McKinsey Digital

Unleashing developer productivity with generative AI

A McKinsey study shows that software developers can complete coding tasks up to twice as fast with generative AI. Four actions can maximize productivity and minimize risks.

This article is a collaborative effort by Begum Karaci Deniz, Chandra Gnanasambandam, Martin Harrysson, Alharith Hussin, and Shivam Srivastava, representing views from McKinsey Digital.

© Getty Images

June 2023

Technology leaders aiming to accelerate software development can expect groundbreaking time savings with generative AI. However, they'll need more than tooling to exploit the full potential of this disruptive technology.

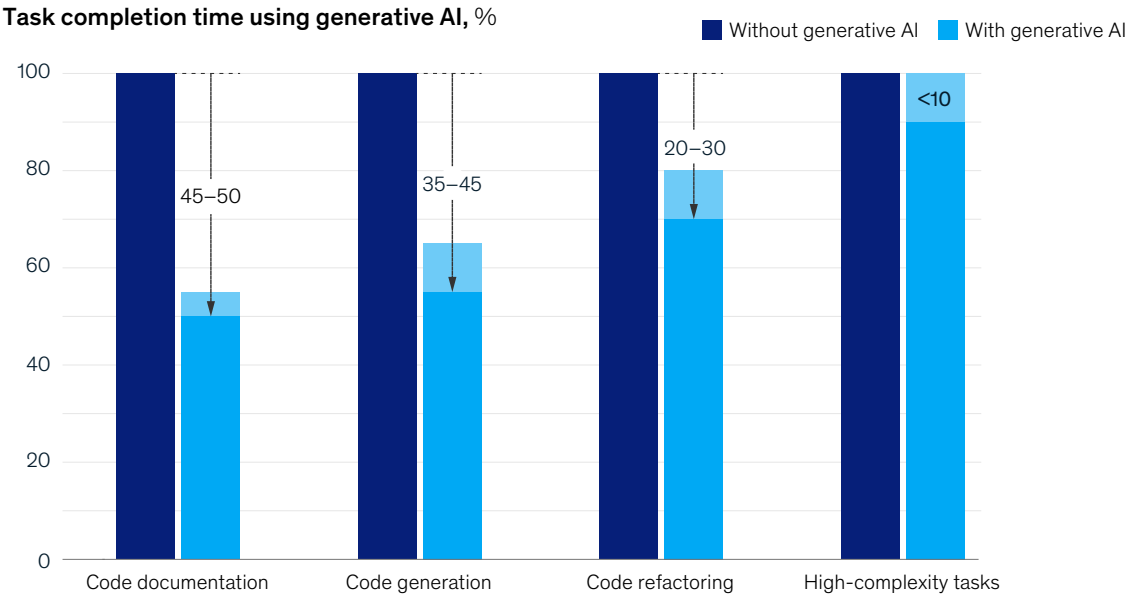
Our latest empirical research finds generative AI–based tools[1] delivering impressive speed gains for many common developer tasks (see sidebar, "About the research"). Documenting code functionality for maintainability (which considers how easily code can be improved) can be completed in half the time, writing new code in nearly half the time, and optimizing existing code (called code refactoring) in nearly two-thirds the time (Exhibit 1). With the right upskilling and enterprise enablers, these speed gains can be translated into an increase in productivity that outperforms past advances in engineering productivity, driven by both new tooling and processes.

Yet, while a massive surge in productivity is possible, our research finds time savings can vary significantly based on task complexity and developer experience. Time savings shrank to less than 10 percent on tasks that developers deemed high in complexity due to, for example, their lack of familiarity with a necessary programming framework. A similar result was seen among developers with less than a year of experience; in some cases, tasks took junior developers 7 to 10 percent longer with the tools than without them.

Using these tools did not sacrifice quality for speed when the developer and tool collaborated. Code quality in relation to bugs, maintainability, and

Exhibit 1

Generative AI can increase developer speed, but less so for complex tasks.



[1] Includes both generative AI–based tools trained to have natural conversations through prompting and those trained specifically on code base and embedded into a developer's integrated development environment (IDE).

About the research

To understand the impact of generative AI–based tools on developer productivity, we set up a lab with more than 40 McKinsey developers who are located across the United States and Asia and have different amounts of software development experience. This lab will serve as an ongoing test bed to understand developments in the industry, including the impact of new tools and developments in existing tools.

For this report, participants were asked to perform common software development tasks in three areas—code generation, refactoring, and documentation—over the course of several weeks. Each task was performed by a test group that had access to two generative AI–based tools and a control group that used no AI assistance. Each developer participated in the test group for half of the tasks and in the control group for the other half.

We collected data in different formats. A demographics survey was used to gauge their years of experience, expertise, and prior knowledge. For measuring time spent on each task, participants recorded start time, end time, and break times. Task surveys after each task captured perceived complexity of the task and developer experience. Judge evaluations during code demos were used to identify successful submissions. Automated reviews of code quality using an open-source platform assessed code readability and maintainability and detected bugs. And a post-experiment survey gathered insights on participants' impressions of the tools and experience across tasks.

readability (which is important for reusability) was marginally better in AI-assisted code. However, participant feedback indicates that developers actively iterated with the tools to achieve that quality, signaling that the technology is best used to augment developers rather than replace them. Ultimately, to maintain code quality, developers need to understand the attributes that make up quality code and prompt the tool for the right outputs.

Together, these findings suggest that maximizing productivity gains and minimizing risks when deploying generative AI–based tools will require engineering leaders to take a structured approach that encompasses generative AI training and coaching, use case selection, workforce upskilling, and risk controls. In this article, we share where generative AI shined in our research, which tasks demanded developer expertise, and what engineering leaders can do to ensure the most effective use of this burgeoning technology.

Where generative AI shined

In our study, we assigned developers some garden-variety tasks that software teams do regularly: refactor a piece of code into microservices to improve maintainability and reusability, build new application functionality to elevate the customer experience, and document code capabilities so future changes are easier.

Across these tasks, our research finds generative AI–based tools enable tremendous productivity gains in four key areas:

— **Expediting manual and repetitive work.** Generative AI can handle routine tasks such as auto-filling standard functions used in coding, completing coding statements as the developer is typing, and documenting code functionality in a given standard format, based on the developer's prompt. In doing so, these tools can free developers to solve more complex business challenges and fast-track new software capabilities.

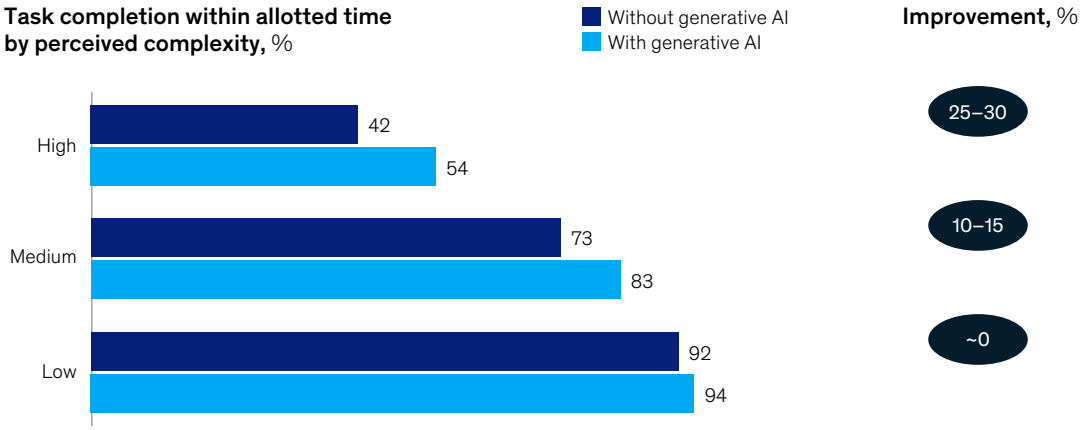
— **Jump-starting the first draft of new code.** When facing a blank screen, developers with generative AI–based tools can request suggestions by entering a prompt in a separate window or within the integrated development environment (IDE) they use to develop software. Developers who did so reported that the generative AI–based tools provided helpful code suggestions. This enabled them to escape writer's block so they could get started more quickly. As one participant shared, the tools enable developers to get in the "flow" sooner.

— **Accelerating updates to existing code.** Participants also reported that when using these tools with effective prompting, they could make more changes to existing code faster. For instance, to spend less time adapting code from an online coding library and improving prewritten code, developers would copy and paste it into a prompt and submit iterative queries requesting the tool to adjust based on the criteria they provided.

— **Increasing developers' ability to tackle new challenges.** While developer time savings with generative AI–based tools were more modest for complex tasks, our research still finds benefits: the technology can help developers rapidly brush up on an unfamiliar code base, language, or framework necessary to get the job done. Furthermore, when developers face a new challenge, they can turn to these tools to provide the kind of help they might otherwise seek from an experienced colleague—for example, explaining new concepts, synthesizing information (say, by comparing and contrasting code from different repositories), and providing step-by-step guides on how to use a framework so they can do the work. Thus, developers using generative AI–based tools to perform complex tasks were 25 to 30 percent more likely than those without the tools to complete those tasks within the time frame given (Exhibit 2).

Exhibit 2

Developers using generative AI to assist with complex tasks were more likely to complete those tasks within a given time frame.



The benefits go beyond these productivity improvements. The research finds that equipping developers to be their most productive also significantly improves the developer experience, which in turn can help companies retain and excite their best talent. Developers using generative AI–based tools were more than twice as likely to report overall happiness, fulfillment, and a state of flow (Exhibit 3). They attributed this to the tools' ability to automate grunt work that kept them from more satisfying tasks and to put information at their fingertips faster than a search for solutions across different online platforms.

Which tasks demand developer expertise

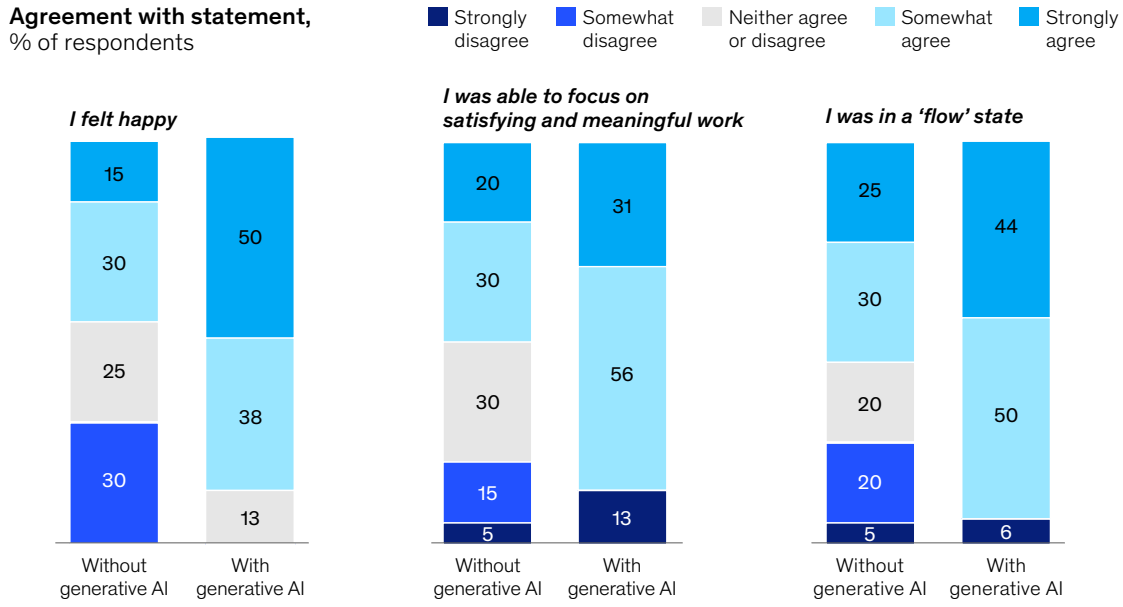
Generative AI technology can do a lot, but our research suggests that the tools are only as good as the skills of the engineers using them. Participant feedback signaled three areas where human oversight and involvement were crucial:

— **Examining code for bugs and errors.** Research participants reported that, at times, generative AI–based tools provided incorrect coding recommendations and even introduced errors in the code. During one task, a developer noted she had to input numerous prompts to correct a tool's erroneous assumption so she could get an answer to a question. In another case, a developer shared that he had to "spoon-feed" the tool to debug the code correctly.

— **Contributing organizational context.** While off-the-shelf generative AI–based tools know a lot about coding, they won't know the specific needs of a given project and organization. Such knowledge is vital when coding to ensure the final software product can seamlessly integrate with other applications, meet a company's performance and security requirements, and ultimately solve end-user needs. As research

Exhibit 3

Generative AI tools have potential to improve the developer experience.



Note: Figures may not sum to 100%, because of rounding.

participants pointed out in their feedback, it will be up to software developers to provide these tools with the context via prompting, including how the code will be used and by whom, the types of interfaces and other systems the software will interact with, the data used, and more.

— **Navigating tricky coding requirements.** Participant feedback also suggests generative AI–based tools are better suited for answering simple prompts, such as optimizing a code snippet, than complicated ones, like combining multiple frameworks with disparate code logic. One participant shared that to obtain a usable solution to satisfy a multifaceted requirement, he first had to either combine the components manually or break up the code into smaller segments. As another participant explained, "[Generative AI] is least helpful when the problem becomes more complicated and the big picture needs to be taken under consideration."

What do these findings mean for technology leaders?

Given these findings, what can technology leaders do to translate these time savings and quality improvements into real productivity gains while minimizing risk when using generative AI in software development? Our research participants' experience suggests starting with four priorities: skill development, pursuing advanced use cases, planning for skill shifts, and risk management.

Provide developers with generative AI training and coaching

For developers to effectively use the technology to augment their daily work, they will likely need a combination of training and coaching. Initial training should include best practices and hands-on exercises for inputting natural-language prompts into the tools, often called prompt engineering. In addition, workshops should equip developers with an overview of generative AI risks, including any industry-specific data privacy or intellectual-property issues and best practices in reviewing AI-assisted code for design, functionality, complexity, coding standards, and quality, including how to discern good versus bad recommendations from the tools.

For developers with less than a year of experience, the research also suggests a need for additional coursework in foundational programming principles—for example, coding syntax, data structures, algorithms, design patterns, and debugging skills—to achieve the productivity gains observed among those with more experience.

Once developers begin using the tools in their day-to-day activities, their skill development should continue with ongoing coaching from senior team members and community building, such as dedicated online channels and team meetings to share practical examples. This effort can foster continuous learning, ensure best practices are shared throughout the organization, and identify any issues early. In our research, participants noted that as they generated more prompts and shared learnings with each other, the quality of their prompts improved.

Pursue advanced use cases beyond code generation

While there is tremendous industry buzz around generative AI's ability to generate new code, our research shows that the technology can have impact across many common developer tasks, including refactoring existing code, which can enable leaders to make a dent in traditionally resource-intensive modernization efforts that often get sidelined due to lack of time. For example, if generative AI–based tools help teams rapidly refactor a legacy application, the teams can redirect their time to closing out a backlog of improvements that have languished on their company's to-do list or improving architectural performance across the entire software platform.

Deploying new use cases requires a careful evaluation of tooling, as a flurry of new generative AI tools are coming to market and different tools excel in different areas. Our research shows that using multiple tools can be more advantageous than just one. During our study, participants had access

to two tools, one that used a foundation model trained to respond to a user's prompt and another that used a fine-tuned foundation model trained specifically on code. Participants indicated that the former, with its conversational capabilities, excelled at answering questions when they were refactoring code. The latter tool, they said, excelled at writing new code, thanks to its ability to plug into their integrated development environment and suggest code from a descriptive comment they noted within their document. However, when developers used both generative AI tools within a given task, as opposed to only one, they realized an additional time improvement of 1.5 to 2.5 times.

Plan for skill shifts

As developers' productivity increases, leaders will need to be prepared to shift staff to higher-value tasks. Baselining productivity and then continuously measuring improvement can reveal new capacity as it emerges across the organization. Leaders should consider how to use their additional capacity and what upskilling is needed to close any skill gaps that may emerge. They might, for example, apply their talent to enable new business expansion or update existing products more often. These assignments would require developers to build new skills in software design and architecture.

Provide risk controls

New data, intellectual-property, and regulatory risks are emerging with generative AI–based tools. Given the speed at which developers can write or update code with these tools, it's easy to imagine how any problems from, say, a coding error or data issue could snowball. As leaders update governance, they should consider potential risks such as the following:

- *data privacy and third-party security,* such as the potential for developers to expose confidential information when prompting the tools
- *legal and regulatory changes,* including changes to the European Union's General Data Protection Regulation (GDPR) and other regulations limiting the use of the technology
- *AI behavioral vulnerabilities,* including the impacts if bad actors plant malicious or malfunctioning code in the public domain to influence the training of large language models or infiltrate organizations
- *ethics and reputational issues* that could arise from using a snippet of code copyrighted by another entity or amid debates on ownership of code the tools generate
- *security vulnerabilities* that can crop up in AI-generated code and put systems (and the organization) at risk

---

Generative AI is poised to transform software development in a way that no other tooling or process improvement has done. Using today's class of generative AI–based tools, developers can complete tasks up to two times faster—and this is just the beginning. As the technology evolves and is seamlessly integrated within tools across the software development life cycle, it is expected to further improve the speed and even quality of the development process. But as our research shows, tooling alone is not enough to unlock the technology's full potential. A structured approach encompassing generative AI training and coaching, use case selection, workforce upskilling, and risk controls can lay a solid foundation for organizations to pursue generative AI's promise of extraordinary productivity and unparalleled software innovation.

Begum Karaci Deniz is a consultant in McKinsey's Bay Area office, where **Chandra Gnanasambandam** is a senior partner and **Martin Harrysson**, **Alharith Hussin**, and **Shivam Srivastava** are partners.

The authors wish to thank Ishita Agarwal, Monica Chandni, Elsie Jiang, Caroline Moody, Akhila Nandgopal, Diego Guerra Orozco, Sia Peng, Vanessa Randall, Shriya Ravi Shankar, Aakanksha Srinivasan, and Olivia Zhang for their contributions to this article.

Copyright © 2023 McKinsey & Company. All rights reserved.