage, address legal and copyright concerns around generated code, and invest in training developers for code review and assessment skills rather than just code writing. *(Explicitly Stated)*