How enterprise engineering teams can successfully adopt AI

WRITTEN BY GITHUB WITH

Foreword

The software development industry has reached a pivotal milestone in the evolution of generative AI (artificial intelligence). While much of the world has grappled with this technology and its use cases, one survey by GitHub found that 92% of developers said they were already using generative AI coding tools both in and outside of work in early 2023[1]. This shows that developers are quickly embracing the benefits of AI in their software builds, often before organizations have considered how best to operationalize AI across engineering teams.

The last time the software industry had to implement change management at this scale was with the introduction of DevOps as a development methodology over a decade ago. Now, the industry is in a watershed moment where developer tool chains are evolving faster than ever before with the widespread availability of AI—one Gartner study has even found that 80% of code will be produced by AI in 2026[2].

As AI tools continue to evolve, engineering leaders now face a world where AI is infused throughout far more of the software development lifecycle (SDLC). So, how should enterprise engineering leaders think about AI's evolving role in software development and ensure that teams are prepared to ship high- quality, secure software at scale?

In this guide, we'll explore best practices for adopting AI-powered software development in enterprise engineering teams and the benefits that come with unifying your tech stack with a single, AI- powered platform.

1: GitHub, "Survey reveals AI's impact on the developer experience," June 2023.

2: ZDNET, "80% of enterprises will have incorporated AI by 2026, according to a Gartner report." October 2023.

What's inside

AI-powered software development, explained

AI coding tools—also known as AI-powered coding assistants or AI programming tools—are software applications that utilize generative AI and machine learning techniques to help developers and programmers in various aspects of the software development process. These tools are designed to augment the capabilities of developers, improve code quality, and accelerate the software development process—and they have become increasingly valuable in modern software development by helping developers be more productive and efficient in their work.

These tools used to primarily live mainly in the integrated development environment (IDE) to help developers write and refine code. Today, industry leaders are increasingly applying AI to more parts of the typical developer workflow ranging from the command line interface (CLI) to infrastructure orchestration, security vulnerability detection, and more.

The promise of how AI is evolving in software development is apparent in the results they provide including increased productivity, collaboration, and innovation on engineering teams when they're writing code. 70% of developers who use AI coding tools, for instance, say they offer them significant advantages at work—and more than 4 in 5 developers expect AI coding tools will make their teams more collaborative[3]. Moreover, research at GitHub has found that developers on average complete tasks up to 55% faster with AI coding tools[4].

AI also serves as a cost-effective solution for engineering leaders by delivering simplified approaches to managing technology stacks. Through efficient resource allocation, task automation, and predictive maintenance, AI can both optimize operational expenses and minimize downtime risks.

These points present a strong incentive for engineering leaders in enterprise environments to operationalize AI across their developer workflows. In the next section, we'll take a look at a strategic approach for engineering leaders to operationalize these tools for their teams.

3: GitHub, "Survey reveals AI's impact on the developer experience," June 2023.

4: GitHub, "Research: quantifying GitHub Copilot's impact on developer productivity and happiness," September 2022.

The evolving capabilities of AI-powered developer platforms

Here's a quick glance at some capabilities of modern AI-powered tools:

- **Code autocompletion** suggests and automatically completes code snippets or commands as a developer types to enhance efficiency and reduce errors.
- **Code generation** tools automatically produce source code or documentation based on predefined templates, which ultimately simplifies and expedites the development process.
- **Code analysis** AI tools leverage machine learning techniques to understand, interpret, and provide insights into software code for quality assurance purposes. For example, AI tools can assess code compatibility across different platforms, frameworks, or libraries, to ensure that software components work seamlessly together.
- **Code refactoring** tools automatically analyze and restructure code to improve its readability, maintainability, and overall quality.
- **Bug detection** can be leveraged to identify and highlight errors or flaws in code during the development phase, which helps developers produce more reliable and robust software.
- **AI-powered application security testing**, which employs machine learning to autonomously analyze code, identify vulnerabilities, and generate remediation suggestions, has the potential to revolutionize how developers build secure applications from the start—and radically transform the traditional definition of "shift left."
- **Collaborative coding** can be encouraged with AI-powered software development tools by providing intelligent code suggestions, automating routine tasks, and offering real-time insights, which enhances communication and productivity among developers working on shared projects.
- **Natural language processing** can be used by AI-powered software development tools to understand and interpret human language, which enables developers to interact with the tools using natural language commands, queries, or comments, and facilitates more intuitive and efficient communication in the development process.

The unique challenges of operationalizing AI in software development at the enterprise level

Amid extensive changes in technology driven by generative AI and increasingly complex codebases, coupled with legacy applications, more and more engineering leaders recognize they need a new approach to deliver innovative, secure software fast and at scale.

Traditional development and DevOps platforms are not as well-suited for the fast-evolving demands of AI-powered development. That's especially true when it comes to the combination of platform engineering, operational management, and developer experience.

Current technology stacks and platforms are meant to support DevOps and DevSecOps teams while tacking on novel AI-powered capabilities—but these tools and capabilities often don't work smoothly together. This can lead to:

- A disconnected experience for development, security, and operations teams.
- Challenges like miscommunication, increased technical and security challenges, an overwhelming amount of technologies, and opaque operational costs.
- Reduced productivity, a weaker security stance, delayed time to market, and, as a consequence, negative effects on an organization's financial performance.

To avoid these pitfalls, organizations can turn to AI-driven developer platforms with built-in toolsets and workflows. The primary driving goals are to minimize the need for developers to shift between different contexts; enhance collaboration among different teams; and remove the obstacles that hinder the development, expansion, and secure delivery of software.

How AI is reshaping the enterprise software development lifecycle

Since the initial launches of popular AI-powered tools GitHub Copilot as an IDE extension and OpenAI's ChatGPT, the pace of innovation and rapid iteration across the technology industry around generative AI has been striking. AI coding tools once solely recommended lines and blocks of code. Now, they're being extended across the entire SDLC.

Some engineering leaders in enterprise environments who were early adopters have already seen the impact AI is having on their development teams. Mercado Libre, for instance, has more than 9,000 developers using GitHub Copilot and has quantified a 50% reduction in the time it takes its engineers to write code with AI[5].

And as these tools evolve to cover more of the developer experience, there's a large scope of where these tools can be used—and are being used—in the SLDC to great effect. The best AI tools and platforms come with integrated capabilities and workflows that reduce context switching, foster collaboration, and remove operational barriers. In simpler terms, these tools make it easier for developers to write, secure, and deliver code—while collaborating more effectively with colleagues by helping explain existing codebases, decision logs, and organizational documentation.

In short, AI is reshaping how software is made in enterprise environments—and engineering leaders are already seeing benefits. For instance, here's an example of how AI—and the GitHub platform specifically—can enhance every part of the SDLC:

1. **Planning.** In the planning phase of the SDLC, AI can help product and development teams conduct market research, generate ideas, assess potential risk, and offer predictive analysis. GitHub Copilot, for instance, can help accelerate software development in the planning stage by suggesting code snippets as developers describe their ideas in natural language.
2. **Solution design.** By integrating AI into the solution design phase, development teams can benefit from intelligent insights into potential solutions, get suggestions for alternative solutions, automate routine tasks such as security vulnerability filtering, and enhance collaboration by giving engineers access to faster solution development. This ultimately leads to more effective and user-friendly software designs. GitHub's developer platform helps developers in the solution design phase by offering version control for design files, AI-generated solution suggestions, easy access to existing solutions and documentation via AI-powered search, more collaborative workflows, and a centralized platform for issue tracking.
3. **Coding and development.** In the coding or development phase, developers have to translate system design specifications into actual code. It's critical that developers follow best practices for writing clean, maintainable, and efficient code, and AI tools such as GitHub Copilot's chat

5: GitHub, "Mercado Libre frees developers' minds to focus on their mission with GitHub."

feature can assist in that process by allowing developers to ask questions about code quality, what existing code in a codebase does, debug code on the fly, or find solutions right from their IDE. Plus, GitHub Copilot can use the full context of your codebase to provide personalized results across the entire developer workflow.

4. **Testing.** The best AI tools can help automate test case generation, analyze large datasets, identify potential bugs and vulnerabilities, and enhance overall test coverage by automatically opening up pull requests with additional suggested tests. GitHub code review, for instance, can help in the testing phase of the SDLC by allowing both QA and engineering teams to collaboratively review and analyze test scripts, which not only ensures code quality but can help identify potential issues or improvements. These tools can also complement static application security testing solutions (SAST). GitHub Advanced Security, for instance, has automated security checks that are run with every pull request, which surface issues in the context of the development workflow so vulnerabilities are fixed in minutes, not months.
5. **Deployment.** During deployment, AI can help simplify the process by handling release management, predicting possible performance issues, and optimizing infrastructure. It also speeds up the process and ensures reliability through automated checks, which reduces the need for manual effort.
6. **Maintenance and support.** Software requires ongoing maintenance to ensure that it remains performant, and developers periodically issue software patches and updates to fix bugs in the software and resolve security issues. AI can help teams more proactively detect and resolve issues by analyzing historical data and patterns, predicting potential system failures, and automating routine maintenance tasks. Additionally, AI-powered coding tools can help engineers work on legacy applications and codebases by making it easier to refactor code—or code in a language that may not be as familiar to them.

The benefits of taking a platform-first approach to AI

A platform-first approach involves integrating artificial intelligence directly into a unified software platform, rather than relying on standalone AI solutions, to incorporate AI capabilities seamlessly within existing development, collaboration, or operational workflows. This results in an AI-powered developer platform that can support your teams from planning to deployment and beyond.

GitHub knows a platform-first approach provides a unified ecosystem where AI capabilities can help developers, and, in turn, offer business benefits around time savings, cost savings, a more secure end product, and faster time to market, increasing overall developer–and organizational–productivity and satisfaction. In addition to those benefits, a platform- first strategy:

- **Builds AI directly into the entire SDLC.** Instead of adding another tool to your technology stack, a platform-first approach means AI is fully baked into a developer platform that can support you at every step of the development process.
- **Reduces context switching.** By unifying your tools in a centralized platform, developers can readily access the products they need to keep them productive and stay in the flow.
- **Provides a custom-tailored experience.** For example, with GitHub Enterprise, AI can access your organization's data and codebase to generate personalized suggestions and responses to queries and natural language prompts related to your documentation and code.
- **Strengthens your security posture.** Integrating AI into a unified platform allows you to apply consistent protocols across your ecosystem to reduce vulnerabilities and create a more robust defense against potential threats.

A roadmap for operationalizing AI in enterprise engineering teams

Amidst the rapid pace of innovation and evolution among AI coding tools, it's critical for engineering leaders to understand what products and platforms are enterprise ready and how to make decisions about where, when, and how to adopt and operationalize these tools at scale.

At GitHub, we often advise companies and customers on best practices around operationalizing AI coding tools. This often breaks down into a structured approach that takes into consideration factors and environments unique to the organization in question.

Here's a roadmap for how to consider this:

1. **Assess your organizational needs and objectives.** Engineering leaders need to pinpoint and understand the specific goals of implementing AI tools, such as improving productivity, code quality, or security, and how these tools can help them achieve key objectives.
2. **Make improving team collaboration your north star.** The implementation of these tools needs to be a team effort and often involves cultural shifts. That makes it critical to involve key stakeholders such as distinguished developers, team leads, and engineering managers in the decision-making process of what new workflows will look like. It's also important to foster open communication about both the benefits and the challenges of AI tools so that teams are prepared to capitalize on their benefits.
3. **Invest in training and onboarding.** To ensure that team members have the necessary skills to effectively use these tools, as well as reduce technical debt, leaders should provide training and onboarding sessions for the engineering teams to get comfortable with navigating the platform. Leading providers of AI-powered coding tools and developer platforms will often offer onboarding courses. At GitHub, we often work with our parent company Microsoft to develop everything from video resources to documentation and helpful guides.
4. **Start small with pilot teams and projects.** Whenever you're adopting a new developer tool or platform, it's best to start small—and that's no different with AI-powered software development. At GitHub, we often advise companies to begin with small-scale pilot projects and teams to test the AI tools' effectiveness and identify any challenges and benefits in advance of rolling out tools to a wider organization.
5. **Feedback cycles.** Leaders should set up feedback loops to collect input and suggestions from engineering teams and use this feedback to continuously evaluate workflows and make sure the AI tools meet evolving needs.
6. **Integrate AI in existing workflows.** At GitHub, we find the best AI tools fit into existing developer workflows—that means developers have less of a need to learn a new workflow than increase throughput in

their existing workflows. For engineering leaders, it's important to ensure their developers stay in the flow and avoid context switching, making it important to strategically select the right AI tools—and platform— that fit seamlessly into established processes.

7. **Evaluate tools stringently for data privacy and security.** Amid a rapidly evolving field of AI coding tools, it's imperative to ask vendors about the data privacy and security standards they have designed their tools around. Moreover, it's important to address concerns when using AI tools with your teams, and set organizational policies and governance standards around the utilization of these tools (i.e., ensuring your developers are using sanctioned tools and not freely available tools on the internet).
8. **Monitor productivity gains and organizational performance.** Similar to investing in observability, engineering leaders should prioritize investment in solutions for monitoring the performance and impact of AI tools on the productivity and code quality of engineering teams. This should include evaluating quantitative data around pull requests, code delivery rates, deployment speeds, and more. Engineering leaders should also seek to evaluate qualitative gains via developer surveys to understand how developers feel about using these tools.
9. **Start small—and scale up across the developer workflow.** To repeat a point above: It's important to start small with AI, and focus on individual parts of the SDLC. This may mean starting with an AI coding tool in your engineers' IDEs to start while piloting other applications—such as in your documentation, version control solution, or otherwise—with small groups of people. In short, leadership should plan for operationalizing AI today while also looking forward to how to scale AI tools across more and more of the SDLC within their organization.
10. **Build out your innersource culture with documentation and best practices.** To maximize the benefits of AI coding tools, engineering leaders should encourage their teams to create documentation, standardize processes, and innersource—or make public—specific code, solutions, and best practices for developers to leverage via AI retrieval tools that can query for this information inside and outside of the IDE. Having an active innersource culture will help organizations win today and tomorrow as AI coding tools can help developers find information faster than traditional means and have the right content to focus on what matters most: building great software.
11. **Focus on continuous improvement.** As AI solutions rapidly advance and evolve, vendors and platform providers will continue to iterate on these tools—and that makes it critical to stay up to date and evaluate what improvements can be made to existing processes and future workflows.

12. **Create an AI-friendly culture that puts developers in control.** Some developers may be concerned that AI will make their roles redundant—but that couldn't be further from the truth. Between historically stagnating productivity rates[6] and a global shortage of engineering talent[7], AI is poised to help developers build software faster, navigate new codebases, upskill on the job, and collaborate more effectively[8]. That makes it important to encourage a culture of innovation and learning within the engineering teams around the utilization of AI.

This AI roadmap for enterprise engineering teams isn't a checklist—it's a strategic guide to help you consider how to seamlessly integrate AI into your workflows. It's all about aligning AI tools with what matters most to your organization, fostering a collaborative culture, and ensuring your team is equipped with the right skills. As the AI landscape keeps evolving, remember that it's not just about adopting new tech. It's about creating an environment where your developers can thrive, innovate, and build fantastic software faster.

6: U.S. Bureau of Labor Statistics, "The U.S. productivity slowdown: an economy-wide and industry-level analysis." April 2021.
7: GitHub, "The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot." June 2023.
8: GitHub, "The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot." June 2023.

Take this with you

The software development industry has reached a point where it stands to be completely revolutionized by AI. And the unique characteristics of AI development, including its intricate workflows, resource demands, diverse toolchains, and collaborative nature, necessitate platforms that are purpose-built to address these challenges.

Meet GitHub

Home to more than 100 million developers, GitHub is the world's most trusted and adopted AI-powered developer platform that's empowering organizations to build, secure, and ship software faster to unlock innovation at scale. Our comprehensive platform integrates enterprise-grade tools, such as CI/CD, automation, application security testing, cloud development environments, collaboration tools, and AI-powered coding tools, to facilitate the swift delivery of secure software. Plus, it is compatible with all cloud providers, so companies can confidently scale their software delivery without sacrificing familiarity.

"We see GitHub's platform continually evolving with new features that are super helpful. The clear winner recently has been GitHub Copilot, where we've seen amazing results from the trials we've been running with our teams."

Lucia Brizuela // Senior Technical Director, Mercado Libre

Next steps

- Learn more about GitHub Enterprise to begin unlocking innovation at scale with AI
- Request your GitHub Enterprise demo
- Take GitHub Copilot on a test flight
- Set up your GitHub Enterprise Cloud trial