



Measuring the Impact of AI Development Tools



Table of Contents

Preface	4
About the Authors	4
Executive Summary	5
1. Introduction	6
The Rise of AI in Software Development	7
Defining AI Coding Assistant Tools	8
2. The Potential Benefits of AI Coding Assistant Tools	9
Enhancing Productivity Through Automation	10
Improving Code Quality and Consistency	11
Streamlined Workflows and Reduced Toil	12
3. The Importance of Measuring the Impact of AI Coding Assistant Tools	13
4. Key Metrics for Measuring Impact	14
Developer Velocity	15
Product Quality	16
Developer Experience	17
How Harness AI Productivity Insights Can Help	19

Table of Contents

5. Methodology for Measurement	20
Controlled Experiments	21
Surveys and Interviews	22
Data Analysis	23
Key Measurement Capabilities	24
6. Challenges and Considerations	25
Defining Clear Measurement Goals	26
Ensuring Data Integrity	27
External Factors	28
7. Balancing Quantitative and Qualitative Data	29
Contextualizing Metrics	29
Crafting Insightful Questions	30
8. The Future of AI in Software Development	31
Emerging Innovations and Trends	32
Societal and Economic Implications	33
Recommended Next Steps	35
How Harness Can Help You	37

Preface

This ebook is a collaboration between Google, the steward of DORA research, and Harness, the AI-native software delivery platform. It combines Google's expertise in performance measurement with Harness's ability to turn development data into insights, offering a practical guide to navigating AI-augmented software development.

Readers will learn which metrics matter and how to collect, interpret, and apply them effectively. Whether you're an engineering leader, DevOps practitioner, or developer, this guide provides research-backed frameworks and actionable strategies to measure AI's impact on productivity, quality, and developer experience.

Turn the page to begin your journey toward a more data-driven future in software delivery.

About the Authors



Nathen Harvey

Google Cloud

Nathen Harvey leads the DORA team at Google Cloud, helping organizations improve software delivery using DORA research. He focuses on enhancing developer experience and performance, supports technical communities like the DORA Community of Practice, and has co-authored key DORA reports and contributed to 97 Things Every Cloud Engineer Should Know.



Prashant Batra

 **harness**

Prashant Batra, Senior Director of Product Management at Harness, brings over 20 years of experience in building AI-driven tools that turn engineering data into predictive insights. He focuses on enhancing visibility into software workflows and bridging technical metrics with strategic decisions.

Executive Summary

- AI coding assistants have become transformative tools in software development, with 81% of organizations increasing AI incorporation into applications, primarily for code writing (74.9%) and information summarization (71.2%)
- The key benefits of AI coding tools include enhanced developer productivity through automation, improved code quality and consistency, and streamlined workflows that reduce manual toil—though developers still need to evaluate and modify AI-generated code.
- Measuring AI's impact requires tracking quantitative metrics (development velocity, code quality, deployment frequency) and qualitative factors (developer satisfaction, cognitive load reduction) through controlled experiments, surveys, and data analysis.
- Primary challenges in measuring AI impact include maintaining consistent measurement approaches across different development contexts, and ensuring data integrity, accounting for external factors like team changes and project complexity.
- Organizations need comprehensive measurement frameworks that can adapt to evolving AI capabilities while balancing technical metrics with human factors. Tools like Harness AI Productivity Insights combine development tool metrics and developer surveys to provide a more complete understanding of the impacts of AI investments and how to optimize their value.
- The future of AI in software development points toward increasingly sophisticated tools with enhanced contextual awareness, though success requires thoughtful integration that considers both productivity gains and broader societal impacts on the workforce.

1 . Introduction

There is no question that artificial intelligence is rapidly reshaping software development. Measuring its real impact has become vital for organizations investing in AI-powered developer tools. This ebook delves into the methodologies, metrics, and frameworks needed to accurately evaluate how AI influences developer productivity, code quality, and business outcomes.

As companies increasingly adopt AI assistants for tasks like code generation, testing, and deployment, understanding their tangible benefits is essential. Justifying these investments and optimizing implementation strategies require a thorough analysis of AI's real-world impact. However, measuring AI's influence poses unique challenges beyond traditional productivity metrics. From quantifying time savings and code quality improvements to assessing developer experience and maintenance implications, organizations need comprehensive strategies to capture AI's true value.

“ By 2027, the number of platform engineering teams using AI to augment every phase of the software development life cycle (SDLC) will have increased from 5% to 40%.

Gartner®, Research Roundup for DevOps, 20 January 2025

This guide offers practical frameworks for evaluating AI tools throughout the software development lifecycle. Drawing on real-world case studies, expert insights, and proven methodologies, readers will discover how to establish meaningful metrics, collect accurate data, and assess AI's impact on both technical and business outcomes.

Whether your organization is just starting to implement AI tools or looking to enhance existing deployments, this ebook provides essential guidance for measuring and maximizing the benefits of AI in software development.

The Rise of AI in Software Development

The integration of artificial intelligence into software development marks a transformative shift in how we build, test, and deploy code. From startups to enterprise organizations, AI tools are fundamentally changing development practices and team dynamics. This evolution represents more than just technological advancement—it signals a new era in software engineering where AI serves as an intelligent collaborator throughout the development lifecycle.

Recent years have seen exponential growth in AI adoption across development teams. What began as simple code completion tools has evolved into sophisticated AI assistants capable of generating complex functions, optimizing performance, and predicting potential issues before they arise. These tools now support developers across the entire software development lifecycle, from initial planning through deployment and maintenance.

The impact extends beyond mere productivity gains. AI tools are influencing architectural decisions, changing how teams approach testing and quality assurance, and reshaping code review processes. They're also democratizing development by making advanced programming capabilities more accessible to developers across skill levels.

Yet this rapid adoption brings new challenges. Organizations must navigate questions about code quality, security implications, and the changing role of human developers. The need to measure and understand AI's impact has become crucial as teams seek to optimize their use of these tools while maintaining code quality and security standards.

This shift toward AI-augmented development represents a crucial inflection point in software engineering. As these tools continue to evolve, their influence on development practices, team structures, and skill requirements will only grow. Understanding and measuring this impact becomes essential for organizations looking to harness AI's potential while managing its challenges effectively.



Defining AI Coding Assistant Tools

The software development landscape is undergoing a paradigm shift with the rise of AI coding assistants like Google Gemini Code Assist, GitHub Copilot, Cursor, or Windsurf. AI coding assistant tools represent a new generation of development solutions that leverage artificial intelligence to enhance the software development process. These tools range from simple code completion engines to sophisticated systems capable of generating entire functions, optimizing performance, and detecting potential issues before they manifest.

At their core, AI coding assistants combine advanced machine learning models trained on vast code repositories with natural language processing (NLP) capabilities. This powerful combination allows them to interpret developer intent, provide contextually relevant code suggestions, and even transform natural language descriptions into fully functional code. Some of the key capabilities of modern AI coding assistants include:

- Intelligent code completion and generation
- Automated testing suggestion and creation
- Bug detection and prevention
- Code refactoring recommendations
- Natural language to code translation
- Documentation generation
- Performance optimization suggestions

While early code assistants focused primarily on autocomplete functionality, today's AI tools serve as collaborative partners throughout the development process. They can understand project context, maintain consistency with existing codebases, and adapt to individual coding styles and preferences.

The evolution of AI coding assistants has profound implications for how we measure developer productivity and code quality. Traditional metrics, which were designed for a pre-AI era, must be reimaged to reflect the transformative impact of these tools on the way developers write, test, and maintain code. To fully harness the potential of AI coding assistants, it is critical to understand their capabilities, limitations, and role in shaping the future of software development.

2. The Potential Benefits of AI Coding Assistant Tools

AI coding assistants are rapidly changing the software development landscape, particularly in the developer's inner loop, where most of the day-to-day coding and development work happens. These tools offer real-time suggestions, automate repetitive tasks, and provide contextual guidance, effectively streamlining the coding process. By integrating directly into the developer's Integrated Development Environment (IDE) or through other interfaces such as chatbots or external web interfaces, AI coding assistants are becoming indispensable for many software professionals. This new approach increases efficiency, reduces the time spent on mundane tasks, and allows developers to concentrate on more complex and creative problem-solving. The potential benefits span various areas of the software development lifecycle, with early adopters already reporting significant positive impacts on productivity and code quality.

This section explores the potential advantages of integrating AI coding assistants into development workflows, specifically focusing on three key areas: enhancing productivity through automation, improving code quality and consistency, and streamlining workflows and reducing toil. These areas are where AI's impact can be most immediately felt by developers. By understanding these benefits, organizations and individual developers can make informed decisions about the adoption and integration of these powerful tools.



Enhancing Productivity Through Automation

One of the most significant benefits of AI coding assistants is their ability to boost developer productivity through automation. AI can perform a range of tasks that are often time-consuming and repetitive, such as writing code, summarizing information, and writing tests.



agree increasing productivity is the biggest benefit that developers identify for AI tools.

By automating these tasks, developers can dedicate more time to higher-level problem-solving and innovation, which leads to significant gains in efficiency. AI coding assistants can synthesize information and provide highly personalized responses, which saves time spent on context switching. This can lead to a significant increase in individual productivity.

Furthermore, AI is helping developers complete their work more efficiently, allowing them to realize value from their work more quickly. This means that developers can finish their tasks faster and with more focus. This increase in efficiency contributes to improved job satisfaction as developers are able to see their work come to fruition more quickly.

According to the [2024 DORA Accelerate State of DevOps Report](#), a majority of developers report positive productivity gains from using AI tools, with over one third reporting moderate or extreme increases in productivity. This shows a widespread benefit from adopting these technologies, suggesting that the potential for productivity gains is significant across the software development industry.

Improving Code Quality and Consistency

AI coding assistants also play a crucial role in improving code quality and consistency. By analyzing code in real time, they can identify potential errors, suggest optimizations, and ensure that code adheres to established coding standards and best practices. AI can assist with tasks such as debugging code, optimizing code, and writing documentation. This not only reduces the likelihood of bugs and vulnerabilities but also contributes to a more maintainable and robust codebase. This improved code quality can lead to reduced code complexity, and improved documentation. These factors together can lead to an overall improved codebase. AI can also help with code review and approval processes leading to faster reviews and approvals.

The consistency that AI brings to code is also invaluable, ensuring that developers across a team follow the same patterns and conventions. This minimizes the variability in coding styles that can make code harder to understand and maintain. While many developers may not completely trust the code generated by AI, they still find the code useful and are willing to tweak it. This indicates a trend toward accepting AI as a tool that enhances developer capabilities rather than something that replaces the human element of writing good code. In fact, developers are becoming used to using AI-generated code and comparing it to early days of using StackOverflow where developers would copy and paste code and have to debug the code to get it to work.

“ One participant even likened the need to evaluate and modify the outputs of AI-generated code to “the early days of StackOverflow, [when] you always thought people on StackOverflow are really experienced, you know, that they will know exactly what to do. And then, you just copy and paste the stuff, and things explode

DORA Accelerate State of DevOps Report, p. 24

Streamlined Workflows and Reduced Toil

AI coding assistants are instrumental in streamlining workflows and reducing toil for developers, freeing up time from manual, less-valuable tasks. By automating tasks like documentation and test writing, AI enables developers to focus on tasks that directly impact product value and innovation. The potential for AI to enhance flow and reduce interruptions can have a positive impact on job satisfaction. The reduction of toil can also prevent burnout and make developers more engaged with their work.

AI's assistance with code reviews and approvals, resulting in faster turnarounds, means that developers spend less time waiting for feedback and can more quickly iterate on their work. By integrating with CI/CD pipelines, AI can also automate many deployment processes, further reducing the manual overhead that developers need to manage. By using AI to its full potential, the developers can spend more time on creative, valuable, and fulfilling work, leading to a more efficient and happier development team.



3 . The Importance of Measuring the Impact of AI Coding Assistant Tools

Understanding and quantifying the impact of AI coding assistants has become crucial as organizations increasingly invest in these technologies. Accurate measurement enables informed decision-making about tool adoption, helps optimize implementation strategies, and justifies continued investment in AI development solutions.

Key areas requiring measurement include productivity gains, code quality improvements, and business value generation. Organizations need to track metrics like development velocity, defect rates, and time-to-market while also considering qualitative factors such as developer satisfaction and code maintainability.

Effective measurement frameworks must account for both direct and indirect benefits. While some impacts, like time saved on code generation, are easily quantifiable, others, such as improved developer experience or reduced cognitive load, require more nuanced evaluation approaches.

Organizations should focus on tracking:

- **Development velocity and time savings**
- **Code quality metrics and defect rates**
- **Developer productivity and satisfaction**
- **Business impact and ROI**
- **Resource utilization and efficiency**
- **Knowledge sharing and team collaboration**
- **Security and compliance improvements**

Establishing baseline measurements before implementing AI tools enables organizations to more accurately assess their impact. Regular monitoring and adjustment of measurement frameworks ensure continued alignment with organizational goals and evolving AI capabilities.

Success requires balancing quantitative metrics with qualitative assessments to build a comprehensive understanding of AI's impact across the software development lifecycle.

4 . Key Metrics for Measuring Impact

Measuring the impact of AI coding assistants requires a comprehensive framework encompassing both quantitative and qualitative metrics across the software development lifecycle.



“Implement a measurement framework that evaluates AI not by sheer adoption but by meaningful downstream impacts — how it helps employees thrive, benefits those who rely on your products, and unlocks team potential.”

[DORA Accelerate State of DevOps Report, p46](#)

Developer Velocity

Development velocity stands as a primary metric, tracking how AI tools affect code generation speed, task completion times, and overall project delivery rates. Organizations must establish clear baselines and monitoring systems to accurately measure changes in development pace. Time allocation tracking becomes crucial, focusing on how developers spend time between creative problem-solving and routine tasks. AI assistants typically reduce time spent on boilerplate code and repetitive operations, allowing more focus on complex architectural decisions and innovation.

Throughput Metrics

Throughput metrics focus on the volume of work completed over a given period.

These include:

- **Deployment Frequency:** Measures how often code is successfully deployed to production. Frequent deployments indicate streamlined workflows and efficient development cycles.
- **Number of Change Sets:** Tracks the number of code changes per deployment. Smaller, more frequent commits support iterative development, simplifies code review, and reduces merge conflicts
- **Number of Features Delivered:** Assesses the rate of feature implementation, highlighting the ability of teams to meet product milestones and respond to customer needs.

Efficiency Metrics

Efficiency metrics evaluate the speed and effectiveness of development processes:

- **Cycle Time:** Captures the time taken from the start of a task to its completion. AI coding assistants can reduce cycle times by automating repetitive steps and providing quick solutions.
- **Lead Time for Changes:** Measures the duration from code creation to deployment. Shorter lead times reflect improved coordination and reduced bottlenecks.
- **Time to Resolution:** Tracks the speed of resolving issues such as bugs and errors. AI tools excel at identifying problems early, enabling faster resolutions.

Product Quality

Quality measurement demands attention to multiple dimensions, including defect rates, debugging time, and review efficiency. AI tools often influence these metrics through automated checks and suggestions, making it crucial to track both immediate and long-term quality impacts. Security vulnerability detection and technical debt accumulation provide additional insights into code quality trends.

These quality metrics must be measured against pre-AI baselines to accurately quantify improvements. Regular assessment ensures AI tools continue to enhance product quality while meeting evolving standards and requirements.

Quality Metrics

These metrics evaluate the robustness and reliability of the code:

- **Defect Rates:** Indicates the frequency of errors or bugs found in production. Lower defect rates signify enhanced code quality facilitated by AI-driven suggestions.
- **Customer Satisfaction Scores:** Surveys and user feedback reflect the perceived quality and reliability of software delivered with AI-assisted tools.
- **Adherence to Performance Benchmarks:** Compares code performance against predefined standards. This metric highlights the role of AI in optimizing performance metrics like execution speed and resource efficiency.



Developer Experience

Developer experience with AI coding assistants can be measured through sentiment and satisfaction surveys, workflow efficiency, and cognitive load reduction. Overall satisfaction metrics encompass tool reliability, suggestion relevance, and workflow integration quality. Regular feedback collection helps optimize AI tool implementation and ensures continued alignment with developer needs.

Mental load reduction shows in decreased time spent on routine tasks, improved problem-solving capacity, and reduced friction in development workflows. Tool adoption patterns and feature utilization rates indicate how effectively developers integrate AI assistance into their daily work.

Satisfaction Metrics and Sentiments

Focusing on the human aspect of development:

- **Developer Surveys:** Regular surveys capture satisfaction levels and perceptions of AI tools' utility and ease of use.
- **Free form feedback:** Present opportunities for identification of hidden perceptions and sentiments.

Behavioral and Health Metrics

Behavioral data provides deeper insights into team dynamics and individual well-being:

- **Burnout Rates:** Tracks indicators of overwork or stress among developers. Effective AI tools should reduce toil and alleviate repetitive burdens.
- **Retention Rates:** Measures the ability to retain top talent, reflecting the positive impact of AI on job satisfaction.

Adoption and Usage Metrics

Adoption and usage metrics reveal how effectively teams integrate AI tools into their workflows, providing crucial insights into the real-world impact of these technologies. By tracking these patterns, organizations can identify both successful adoption strategies and potential barriers to effective implementation. Understanding both adoption and ongoing usage metrics helps organizations optimize their AI implementation strategies and ensure sustained value from their investments.



Adoption Metrics

Initial adoption metrics focus on how quickly and thoroughly teams begin incorporating AI assistants into their development processes:

- **Tool Utilization Rate:** Frequency and duration of AI assistant usage across different development tasks and team members
- **Initial Feature Coverage:** Which basic capabilities do teams leverage during their first months of implementation
- **Onboarding Success:** Rate at which new users successfully integrate the tools into their workflows

Usage Metrics

Long-term usage metrics reveal the depth and sustainability of AI tool integration in development workflows:

- **Suggestion Acceptance Rate:** Percentage of AI suggestions that developers accept and integrate into their code
- **Production Success Rate:** Percentage of AI-suggested code that survives review and makes it to production
- **Feature Usage Distribution:** Which AI assistant capabilities are most frequently used and by whom
- **Usage Consistency:** Whether developers maintain consistent tool usage over time or show declining engagement

How Harness AI Productivity Insights Can Help

Harness AI Productivity Insights (AIPI) empowers engineering leaders to quantify the impact of AI coding assistants by combining SCM data, cohort-based analysis, and developer sentiment surveys. With AIPI, organizations can measure AI's influence on developer velocity, code quality, and overall efficiency, ensuring that investments in AI tools translate into tangible business value.

By leveraging automated data collection from Source Code Management (SCM) tools, AIPI seamlessly tracks key velocity and quality metrics such as PR lead time, commit frequency, and rework rates. Through customizable cohort comparisons, teams can analyze AI adoption trends, contrasting AI-assisted developers with non-AI users to benchmark effectiveness. Additionally, developer sentiment surveys provide qualitative insights into how AI tools impact productivity, collaboration, and job satisfaction.

With Harness AI Productivity Insights, organizations gain a holistic, data-driven framework to optimize AI tool adoption, improve developer workflows, and validate the ROI of AI coding assistants with confidence. By continuously monitoring AI's impact, teams can refine implementation strategies, maximize efficiency, and drive meaningful improvements in software development.

5 . Methodology for Measurement

Measuring the impact of AI coding assistants requires a structured approach combining qualitative data collection with quantitative assessment methods. Organizations should establish baseline measurements before AI tool implementation to enable accurate impact evaluation.

Key concepts in the methodology for measurement include controlled experiments to enable cohort-based comparisons, surveys and interviews to collect qualitative input, and data analysis to distill the large data-sets into comparable information. Success requires consistent application of measurement methodologies across teams and projects, while maintaining flexibility to adapt to different development contexts and requirements.

Controlled Experiments

Controlled experiments provide rigorous evidence of AI coding assistant impact through systematic comparison of development approaches. These experiments typically involve splitting comparable development tasks between teams using AI tools and those using traditional methods.

Task selection requires careful consideration of complexity, scope, and team expertise to ensure valid comparisons. Control groups maintain existing development practices while experimental groups integrate AI assistance. Both groups track identical metrics including development time, code quality, and resource utilization.

Statistical analysis of results helps quantify AI tool impact while controlling for variables like developer experience and project complexity. Time-series experiments track changes in team performance as they adapt to AI assistance, revealing both immediate gains and learning curve effects.

Randomized trials across different project types and team compositions help establish broader patterns of AI tool effectiveness. Cross-team experiments illuminate how AI impact varies with different development styles and expertise levels.

Documentation of experimental conditions, clear success criteria, and thorough data collection ensure reproducible results. Regular calibration of experimental design maintains validity as AI capabilities evolve.

Surveys and Interviews

Qualitative feedback through surveys and interviews provides crucial insights into the human experience of AI coding assistant adoption. Structured surveys capture quantifiable data about developer satisfaction, tool effectiveness, and workflow changes, while semi-structured interviews reveal deeper insights into usage patterns and challenges.

Regular pulse surveys track changes in developer sentiment and tool adoption over time. Success requires careful question design, consistent data collection, and systematic analysis of responses. Regular refinement of survey instruments ensures continued relevance as AI capabilities and team needs evolve.

Anonymous feedback channels encourage honest assessment of AI tool strengths and limitations. The feedback system should summarize answers grouped by cohort.

One-on-one interviews with developers reveal nuanced perspectives on how AI tools affect daily work. Discussion topics explore workflow changes, learning curves, and unexpected benefits or challenges. Team lead interviews provide a broader perspective on productivity and quality impacts.

Focus groups facilitate discussion of shared experiences and best practices across development teams. These sessions help identify common patterns in tool usage and areas for improvement. Cross-functional interviews with product managers and technical leads illuminate broader organizational impact.



Data Analysis

Data analysis methodologies for AI coding assistant impact combine quantitative metrics with qualitative insights. Version control systems, development environments, and project management tools provide raw data about code generation, quality metrics, and development velocity.

Statistical analysis examines patterns in development speed, code quality, and resource utilization before and after AI tool adoption. Time series analysis reveals trends in productivity and quality metrics, while correlation analysis identifies relationships between AI tool usage and performance improvements.

Comparative analysis examines performance differences between AI-assisted and traditional development approaches. This includes analyzing code quality metrics, development velocity, and resource utilization across different teams and project types.

Dashboards and visualization tools present key metrics in accessible formats for stakeholders. Regular reporting cycles ensure timely insights into AI tool impact while enabling quick response to emerging trends or issues.

This holistic approach helps quantify ROI and strategic value while identifying opportunities for optimization.

Key Measurement Capabilities

These key measurement capabilities are needed to ensure that the methodology of measurement provides reliable and trustworthy results.

Automated data collection through development environments and version control systems provides objective metrics on code generation, quality, and deployment frequency. Integration with project management tools enables tracking of development velocity and resource utilization.

Qualitative assessment methods include developer surveys, workflow analysis, and code review feedback. Regular retrospectives help capture insights about tool effectiveness and areas for improvement. Time studies and workflow mapping reveal changes in development patterns and efficiency gains.

A/B cohort testing methodologies allow organizations to compare AI-assisted and traditional development approaches. This controlled testing helps isolate the specific impact of AI tools on development outcomes.

Continuous monitoring systems track real-time metrics while providing trend analysis over time. This ongoing measurement enables organizations to identify patterns, adjust implementation strategies, and optimize tool usage for maximum benefit.

Time-based frameworks enable analysis of both immediate and long-term impacts. Short-term productivity gains should be balanced against sustainable quality improvements and long-term maintainability. Regular calibration of measurement methods ensures continued relevance as AI capabilities evolve.

Integration with existing development metrics and KPIs helps contextualize AI tool impact within broader organizational goals. This holistic approach enables better understanding of ROI and strategic value.

6 . Challenges and Considerations

Measuring AI coding assistant impact presents unique challenges in data collection, analysis, and interpretation. Tool effectiveness varies across different development contexts, team compositions, and project types, making standardized measurement complex.

Measurement consistency requires careful standardization across teams and projects. Different development contexts and tool usage patterns can lead to variable results, necessitating flexible yet comparable measurement approaches. Evolving AI capabilities require regular adjustment of measurement frameworks. As tools advance, organizations must update their metrics and analysis methods to capture new forms of impact.

Defining Clear Measurement Goals

Clear measurement goals serve as the foundation for evaluating AI coding assistant impact. Organizations must align measurement objectives with business goals while considering technical, organizational, and human factors.

Primary measurement areas include development velocity, code quality, and developer experience. Each area requires specific success criteria and measurement approaches tailored to organizational context.

Goal definition must balance quantitative metrics with qualitative indicators. While code generation speed and defect rates offer concrete measurements, factors like developer satisfaction and knowledge sharing require nuanced evaluation approaches.

Timeframe considerations affect goal setting across different measurement periods. Short-term adoption metrics differ from long-term impact assessment, requiring flexible goal frameworks that evolve with tool maturity.

Success criteria should reflect realistic expectations while maintaining ambitious targets. Goals must account for learning curves, team adaptation periods, and varying impacts across different development contexts.

Regular review and refinement of measurement goals ensure continued relevance as AI capabilities and organizational needs evolve. Stakeholder alignment remains crucial for effective goal setting and measurement success.

Look for measurement tools that provide flexible report definition and centrally shared results dashboards to help mitigate these challenges.

Ensuring Data Integrity

Data integrity in measuring AI coding assistant impact requires robust collection methods and validation processes. Automated data collection through development tools must be verified for accuracy and completeness, while manual data entry needs standardized processes to prevent inconsistencies.

Source control systems, development environments, and project management tools should be properly configured to capture relevant metrics without gaps or duplications. Data validation routines help identify anomalies or collection errors before they impact analysis.

Privacy and security considerations necessitate careful handling of sensitive development data. Access controls and anonymization processes protect confidential information while maintaining measurement accuracy.

Cross-validation between different data sources helps ensure consistency and reliability. Regular audits of data collection methods and storage systems maintain integrity throughout the measurement process.

Standardized data formats and consistent metadata facilitate accurate analysis. Clear documentation of collection methods and data transformations ensures reproducibility and trustworthy results. Regular data quality assessments identify potential issues early, enabling prompt corrective action. Success requires ongoing monitoring and maintenance of data collection infrastructure.

Look for measurement tools that provide solid toolchain integrations and automated integration monitoring and alerting to help mitigate these challenges.

External Factors

External factors significantly influence AI coding assistant impact measurement. Market conditions, technology changes, and industry trends affect tool adoption and effectiveness. Economic conditions influence investment in AI tools and related infrastructure.

Team composition changes, including turnover and new hires, impact measurement consistency. Experience levels, technical backgrounds, and individual preferences toward AI tools vary across developers.

Project characteristics affect tool effectiveness measurements. Complexity levels, technical requirements, and deadlines influence how teams utilize AI assistance. Legacy system constraints and technical debt can limit tool impact.

Organizational changes, including restructuring or process modifications, affect measurement accuracy. Policy changes, security requirements, and compliance updates may alter how teams can use AI tools.

Competitive pressures and industry adoption rates influence implementation timelines and expectations. Technology evolution, including AI capability advances and new tool releases, requires ongoing measurement framework adjustments.

Workplace environment changes, including remote work transitions or hybrid models, affect tool usage patterns and team collaboration metrics. Infrastructure changes impact tool performance and availability.

These types of challenges can't all be solved by your measurement tool, but look for measurement tools that provide a flexible measurement system that can simultaneously collect, track, and analyze the necessary data for many different cohorts, over many different reports, so it can provide organization, team, and situation-specific measurements and flex with changing scenarios.

7. Balancing Quantitative and Qualitative Data

Measuring AI coding assistant impact presents unique challenges in data collection, analysis, and interpretation. Tool effectiveness varies across different development contexts, team compositions, and project types, making standardized measurement complex.

Measurement consistency requires careful standardization across teams and projects. Different development contexts and tool usage patterns can lead to variable results, necessitating flexible yet comparable measurement approaches. Evolving AI capabilities require regular adjustment of measurement frameworks. As tools advance, organizations must update their metrics and analysis methods to capture new forms of impact.

Data Analysis

Raw performance metrics require careful contextualization against project complexity, team experience levels, and business requirements. Historical baselines and team-specific contexts help establish meaningful benchmarks while accounting for normal performance variations.

Environmental factors such as infrastructure changes, tooling updates, and process modifications can significantly influence metric interpretation. Understanding these external variables ensures more accurate assessment of AI tool impact.

Business context ultimately shapes metric relevance and priority, with organizational goals and market conditions determining how different measurements should be weighted. Industry standards, such as [DORA metrics](#), and competitive benchmarks provide additional context for evaluating the significance of observed improvements.

Crafting Insightful Questions

Effective qualitative assessment requires carefully designed questions that probe beyond surface-level feedback, examining specific aspects of tool usage, workflow changes, and perceived benefits while avoiding leading or overly broad inquiries. By investigating multiple dimensions including adoption patterns, developer experience, team collaboration, and technical impact, organizations can build a comprehensive understanding of both individual and organizational benefits of AI coding assistants.

Suggested topics to explore include productivity impact, innovation, efficiency, quality of work, and overall satisfaction.

Here are some example questions from the DORA/Harness co-branded survey in the Harness AI Productivity Insights tool (scaled answers)

Productivity

When work is assigned to me,
I have all the necessary
information to start the work
immediately

Innovation

How frequently do you feel
like you're able to find creative
solutions to technical
challenges?

Efficiency

Our code review process
effectively identifies bugs,
potential security issues, or
architectural flaws

Quality

How confident are you that
the tests you write accurately
reflect the quality and stability
of the code?

Satisfaction

How often do you experience
a state of deep focus and
productivity during
development tasks?

Regular refinement of questions ensures continued relevance as teams gain experience with AI tools, while structured collection methods like surveys, interviews, and retrospectives help capture evolving patterns in tool effectiveness.

8 . The Future of AI in Software Development

“ The future effects AI will have on our world remain unclear. But, this year, our survey strongly indicates that AI has produced an unignorable paradigm shift in the field of software development.

DORA Accelerate State of DevOps Report, p26

The trajectory of AI in software development points toward increasingly sophisticated and integrated development experiences, with future AI coding assistants featuring enhanced contextual awareness, deeper understanding of system architectures, and more nuanced code generation capabilities. These advances promise to transform how software is created, making development more accessible while enabling experienced developers to work at higher levels of abstraction.

As AI capabilities expand, the role of developers will continue shifting toward higher-level problem solving and creative design. This evolution requires ongoing adaptation of skills, processes, and measurement approaches, with success depending on the balanced integration of AI capabilities with human expertise.



Emerging Innovations and Trends

Next-generation AI development tools are evolving toward more sophisticated natural language understanding and contextual awareness, enabling intuitive translation of high-level requirements into production-ready code while maintaining architectural consistency. These advances are accompanied by automated system design capabilities that can suggest optimal architectures based on requirements and constraints.

Cross-platform development assistance is improving as AI tools better understand different technology stacks and frameworks, while automated testing and validation capabilities expand to cover more complex scenarios. Enhanced security analysis is becoming more sophisticated, integrating threat modeling and compliance checking throughout the development process.

Success metrics must adapt to capture both technical improvements and enhanced developer experiences in this rapidly advancing landscape. Organizations need to consider how to measure and evaluate these new capabilities while ensuring they maintain focus on meaningful outcomes.

Societal and Economic Implications

“ While AI coding tools democratize software development, they also raise concerns about job displacement and the need for reskilling in the tech workforce.

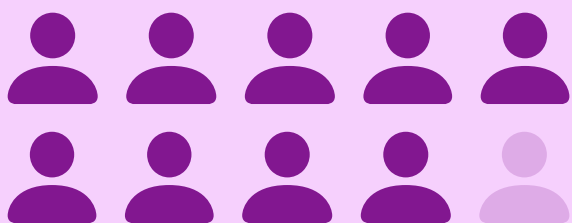
Economic Impact of AI on Software Development, World Economic Forum, 2023, pg7

AI coding assistants are reshaping the software development landscape with broad societal and economic impacts, creating both opportunities and challenges in the job market as these tools augment developer capabilities. While potentially affecting entry-level positions and traditional coding roles, they also democratize development capabilities and create new specialized positions.

Economic implications extend beyond direct productivity gains to industry-wide effects on software development costs and accessibility. Smaller organizations gain access to capabilities previously limited to large enterprises, while early adopters establish competitive advantages in an evolving market.

Workforce development faces transformation as skill requirements evolve, requiring educational systems and professional training to adapt to AI-augmented development environments. Success in navigating these changes requires balanced consideration of productivity gains against broader societal impacts, including effects on workforce development and industry evolution.

The responsible deployment of AI tools must consider ethical implications, including bias prevention, privacy protection, and maintaining developer autonomy. Organizations must ensure these tools support skill growth rather than creating unhealthy dependencies or knowledge gaps.



“ 90% of respondents are concerned that AI tools will replace developers

The State of Software Delivery Report, 2024, pg14



Recommended Next Steps

Organizations looking to measure the impact of AI coding assistants should begin by establishing clear baseline metrics across key areas: developer velocity, code quality, and team satisfaction. This foundation enables meaningful before-and-after comparisons and helps identify areas where AI tools can deliver the most value. The initial focus should be on implementing a balanced measurement framework that combines quantitative metrics (like deployment frequency and defect rates) with qualitative insights from developer surveys and interviews. Organizations should also identify pilot teams for controlled experiments, allowing for systematic comparison between AI-assisted and traditional development approaches.

Success in measuring AI's impact requires ongoing commitment to data collection, analysis, and framework refinement. Organizations should invest in automated data collection tools that integrate with existing development environments, establish regular review cycles to assess and adjust measurement approaches and maintain open communication channels for developer feedback. As AI capabilities evolve, measurement frameworks must adapt accordingly while maintaining focus on meaningful business outcomes. Leadership should prioritize transparent communication about measurement goals and results, ensuring that insights drive continuous improvement in AI tool implementation and usage across the organization.

From Insight to Impact: Charting Your Path

The world of AI is changing fast, and your organization needs to adapt. This ebook provides the map, but you're the one steering the ship. Here's how to navigate your journey toward continuous improvement.

The Core Practice: Start with Your Reality
Continuous improvement always begins with an honest look at where you are today. That's why your first step must be to capture a set of baseline metrics. As you introduce new AI tools and practices, these metrics will allow you to see what's working, what isn't, and why.

A Mindset for Success: Embrace the Lessons
Your journey will have both advances and setbacks. Treat them both as successes. As long as you learn from every outcome, you are making progress. These lessons will inform every future experiment and decision.

Your Next Step

Use the advice in this ebook to identify your first improvement experiment. How can you assess where you are today, and what's the first small change you can make? This iterative cycle of assessing, experimenting, and learning is how you will build lasting capacity for change.

How Harness Can Help You

Harness AI Productivity Insights (API) helps engineering leaders measure how AI coding assistants like Google Gemini Code Assist impact developer productivity. The platform generates detailed reports comparing teams using AI tools against those who don't, tracking both performance improvements and learning curves over time. API looks beyond basic adoption metrics to measure meaningful impacts through code velocity, quality metrics, and developer feedback surveys.

Through simple one-click integration with source code management systems, API provides immediate visibility into productivity gains from AI assistant usage. The tool can be deployed in minutes as part of Harness' Software Engineering Insights (SEI) module. This comprehensive approach helps organizations understand their return on investment in AI coding tools by combining quantitative metrics with qualitative developer insights, giving engineering leaders the data they need to optimize their AI implementation strategies.


Engineering teams can use API to:


- Compare productivity between AI users and non-users
- Track performance improvements as developers master AI tools
- Measure code quality and velocity metrics
- Collect and analyze developer feedback
- Make data-driven decisions about AI tool investments



The AI-Native Software Delivery Platform™

Follow us on

 /harnessio

 /harnessinc

Contact us on

www.harness.io

