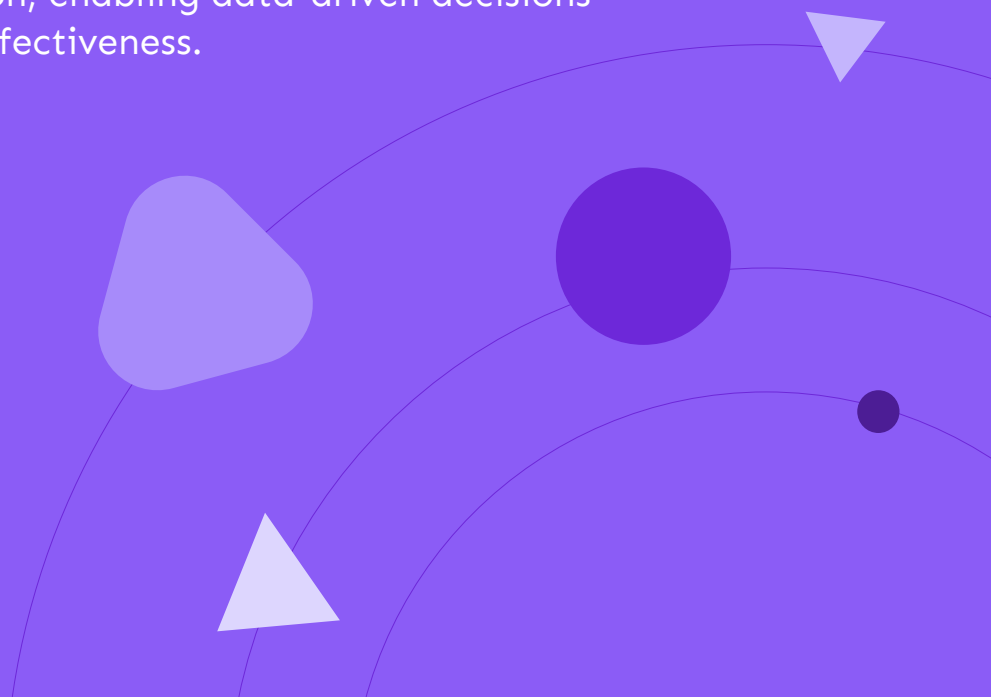
Allstacks

WHITEPAPER

# Using Allstacks to Measure the Impact of Github Copilot AI on Software Development

Allstacks has proudly introduced a groundbreaking analytics suite that reveals the true business impact of GitHub Copilot on your development teams. Our intuitive dashboard delivers the metrics that matter, showing leaders exactly how AI is accelerating delivery while maintaining quality.

The results so far are exciting! Teams are achieving remarkable efficiency gains, and our metrics provide a clear roadmap for optimizing AI integration into your existing workflows. We've created a powerful analytical foundation that takes the guesswork out of AI adoption, enabling data-driven decisions that enhance your team's effectiveness.



# Summary

The integration of AI tools in software development promises significant productivity gains, yet measuring their actual impact has remained challenging. Based on composite Allstacks customer data, we've conducted analysis of engineering teams using GitHub Copilot, revealing nuanced patterns in development efficiency and code quality.

Our research shows that while some teams achieve anywhere from 20% up to 40% faster delivery times with AI assistance, success depends heavily on implementation strategy and measurement frameworks. Through analyzing multiple engineering metrics—from cycle time to code quality—we've identified key factors that determine AI tool effectiveness in development environments.

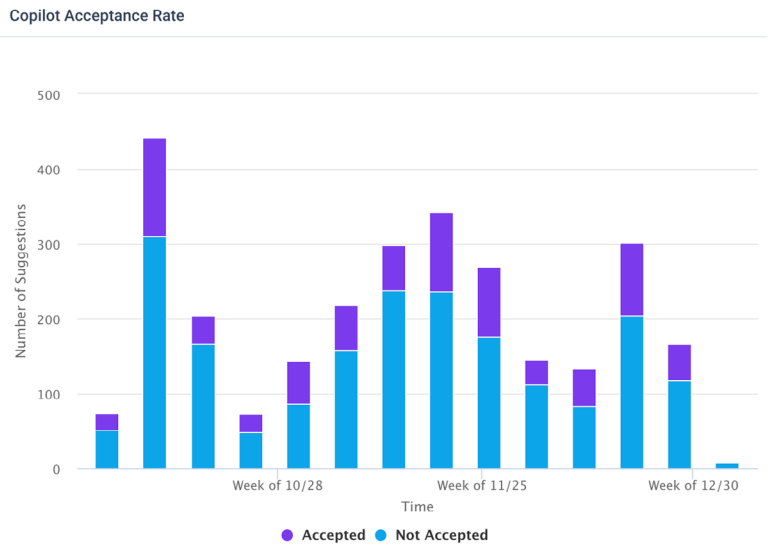
This paper explores:

- Quantifiable impacts on development velocity and code quality
- Critical success factors for AI tool integration
- Measurement frameworks for evaluating AI assistance
- Best practices for maximizing return on AI investment

Using real-world data from development teams, we'll examine how organizations can effectively measure and optimize their AI-assisted development processes while maintaining high quality standards.

SECTION ONE

# Acceptance vs. Rejection of Suggestions




## Overview

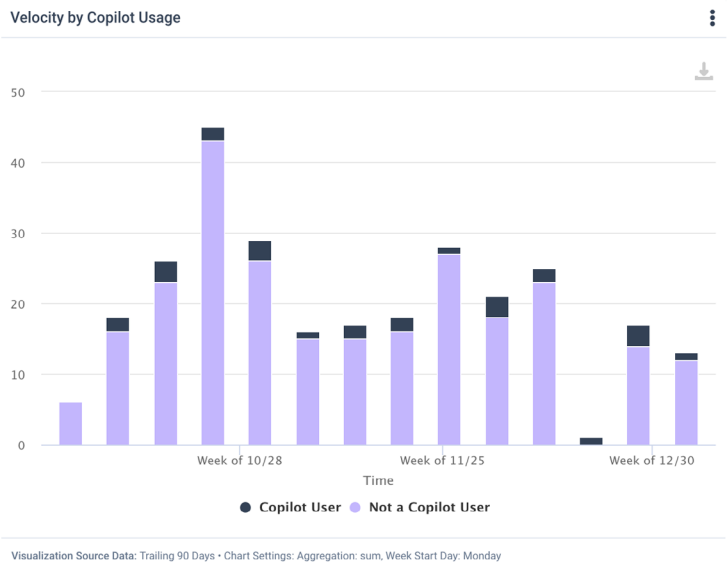
This chart provides interesting insights about how developers are interacting with Copilot's suggestions. Copilot acceptance patterns indicate developers are selectively using AI suggestions, with a consistent 25-30% acceptance rate. This measured approach shows developers are thoughtfully evaluating Copilot's output rather than blindly accepting suggestions. The steady acceptance ratio indicates developers have established reliable criteria for determining useful AI contributions.

## Key Findings

- Week-to-week suggestion volume varies substantially, suggesting Copilot's utility fluctuates based on development task types. In our analysis, maintenance of software (lower) and the build out of new software (higher) have different acceptance rates due to contextual dependencies
- Stable acceptance ratio across varying volumes indicates developers have developed reliable criteria for evaluating AI suggestions
- Developers are using Copilot as a complementary suggestion tool rather than a replacement, showing appropriate skepticism while still leveraging its benefits
- The data suggests successful integration of AI assistance while maintaining human oversight and judgment in the development process

SECTION TWO

# Impact of Copilot on Time to Complete




## Overview

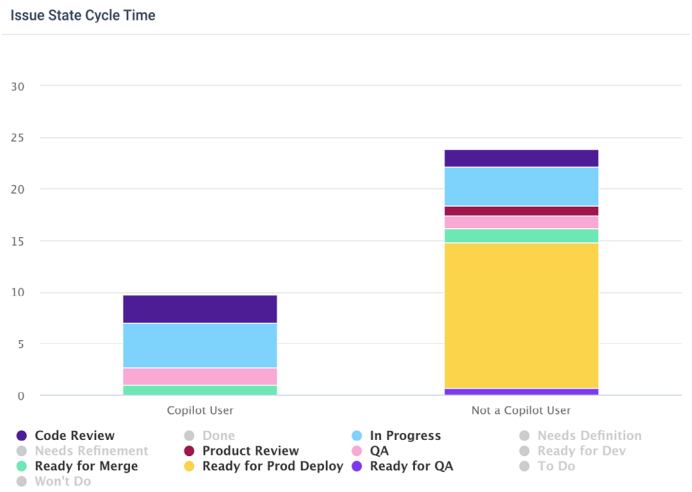
Analysis of development completion times shows Copilot delivers consistent efficiency gains, with the most significant impact during high-workload periods. While not revolutionizing development processes, it provides reliable optimization, particularly for routine and boilerplate tasks.

## Key Findings

- Teams using Copilot can demonstrate 30-40% faster completion times across development tasks
- Benefits are most pronounced during high-workload periods, suggesting particular value for intensive development cycles
- Efficiency gains appear consistent and predictable, enabling better sprint planning and resource allocation
- While Copilot optimizes routine development work, it maintains the need for developer oversight on complex tasks, indicating it's best used as an enhancement to existing processes rather than a replacement

SECTION THREE

# Issue Cycle Time




## Overview

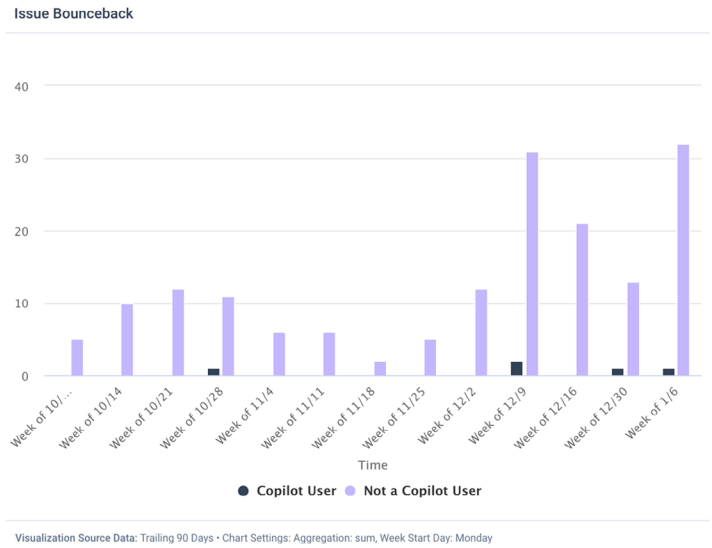
Analysis of Issue State Cycle Time reveals significant differences between Copilot and non-Copilot users across various development stages. The data shows AI assistance substantially reduces overall cycle time, with notable variations in different phases of the development process.

## Key Findings

- In this example, Copilot users achieved faster overall cycle times (10 units vs 24 units) compared to non-Copilot users
- The most significant time savings appear in the “Ready for Prod Deploy” phase, indicating faster preparation for production releases
- AI assistance impacts all development stages, from code review through QA, suggesting comprehensive workflow benefits
- While Copilot reduces cycle time, critical phases like code review and QA maintain appropriate durations, indicating no compromise in quality control
- The data helps identify specific development phases where process improvements could further optimize AI-assisted development, such as code reviews and production readiness processes

SECTION FOUR

# Issue Bounceback




## Overview

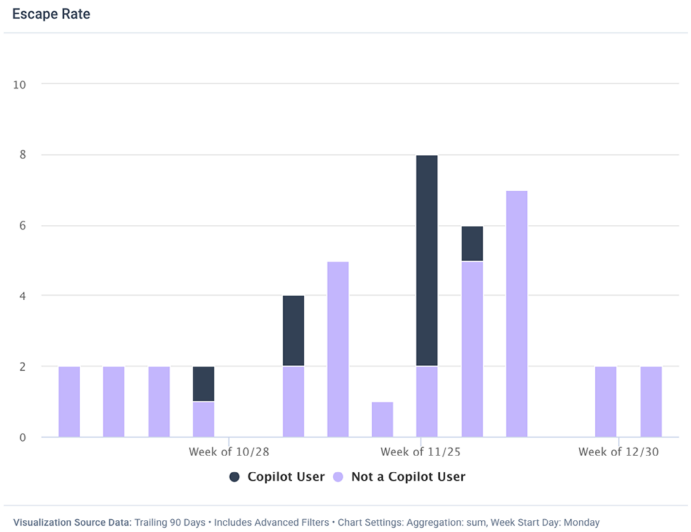
Analysis of code bounceback rates demonstrates Copilot's significant impact on initial code quality and development efficiency. The data suggests that Copilot may be providing significant value in reducing initial code defects and improving first-pass quality, which could lead to more efficient development processes.

## Key Findings

- Copilot users maintain consistently lower bounceback rates when compared to non-Copilot users
- AI-assisted development shows remarkable stability in code quality, with minimal fluctuation in bounceback rates even during high-stress periods
- The consistency of low bouncebacks for Copilot users suggests this isn't just a random improvement
- Consistently low bouncebacks across time periods suggest Copilot is making contextually appropriate and reliable code suggestions before moving to the next lifecycle stage

SECTION FIVE

# Escape Rate for Bugs




## Overview

This chart shows a critical finding and observation when using AI Assist tooling. This analysis highlights the need for specialized testing and review processes for AI-generated code to maintain quality standards throughout the development lifecycle.

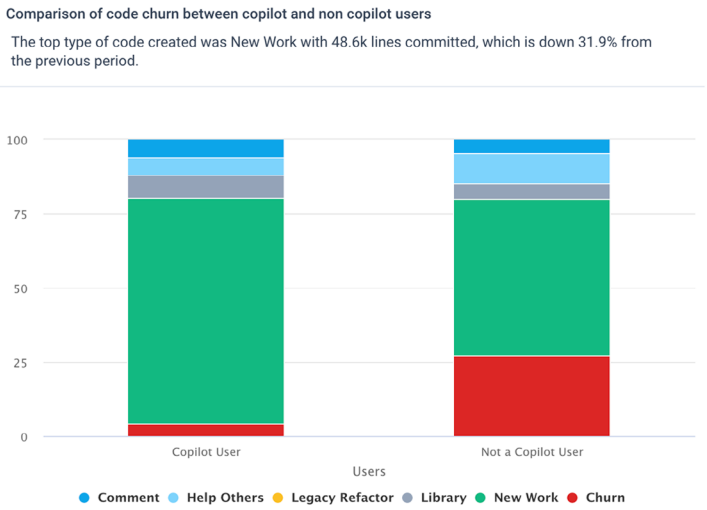
While Copilot demonstrated strong performance in initial code quality, data reveals important considerations regarding production bug escape rates.

## Key Findings

- When it comes to overall product quality, there are no shortcuts. It is vital to remain vigilant with regard to all review and testing processes!
- It is important to remember that upon acceptance of any AI suggestion, the code ownership stays with the team and individuals
- Copilot users experience variable escape rates, which may indicate potential over-reliance on AI suggestions without sufficient critical review
- Teams need to maintain rigorous QA practices despite AI assistance, with particular attention to edge cases and integration testing
- Success with AI tools requires balancing increased development speed with appropriate quality controls and testing procedures

SECTION SIX

# Code Churn




## Overview

A comparison of code churn between Copilot and non-Copilot users reveals significant differences in code stability and development patterns. The data shows how AI assistance affects various types of code changes, from new work to maintenance activities.

## Key Findings

- Non-Copilot users show significantly higher code churn rates (red section ~25%) compared to Copilot users (~5%), suggesting AI assistance leads to more stable, purposeful code changes
- Both groups maintain similar levels of new work (green section ~55%), indicating Copilot doesn't fundamentally change development output but rather improves its efficiency
- Copilot users demonstrate a more balanced distribution across different coding activities, including comments, help documentation, and library work
- The lower churn rate for Copilot users suggests AI assistance may help developers write more accurate code the first time, reducing the need for frequent revisions
- This suggests Copilot helps create more stable, maintainable code while reducing unnecessary code churn and revisions.

CONCLUSION

# The Allstacks View

Allstack's analysis shows GitHub Copilot delivers compelling business value through smart integration with development workflows. Teams leveraging our metrics-driven approach are seeing impressive gains: 20-40% faster completion times, reduced code churn, and higher first-pass quality. By establishing clear baselines and monitoring key performance indicators, organizations can confidently accelerate development while maintaining quality standards.

The data is clear—when properly implemented with the right metrics and quality controls, AI-assisted development creates a powerful competitive advantage. Teams deliver more value faster, while maintaining the high standards your business demands. As we look ahead, the combination of AI assistance and data-driven oversight promises to transform software development efficiency.

And that's it... no spreadsheets, late nights, or frustration allowed.

**Ready to get started? ↓**

## Book a demo today or try Allstacks for free.