

THE NEW/STACK

Better, Faster, Stronger:

How **Generative AI**
Transforms Software
Development

The New Stack

Better, Faster, Stronger: How Generative AI Transforms Software Development

Alex Williams, Founder and Publisher

Ebook Team:

Andrew Tillery, Digital Marketing Manager

Ben Kubany, Project Manager

Celeste Malia, Marketing Consultant

Diana Gonçalves Osterfeld, Designer

Heather Joslyn, Editor in Chief

Janakiram MSV, Author

Judy Williams, Copy Editor

Michael Baker, Copy Editor

Supporting Team:

Aaron J. Ban, Director of Software Development

Michelle Maher, Assistant Editor

Tony Sala, Director of Sales

Vicki Walker, Senior Sponsor Editor

Vinay Shastry, Director of DevOps

© 2024 The New Stack. All rights reserved.

20240312

Table of Contents

Sponsor.....	4
Introduction: What's Generative AI for Developers?	5
How Software Development Evolved	9
How the Software Development Life Cycle Has Changed	9
Navigating Today's Software Development Challenges.....	12
How SDLC Challenges Hold Businesses Back.....	15
Transforming the SDLC With Generative AI.....	17
How AI Coding Assistants Empower Developers	17
AI Assistants for DevOps and Data Science.....	21
The Impact of AI Coding Assistants on Teams.....	22
How Businesses Benefit From AI Coding Assistants	25
The Risks Associated With AI Coding Assistants	27
Assessing the Key Risks Organizations Face	27
Understanding the Legal Implications	29
The Rise of Shadow AI in Enterprises.....	31
How To Choose an AI Coding Assistant Platform: Checklist.....	33
Safe Deployment of AI Assistants	34
Integrate AI Coding Assistants With DevSecOps	34
Prioritize Privacy and Data Security	35
Choose Private and Isolated Code-Generation Models.....	36
Ensure Compliance With Open Source Licenses	37
Conduct Stringent Code Reviews.....	38
Track and Measure Developer Productivity Gains	39
Conclusion	41
About the Author	43
Disclosure	44

Sponsor

We are grateful for the support of our ebook sponsor:



Tabnine is the AI coding assistant that accelerates and simplifies software development while keeping your code private, secure and compliant.

INTRODUCTION

What's Generative AI for Developers?

[The launch of OpenAI's ChatGPT](#) in November 2022 marked a milestone in the evolution of computing. Generative AI, a term that was previously confined to academicians, researchers and senior data scientists, instantly became a buzzword. The developers of this technology made [artificial intelligence](#) ubiquitous by making the technology easier to understand and bringing it closer to consumers.

Artificial intelligence is typically used to perform predictions or classifications, such as identifying an image of an animal as a dog or cat or detecting if a credit card transaction is genuine or fraudulent. Generative AI aims at creating new content in the form of text, images or audio based on foundation models that are trained on vast datasets available in the public domain.

Foundation models such as [large language models \(LLMs\)](#) and diffusion models are already powering some of the most popular tools like ChatGPT and Midjourney, which are used to [produce content based on a simple textual prompt](#).

Code models are specialized language models trained specifically to understand, generate, interpret and sometimes execute code. Code foundation models are trained

on vast datasets made up of code from a variety of programming languages, libraries and frameworks, collected from public repositories and other sources.

Developers access these models through tools within their integrated development environments (IDEs) or through specific APIs, enabling features like code completion, bug detection, code summarization and even the generation of code snippets based on natural language descriptions. These models significantly enhance developer productivity and satisfaction by providing intelligent suggestions and automating repetitive coding tasks.

The combination of code models and tools has made developers more productive by generating code as they type, or based on written prompts or inline comments.

When integrated with mainstream developer tools and dev environments, they become trusted coding assistants, coaches and efficient pair programmers for developers.

Generative AI for software development such as [GitHub Copilot](#) and [Tabnine](#) can be leveraged by developers through plugins in their IDEs.

AI-based coding assistants increase developer productivity and reduce the time required to write routine, mundane code, empowering the programmer to focus on the core logic, which is the essence of software development. These coding assistants operate in the background and are instantly available to the developers when they need them.

Developers are quickly embracing AI coding assistants to increase their productivity — and make their organizations more competitive. [Seventy percent of developers reported they are using or will use AI in their development process](#), according to the 2023 Stack Overflow Developer Survey.

What's more, their organizations are placing bets on generative AI to help them become more nimble, seize new business opportunities and serve customers better. Forty percent of companies in McKinsey's 2023 Global Survey said [they are increasing their investment in AI](#). In the consultant's report on the state of AI released last year, more than two-thirds of the survey's 1,600-plus participants worldwide said they expect their company's investment in AI to increase in the next three years.

Some coding assistants include specialized chatbots that are being embedded into IDEs to bring familiar conversational AI functionalities directly to developers. This integration allows developers to interact with their IDEs using natural language, making it easier to perform tasks such as searching for files, understanding complex codebases, generating code snippets, debugging and even getting explanations on code functionalities.

By leveraging the power of natural language processing and machine learning, these coding assistants can understand and execute developer queries, streamlining development workflows and enhancing productivity. This approach not only makes the IDE more intuitive and user-friendly but also leverages the full potential of generative AI to support and accelerate the entire software development process, not just programming.

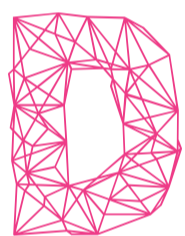
Like other generative AI tools, code generated by models comes with its own set of challenges and risks. Since the models are mostly trained on public code repositories, they tend to inherit the same issues and blindspots often noticed in junior developers' code by senior developers and experienced programmers. This is similar to the risks associated with LLMs, leading to bias, inaccuracy and [hallucinations](#) in the generated content.

From an ethical, compliance and legal standpoint, code models trained on permissively licensed data foster creativity and collaboration. Conversely, models trained on data from unlicensed public sources may infringe on copyrights, raise privacy concerns and perpetuate biases present in the data. Consequently, these issues could pose significant challenges for AI development.

Despite the challenges, generative AI has a significant impact on software development, influencing the evolution of the software development life cycle (SDLC). This ebook focuses on how developers can embrace generative AI while factoring in the associated risks and benefits.

CHAPTER 01

How Software Development Evolved



Developers are expected to build and change their code a lot faster these days than they have in the past. The changing nature of technology and the rising demands of the global market are the driving forces behind this evolution.

With a workforce spread across the globe, intense competition and an ever-increasing demand for qualified developers, engineering teams today face a unique convergence of challenges. In exploring these developments, this chapter looks at how software creation is changing fundamentally.

How the Software Development Life Cycle Has Changed

During the initial days of computing, software applications were built following the waterfall model. For almost two decades, this model remained the gold standard of software development. With its well-defined, linear and sequential approach, it offered a predictable path to shipping software.

In the waterfall model, a product or solution is built through a sequence of steps that starts with a subset of the requirements. The approach meant that the team could not proceed until all of the dependencies that came before them were resolved, which had a significant impact on the timeline because every step had to be completed to reach the final milestone.

While this model simplified development, it wasn't fast or flexible enough to keep pace with the evolution of technology. Waterfall production struggled to accommodate changes once the process was underway, often leading to costly overruns and outdated systems by the time of delivery. The inability to adapt to changes during implementation frequently resulted in expensive delays and systems that were already obsolete when they were delivered.

The emergence of personal computing and client/server architecture, which has increased the cadence of software delivery, gave rise to the popularity of agile methodologies. Based on an iterative development approach, agile brought about a fundamental shift in the SDLC.

In agile development, cross-functional teams collaborate to define the project requirements and solutions. These teams are self-organized and work in a collaborative manner to deliver value incrementally.

Agile methodologies like [Scrum](#) and [kanban](#) prioritize continuous improvement, customer input and swift adaptation to change. These approaches not only enhance developer productivity but also ensure that software solutions are in line with customer requirements and market dynamics. By adopting these methodologies, organizations can foster a culture of innovation and responsiveness, ultimately delivering greater value to their customers.

The next important milestone was [DevOps](#), a methodology that brought collaboration between development and operations teams. One of the key factors that contributed to the rise of DevOps was the advent of programmable infrastructure based on virtualization and cloud computing. This encouraged administrators to use scripts to provision and configure infrastructure (known as [Infrastructure as Code](#)) while exposing developers to the environment where the code is ultimately deployed.

DevOps significantly expanded the agile philosophy beyond development to incorporate operations, fostering a culture of collaboration and shared accountability. By automating the software release process, DevOps introduced practices like [continuous integration and delivery \(or CI/CD\)](#), continuous testing, continuous deployment and continuous monitoring, enabling faster and more frequent deployment of software.

This integration significantly reduced time to market and enhanced product quality, allowing organizations to ship software more often to respond to market trends and customer demands.

As we continue to evolve the SDLC, the integration of generative AI and coding assistants into a developer's workflow seems like the next logical step. These technologies combine efficiency and innovation to accelerate development while reducing errors. The ability of generative AI to generate full-fledged functional code streamlines processes, whereas coding assistants provide real-time, context-sensitive suggestions and best practices that increase developer speed and productivity.

The potent combination of coding assistants and DevOps principles promises

enhanced productivity, improved code quality and the ability to build better, faster and stronger software solutions. In subsequent sections, we will delve deeper into the impact of generative AI on the SDLC.

Navigating Today's Software Development Challenges

Organizations worldwide face unprecedented challenges that have a significant impact on the productivity and performance of their development teams.

One of the critical issues is the globalization of the workforce and the increasing prevalence of remote work. Although this change opens up access to a larger talent pool, it can make coordination and communication more difficult. Teams that are spread across multiple time zones and cultures often struggle to stay in sync, leading to unwanted delays and communication gaps.

Reliable communication tools and practices can facilitate seamless collaboration, reducing friction caused by time zone differences and remote locations.

As digital solutions continue to evolve, businesses are under tremendous pressure to consistently innovate and deliver at a rapid pace. The constant push for rapid development and deployment poses a significant risk of [developer burnout](#), ultimately leading to a potential compromise in software quality. Development teams must walk a fine line every day between the need for speed and an uncompromised commitment to producing quality code.

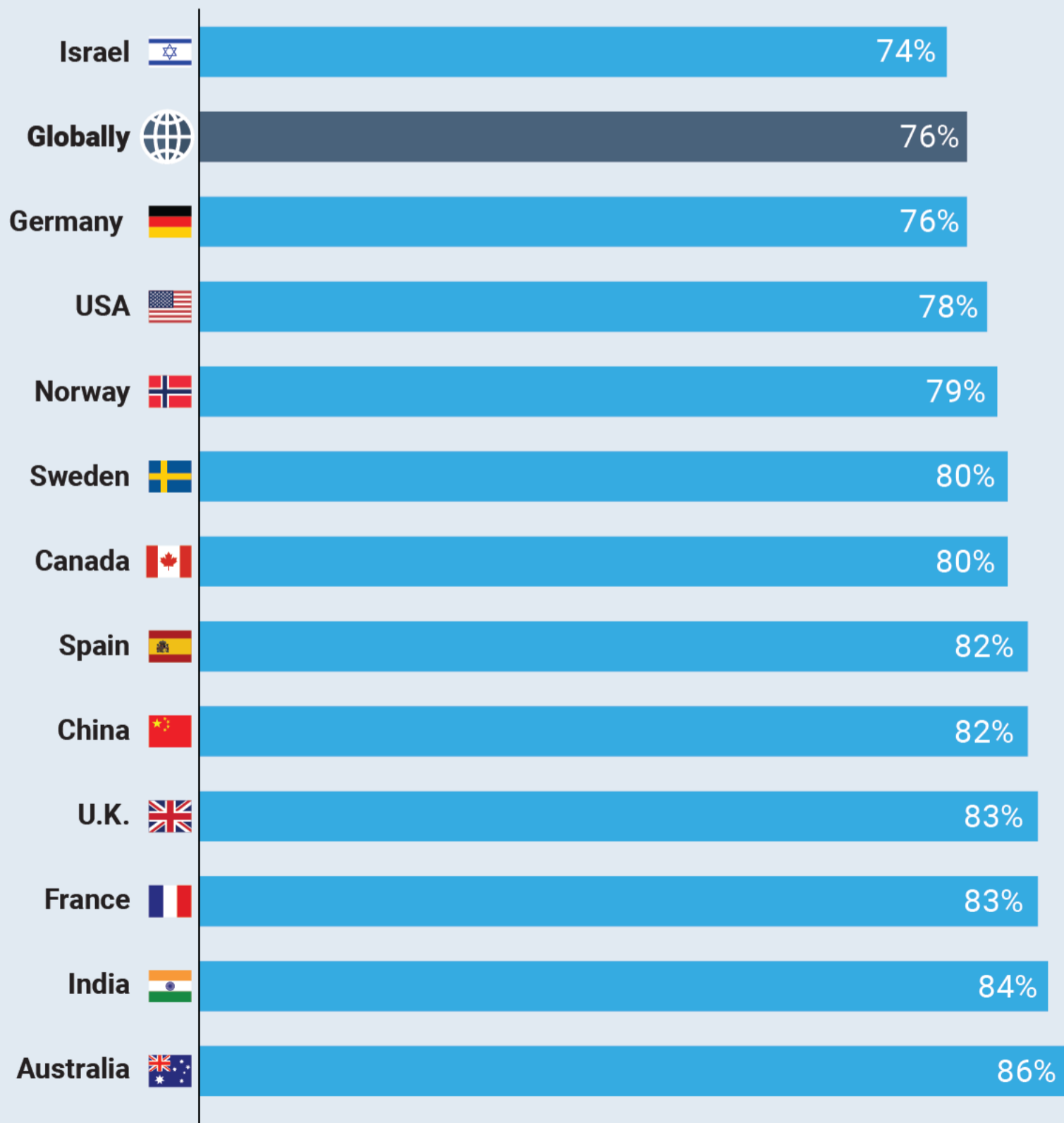
The acute shortage of talent compared to the demand for experienced developers across the world only compounds these issues, and the talent crunch is likely to get worse as birth rates fall and baby boomers and Gen Xers retire.

By 2025, the world will be 4 million people short of the talent it needs to fill available jobs, according to the International Data Corporation (IDC). The potential in unrealized revenue by that time, according to Korn Ferry, [could reach \\$8.5 trillion](#).

Figure 1.1 As the demand for digital services increases, there simply aren't enough experienced developers to fill open positions.

A Global Tech Talent Shortage

There aren't enough developers to fill the demand for their skills. Here's the percentage of employers in each country that reported difficulty in filling IT and tech roles in 2023.



This scarcity coincides with an ever-increasing demand for digital transformation and modern software solutions. Consequently, there is now a fiercely competitive market for qualified developers. In order to increase their talent pool, [businesses are spending more on training and development](#).

Inherently, technology's fast-moving evolution poses substantial challenges for developers. They must navigate the dynamic maze of programming languages, platforms, frameworks and tools. Staying current with the latest innovations while maintaining existing systems means continuous learning and navigating ever-expanding complexity.

In addition, development teams are increasingly confronted with security and compliance challenges.

The integration of software into various aspects of work and personal life increases the risk of security breaches and data privacy violations. To ensure compliance with evolving international regulations and standards, developers must now prioritize robust security measures from the outset. The paradigm shift toward a “security-first” approach (also known as “[shift left](#),” delegating more responsibility for security to the developer) increases the need for resources and skills, making current developers' challenges more severe.

Each of these factors, from globalization and market pressures to talent shortages, technological complexity and security concerns, shapes the current state of software development. They bring new challenges that demand innovative solutions and a fundamental transformation of traditional development practices in order to deliver high-quality, secure software in a rapidly changing world.

How SDLC Challenges Hold Businesses Back

When organizations can't hire enough experienced developers, or see their developers' productivity slowed by complexity, compliance and security demands, or by a development environment that makes collaboration and creativity difficult, business suffers. These issues have a direct impact on the company's operational efficiency, market competitiveness and overall financial success.

The cost of software development is rapidly increasing for a number of reasons. First, teams are distributed across the globe, which complicates communication and increases the expense of collaboration. Second, there is intense competition for the limited pool of highly skilled developers, leading to increased costs. To remain competitive, companies must not only pay their devs handsomely but invest heavily in communication tools, training and readiness programs.

Despite these efforts, it's tough to keep pace. In the rush to meet tight deadlines and stay ahead in a competitive race, the quality of the final product can suffer. Software plagued with bugs and usability issues can damage a company's reputation. In severe cases, it can also lead to legal and compliance issues, especially in industries where software reliability and security are paramount. This decline in quality and the ensuing customer dissatisfaction can have a long-term negative impact on the brand and its market position.

Security concerns add another layer of business implications. A single security breach can have catastrophic consequences, including financial losses, legal liabilities and irreparable damage to a company's reputation. Companies that fail to prioritize security in their SDLC may risk not only data breaches but also potential noncompliance with increasingly stringent regulatory standards.

Finally, the intense pressure to deliver quickly and the complexity of the technological landscape can lead to employee burnout. This pressure affects productivity and can lead to high turnover rates. Recruiting and training new staff is a costly and time-consuming process that can delay projects and strain already tight resources. A demotivated workforce may stifle innovation, a key driver of success in the technology sector.


The convergence of these challenges underscores the urgent need for a paradigm shift in how software is developed and managed. The challenges faced by developers today are not just technical problems; they are deeply intertwined with business outcomes. Addressing these challenges is crucial for companies to remain viable, competitive and profitable in an increasingly digital world. The need for change is not just a matter of staying current with technological trends: It's also a mandate for sustainable business success.

Generative AI presents a transformative solution to contemporary software development challenges. GenAI's ability to automate coding tasks and generate documentation significantly reduces development time, alleviating talent shortages and minimizing burnout among developers. This new technology expedites time to market and addresses security concerns efficiently.

As we venture further, we'll explore how generative AI streamlines processes, and why its development marks a paradigm shift in the SDLC.

CHAPTER 02

Transforming the SDLC With Generative AI

 In this chapter, we explore the transformative impact of generative AI on the software development life cycle, focusing on AI-driven tools. These AI assistants, which are quickly becoming an expected part of every integrated development environment (IDE), are completely changing the way developers write, evaluate and manage code.

By offering real-time suggestions, inline code completion and chat support, they not only enhance coding efficiency but also redefine traditional approaches to problem solving in software development. Here, we investigate their capabilities, their implications and the future of AI-assisted coding, with an emphasis on how these tools transform the SDLC.

How AI Coding Assistants Empower Developers

In the complex world of software development, the adage, “A worker is only as good as their tools,” holds great significance. The IDE is at the heart of a developer’s arsenal, serving as more than just a tool in their coding endeavors. The efficiency,

capabilities and intuitiveness of an IDE are critical factors that influence a developer's productivity and creativity, as well as the quality of the final product.

Selecting the right IDE is not a simple matter of choice but rather a crucial element in the process of translating ideas into code.

IDEs have evolved over time to meet the ever-increasing complexity of software development and testing. Developers can use open source IDEs like Eclipse and Visual Studio Code, as well as commercial products like JetBrains.

One of the most recent additions to these already powerful tools has been the tight integration of AI coding assistants. Products such as Duet AI, GitHub Copilot and Tabnine are available as plugins for and extensions to mainstream IDEs, making them easily accessible to the developer community. Once installed, they empower developers with the following capabilities:

AI-driven code completion: AI-driven code completion tools predict and suggest contextually relevant code snippets as developers type. These tools use large language models (LLMs) trained on vast codebases, enhancing coding efficiency and reducing the likelihood of syntactical errors. They adapt to individual coding styles and project-specific patterns, streamlining the development process and enabling developers to focus on more complex and creative aspects of programming.

Natural language code generation: Natural language code generation allows developers to transform plain language descriptions into functional code, bridging the gap between conceptualization and implementation.

This technology leverages advanced AI models to interpret human language, thereby enabling developers to generate code snippets, algorithms or even entire modules

from simple descriptions. The flexibility provided by this technology democratizes programming, making it more accessible to those with limited coding expertise and accelerating the development cycle.

Code explanation: AI coding assistants with a chat interface provide developers with insights into complex codebases, explaining the functionality and logic of existing code in understandable terms. This is particularly beneficial for understanding legacy systems or any code created by other developers. By offering detailed explanations and clarifying intricate code segments, these tools aid in faster onboarding, better maintenance and enhanced collaboration among developers with varying levels of expertise.

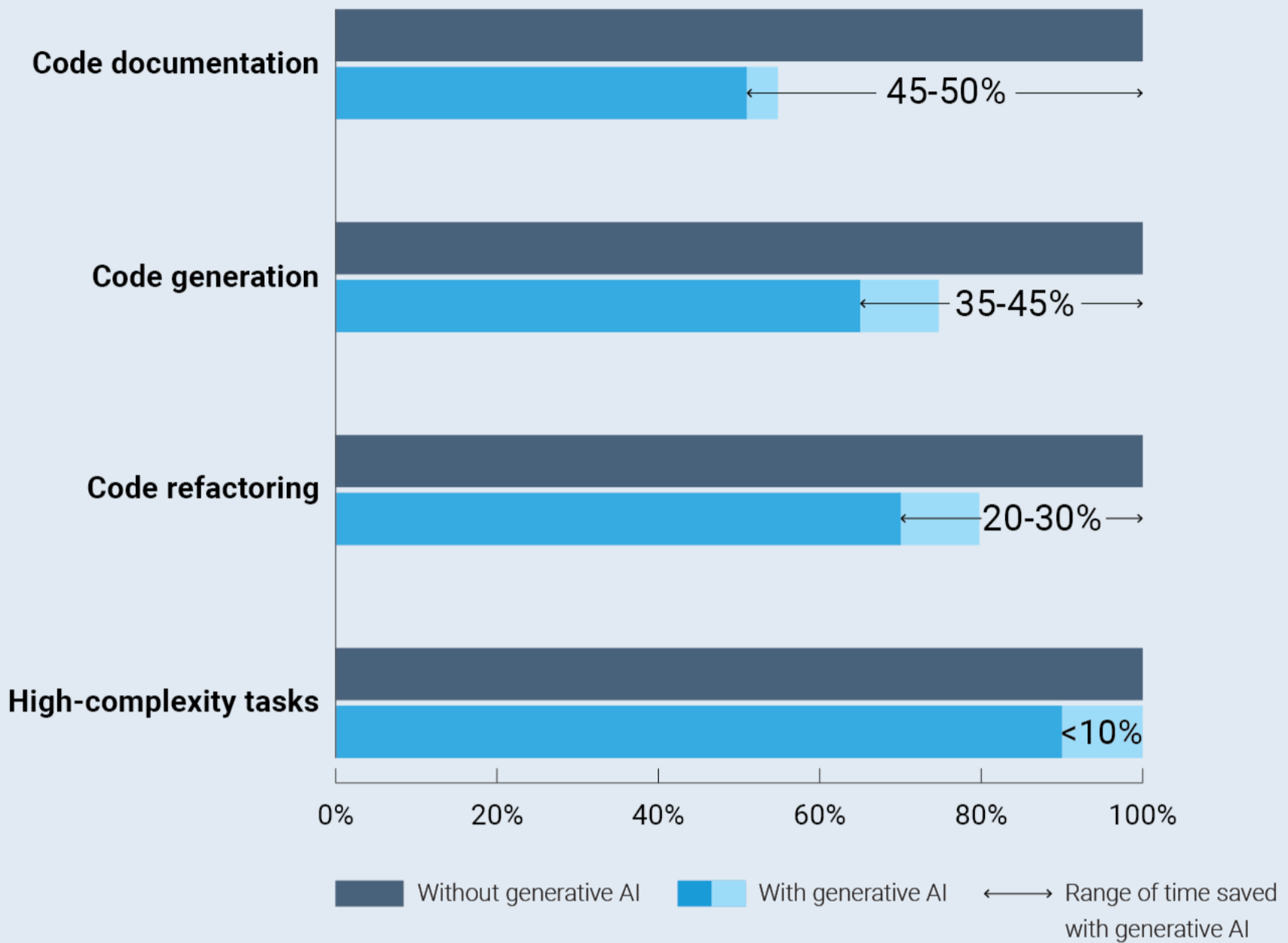
Unit test generation: AI in [unit test](#) generation automates the creation of test cases, ensuring that individual units of source code function as intended. By analyzing the codebase, AI algorithms can generate comprehensive and relevant test scenarios, reducing the manual effort involved in testing and helping developers identify potential issues early in the development cycle. This leads to more robust and reliable software, as well as a more efficient testing process.

AI-generated documentation: AI coding assistants can automatically [create and update technical documentation](#) for software projects. Using natural language processing and interpreting the context of the code, AI generates clear, concise and up-to-date documentation.

This saves time and effort for developers, and ensures that the documentation remains consistent with the codebase, facilitating better understanding and easier maintenance of the software over time.

How Generative AI Saves Developers Time

Percentages show how much less time is needed to complete these tasks:



Source: "Unleashing developer productivity with generative AI," McKinsey Digital.

© 2024 THE NEW STACK

Figure 2.1 Generative AI-based development tools can drastically speed up the completion of more mundane developer tasks, though the gains are smaller for more complex activities, according to a 2023 study by McKinsey Digital.

AI coding assistants revolutionize the developer experience, offering intelligent code completion, code generation based on comments written in natural language, and insightful explanations. These assistants streamline unit testing and maintain up-to-date documentation, significantly enhancing efficiency, reducing errors and enabling developers to focus on creative problem solving, thereby accelerating the software development life cycle.

AI Assistants for DevOps and Data Science

AI coding assistants, particularly those with advanced chat capabilities, also play a role in DevOps and data science. These assistants are equipped with sophisticated algorithms trained on relevant datasets. They are not just coding tools but intelligent partners capable of interpreting and executing complex tasks related to infrastructure management and data science operations.

AI coding assistants with chat capabilities significantly streamline infrastructure-related tasks. They can interpret and execute commands for automating deployment processes, configuring cloud environments and managing [containers](#) and [microservices](#).

For instance, a DevOps engineer could interact with the assistant using natural language to set up a Docker container or orchestrate Kubernetes clusters. This reduces the need for extensive scripting and manual intervention, allowing for more efficient and error-free infrastructure management.

Given that the scriptwriting process for configurations is prone to errors, chatbots expedite this process. They generate customizable templates, thereby streamlining the overall procedure.

These assistants also play a crucial role in CI/CD pipelines. They can assist in writing and maintaining scripts for automated testing and deployment, and can even suggest optimizations in the pipeline based on the latest DevOps practices.

Assistants that have been integrated into various tools and platforms provide a unified interface for handling complex DevOps workflows, thus bridging the gap between code development and operational implementation.

In the realm of data science, AI coding assistants with chat functionalities are transforming the way data scientists work with code. These assistants can generate data-specific code, including data cleaning, transformation, visualization and machine learning model development, simply from a description of the desired outcome. This capability is particularly valuable for data scientists who may not be proficient in specific programming languages or libraries.

For example, a data scientist can describe the need to visualize a certain trend in a dataset, and the AI assistant can promptly generate the appropriate Python code using libraries like [Matplotlib](#) or [pandas](#). This not only speeds up the coding process but also ensures that the code is optimized and adheres to best practices.

Additionally, these assistants can provide real-time suggestions for improving the performance of machine learning models, optimizing hyperparameters and even recommending alternative algorithms based on the data characteristics.

AI coding assistants with chat capabilities are extending their utility beyond traditional software development, becoming indispensable tools for DevOps and data science professionals. By handling a range of tasks, from infrastructure setup to complex data analysis, they enhance efficiency and accuracy. Most importantly, they enable professionals in these fields to focus on strategic and creative aspects of their work — the things most likely to help their organizations seize business opportunities and better serve their customers.

The Impact of AI Coding Assistants on Teams

AI coding assistants have a significant impact on the teams within an organization, streamlining engineering processes.

First, these tools democratize coding skills within a team. By translating natural language into code and explaining complex code segments in simpler terms, these tools make programming more accessible to team members with varying levels of expertise.

This inclusivity not only enriches the team with diverse perspectives but also facilitates smoother onboarding of new members and cross-training among different roles. As a result, teams become more versatile and resilient, capable of tackling a wider range of projects with greater efficiency.

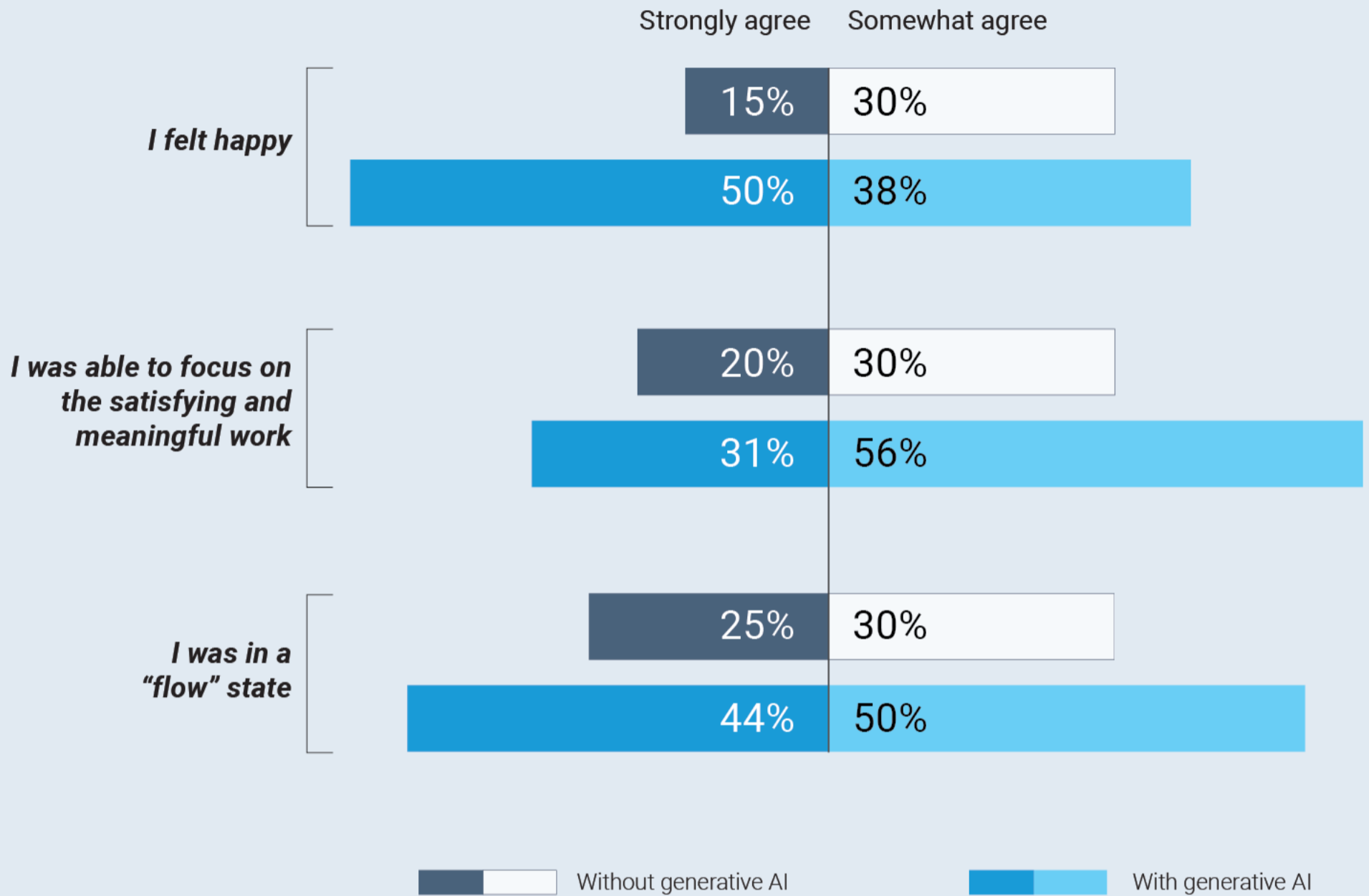
By providing clear, understandable explanations of complex code segments, the tools act as virtual mentors, bridging knowledge gaps and ensuring that all team members, regardless of their experience level, can understand and contribute to the codebase. This accelerates the learning curve for newcomers and ensures a more seamless transition of knowledge when team members rotate or transition in or out of a team, thereby maintaining continuity and efficiency in development projects. Cross-functional collaboration is made easier.

Furthermore, AI coding assistants facilitate a culture of continuous learning and improvement. They serve as a constant source of knowledge and guidance, keeping the team updated with the latest coding standards and practices.

This aspect is particularly beneficial for leaders aiming to foster a culture of innovation and excellence within their teams. By integrating AI tools into the development process, leaders can encourage their teams to embrace new technologies and methodologies, thereby staying ahead in the rapidly evolving tech landscape.

Generative AI Can Improve the Developer Experience

Agreement with statement:



Source: "Unleashing developer productivity with generative AI," McKinsey Digital.

© 2024 THE NEW STACK

Figure 2.2 Developers who use generative AI tools reported that coding assistants improved their ability to focus on coding and also their happiness with their work, according to a 2023 study by McKinsey Digital.

The impact of AI coding assistants extends beyond mere operational efficiency. They also play a crucial role in shaping work culture and team morale. The reduction in mundane tasks and the availability of instant assistance can significantly reduce developer burnout and increase job satisfaction. Teams that feel supported and empowered by technology are more likely to be engaged, motivated and committed to their work, leading to higher retention rates and a stronger sense of community.

How Businesses Benefit From AI Coding Assistants

AI coding assistants offer significant advantages to businesses, particularly in ensuring strict license compliance. These tools automatically monitor and enforce adherence to various software licenses, including [open source agreements](#). This is crucial for avoiding legal issues and financial penalties associated with license violations. By integrating these assistants, businesses can ensure ethical software development and protect themselves from potential legal challenges.

In terms of security and privacy, AI coding assistants equipped with enterprise-grade capabilities are invaluable. They identify and address security vulnerabilities within the code, significantly reducing the risk of data breaches.

Finally, the availability of fully private models in AI coding assistants is a game changer for businesses. This feature ensures that all proprietary data, algorithms and coding practices remain confidential and exclusive to the organization. It's a key advantage for companies looking to leverage AI in their coding processes without compromising their operational secrecy or losing their competitive advantage by exposing their proprietary technologies and methodologies.

If your organization's code is of sufficient quality and volume, it can be added to the training data for a fully private model, such as Tabnine's, resulting in a custom model tailored to each engineering team. After such fine tuning, the quality and relevance of the content that is generated would be significantly improved.

AI coding assistants offer an array of advantages that benefit individual developers, operators, teams and organizations as a whole. For developers, they simplify and expedite coding processes; in DevOps, they streamline and secure infrastructure

management. Teams benefit from improved collaboration and knowledge transfer, and businesses gain from enhanced compliance, security and intellectual property protection. These tools not only optimize existing processes but also open new avenues for innovation and efficiency.

However, as we move forward, it's crucial to explore the other side of the coin: the risks associated with generative AI. Despite their numerous advantages, these technologies also bring challenges and potential pitfalls that necessitate careful consideration.

From ethical dilemmas to technical limitations, the following chapter will delve into the complexities in and implications of relying on AI in your organization, ensuring a well-rounded understanding of this transformative technology.

CHAPTER 03

The Risks Associated With AI Coding Assistants

In the preceding chapter, we explored the numerous advantages of incorporating AI coding assistants. But every technology also carries risks associated with its usage. These risks encompass the potential exposure of internal code, legal liabilities arising from the use of copyrighted code and the ramifications of employing unverified datasets.

Assessing the Key Risks Organizations Face

In early 2023, [Samsung Electronics faced a significant security incident](#) when its developers used the publicly available AI chatbot ChatGPT to generate code, leading to an accidental leak of sensitive internal source code. Concerned about the potential for sensitive data to be stored on servers owned by companies like Google and Microsoft, Samsung issued a memo banning the use of ChatGPT and other AI-powered chatbots by its employees.

The incident highlighted the risks of sharing proprietary information with AI services that store user interactions for model-training purposes. Samsung's response included considering disciplinary action for employees who continued to use ChatGPT and developing in-house tools to replace common uses of the chatbot, such as translation and code debugging.

Non-private generative AI models, while powerful, come with inherent risks, particularly when integrated into AI coding assistants. The primary concern revolves around the unintentional exposure of an organization's code and intellectual property to external entities, including competitors and other companies. Here are some key aspects to consider:

Code leakage: When using non-private generative AI models, the code generated by these models could inadvertently contain elements of your organization's proprietary code. This can happen when the model's training data includes publicly available code repositories or datasets that contain fragments of your codebase. Code leakage can lead to a loss of competitive advantage and potential legal issues.

Intellectual property exposure: In addition to code leakage, non-private generative AI models may expose an organization's intellectual property. This could include algorithms, trade secrets or proprietary methodologies used in software development. If these aspects of your intellectual property are embedded in code generated by AI coding assistants, they may be exposed to others who have access to the generated code.

Data privacy compliance: Nonprivate generative AI models that process and generate code using sensitive data must adhere to strict data protection standards. Violations of data privacy laws can result in severe penalties and reputational

damage. Organizations should conduct thorough assessments to confirm that their AI coding assistants comply with data privacy regulations and safeguard user and organizational data effectively.

Competitive disadvantage: Exposing your organization's code or intellectual property to other companies, intentionally or unintentionally, can put you at a significant competitive disadvantage. Competitors may gain insights into your development processes, technologies or strategies, potentially giving them an edge in the market.

Understanding the Legal Implications

It has become increasingly common across various industries to use machine learning models trained on unverified datasets. Some AI coding assistants are based on models trained on multiple unverified datasets available in the public domain. While these models offer the promise of automation and speed, they also raise significant legal concerns that organizations must carefully navigate. Here are some legal implications of using unverified data in training models:

Intellectual property infringement: One of the foremost legal concerns is the risk of intellectual property infringement. Unverified datasets may contain copyrighted material, proprietary information or confidential data sourced without proper authorization. When these datasets are used for training large language models, organizations run the risk of unintentionally infringing upon intellectual property rights. Such infringements can lead to costly lawsuits, damages and reputational harm.

[GitHub Copilot, according to the company](#), is based on OpenAI's model [Codex](#), which

is trained on publicly accessible sources regardless of license. Tabnine reports that its AI model is trained exclusively on source code with permissive licenses.

Data privacy violations: Data privacy laws and regulations, such as the European Union's GDPR and the California Consumer Privacy Act, require organizations to handle personal data with the utmost care. Unverified datasets may contain sensitive personal information acquired without proper consent or compliance with data protection laws.

Using such datasets can result in severe violations of data privacy regulations, leading to fines and legal penalties. Organizations must ensure that their data sources align with applicable data privacy laws to avoid legal consequences.

Bias and discrimination: Models trained on unverified datasets may [inherit biases present in the data](#), which can result in discriminatory outcomes. This raises ethical and legal concerns related to fairness and equal treatment. Discriminatory AI models can lead to accusations of discrimination, bias and potential lawsuits. Organizations must actively work to mitigate bias in their models and ensure that they adhere to anti-discrimination laws and regulations.

Accountability and liability: Determining accountability and liability in cases where unverified dataset-trained models cause harm or errors can be complex. Legal disputes may arise concerning who is responsible for the model's behavior and any resulting damages. It is essential to establish clear accountability mechanisms and legal frameworks for addressing model-related issues to protect organizations from potential legal liability.

The Rise of Shadow AI in Enterprises

“Shadow AI” refers to the use or development of AI systems, solutions and services within an organization without explicit organizational approval or oversight. It is a form of “shadow IT,” where employees use technology outside the control of the IT department. However, shadow AI presents unique challenges due to the nature of AI and its potential impact on data security, compliance and privacy.



SUMMARY

Tabnine is the AI that you control — helping development teams of every size use AI to accelerate and simplify the software development process without sacrificing privacy, security or compliance.

KEY FEATURES

Tabnine boosts engineering velocity, code quality and developer happiness by automating the coding workflow through AI tools customized to your team.

Tabnine supports more than 1 million developers across companies in every industry.

The risks associated with shadow AI are multifaceted, encompassing various critical concerns. First and foremost, data security becomes a significant issue, as unauthorized AI tools may not adhere to the organization’s security protocols, thereby increasing the risk of potential data breaches.

Moreover, shadow AI poses a threat to compliance with regulatory requirements, especially when handling sensitive or regulated data. This noncompliance can result in severe legal and reputational consequences for the enterprise.

Another concern lies in data quality and bias, as unvetted datasets used in shadow AI can lead to biased algorithms and skewed decision-making processes.

When AI is implemented without careful planning, problems with integration and interoperability can make it hard for existing systems and infrastructure to work together smoothly. These risks underscore the critical importance of addressing and managing AI within enterprises, before “shadow” workflows emerge.

Personal projects are often a way that shadow AI (and for that matter, shadow IT) emerges in an organization. If generative AI tools and platforms are used without formal governance, legal and ethical dilemmas can rear their heads. Lawyers may need to intervene to address issues such as copyright violations or compliance risks.

Using AI technology without oversight in an enterprise means an increased likelihood of generating poor-quality, unreliable and unsecured code. Furthermore, intellectual property challenges can emerge due to AI-generated code that infringes on copyrights or patents. These issues emphasize the need for responsible and ethical deployment of AI in coding projects, with legal oversight playing a pivotal role in mitigating risks and ensuring compliance.

In the next chapter, we will look at some best practices for adopting and implementing AI coding assistants in organizations.

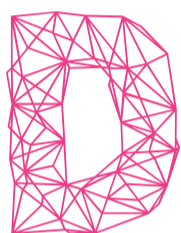
How To Choose an AI Coding Assistant Platform: Checklist

Enterprise leaders face an intricate decision-making process when selecting an AI coding assistant. Each tool or platform offers its own set of unique features and capabilities. This section identifies the crucial factors and considerations to consider when selecting an AI coding assistant.

- Compliance management:** Ensure the platform adheres to industry best practices for security, reliability, compliance and resilience.
- Privacy control:** Confirm that the platform guarantees code privacy, never stores or shares your code with third parties, and does not train on your code unless you opt for a private custom model.
- AI optimization:** Look for a platform that offers AI models trained on credible open source licenses with permissive terms and the ability to optimize these models for your specific use case.
- On-premises deployment:** The platform should offer the flexibility to run on premises or in a virtual private cloud, allowing full control over data and infrastructure.
- Open source code training:** Verify that the AI assistant is exclusively trained on permissively licensed open source repositories, ensuring compliance with open source licenses.
- Custom AI models:** Check if the platform can train custom AI models on your private codebase, ensuring personalized and relevant code suggestions.

CHAPTER 04

Safe Deployment of AI Assistants



Developers are embracing AI coding assistants for several very good reasons: they can automate tedious tasks, provide intelligent code snippets and reduce development time. IT decision-makers are excited about the technology's potential to accelerate developer productivity and generate benefits in customer service and revenue.

However, as discussed in the previous chapter, the use of AI presents new challenges and risks, particularly in terms of security and code quality. This chapter delves into best practices for the secure deployment of AI coding assistants.

Integrate AI Coding Assistants With DevSecOps

[DevSecOps](#) can enforce ethical coding practices while also ensuring the quality and reliability of AI-generated code. It can also tailor code reviews to address the specific challenges presented by AI-generated code, as well as provide continuous monitoring and logging to track the behavior of such code once it's deployed.

Integrating AI coding assistants into DevSecOps is a strategic step toward increasing development agility while maintaining strong security practices. These assistants

can seamlessly integrate into the DevSecOps pipeline, beginning with their inclusion in CI/CD workflows. During the build and deployment phases, they automatically suggest improvements and identify security vulnerabilities by analyzing code commits in real time.

Incorporating AI models for automated code security scanning is another critical step. These models can identify potential vulnerabilities and compliance violations, enabling proactive remediation before code reaches production. They can also perform automated code reviews alongside human reviewers, which ensures code quality, adherence to security standards and accelerated development cycles.

To make this integration effective, it's essential that the model providers continuously train their code generation models with historical security data regarding evolving threats. This practice improves their ability to identify emerging vulnerabilities. Furthermore, maintaining a feedback loop between the development, security and operations teams ensures that AI coding assistants adapt to the organization's specific DevSecOps requirements and adhere to the desired security policies.

Incorporating AI coding assistants into the DevSecOps workflow enables organizations to strike a fine balance between rapid development and robust security, lowering risks and improving overall software quality.

Prioritize Privacy and Data Security

Privacy and data security are paramount concerns when integrating AI code assistants into the development process. These assistants often require sharing code snippets or even entire projects with third-party services, which can raise

significant apprehension about the privacy and security of sensitive code.

To mitigate these concerns, access-control mechanisms should be put in place to ensure that only authorized individuals or systems can reach and interact with the shared codebase. This helps in preventing unauthorized access or data breaches.

Furthermore, organizations must ensure that customer code or training data is not made available to the solution provider. This can be achieved through strict data usage agreements and by thoroughly vetting the privacy policies of AI coding assistant providers. By prioritizing privacy and data security, organizations can confidently leverage the benefits of AI code assistants while safeguarding their sensitive information.

Choose Private and Isolated Code-Generation Models

Organizations must prioritize privacy and data security by choosing AI coding tool providers that offer private deployments. These providers offer on-premises solutions, allowing enterprises to deploy the models within their own infrastructure.

This approach eliminates the need to share sensitive code externally, ensuring complete control over the data. It also addresses concerns related to data sovereignty and compliance with regional regulations by allowing organizations to keep data within their chosen geographical boundaries or data centers.

Strong encryption should be used for data transmission between the organization's environment and the private deployment of the AI coding assistant. Organizations should have robust [access controls and authentication mechanisms](#) in place that allow them to define and manage who has access to the private deployment. This includes [role-based access controls](#), which limit permissions. Detailed audit trails

should also be maintained, allowing organizations to track user activity and ensure accountability.

It is critical to update the private deployment environment on a regular basis with security patches and to follow relevant data protection and privacy regulations, such as GDPR or the Health Insurance Portability and Accountability Act (HIPAA). Secure configuration with measures such as firewalls and intrusion-detection systems enhances data security. Developers must be trained to use technology responsibly, including privacy and security best practices.

Organizations can confidently leverage the benefits of AI coding assistants while maintaining strict control over sensitive code and data by choosing AI coding tool providers that offer private deployments and implementing these comprehensive measures. This approach not only protects privacy and data security, but also supports the organization's compliance and risk-mitigation goals.

Ensure Compliance With Open Source Licenses

AI coding assistants that are integrated into your organization's SDLC must adhere to open source licensing to ensure the ethical and legal use of open source code.

The providers of AI coding tools should have robust mechanisms in place to track and manage the use of open source code. This includes maintaining a comprehensive database of open source libraries and their associated licenses and actively monitoring the code snippets provided by the AI assistant to ensure compliance.

Currently, you have two options to ensure that you're in compliance with licensing: Either use AI coding assistants that are exclusively trained on permissively licensed code, or use one that reports where the code it generates came from so you can check

its license yourself. The latter is a more cumbersome and less reliable process.

By using AI models trained solely on codebases with permissive licenses, organizations can significantly reduce the risk of inadvertently incorporating code with restrictive licenses into their projects.

These AI models are designed to understand and generate code in alignment with permissive licenses, such as the [MIT License](#) or [Apache License](#). As a result, they provide code suggestions that inherently comply with the license terms, minimizing the likelihood of legal complications related to code usage.

Choosing AI coding assistants trained on permissively licensed code is a strategy that aligns with both ethical and legal standards. It ensures that the code recommendations and snippets provided are inherently compliant with open source licenses, eliminating the need for extensive manual review and reducing the risk of license violations.

Conduct Stringent Code Reviews

Regularly reviewing AI-generated code is a fundamental practice for ensuring that the benefits of AI are harnessed effectively while mitigating potential risks. These reviews encompass several crucial aspects, including best practices, security and maintainability.

Scrutinizing AI-generated code for best practices is essential to maintaining coding standards and consistency within the organization. This step involves assessing whether the code aligns with established guidelines, adheres to coding conventions and follows architectural principles. By adhering to these standards, organizations can avoid codebase fragmentation and maintain a cohesive and comprehensible codebase.

Security reviews are paramount in identifying and rectifying vulnerabilities within the AI-generated code. Ensuring that the code doesn't introduce security loopholes or expose sensitive data is crucial. Regular security assessments by internal teams and external auditors help fortify the system against potential threats, keeping the organization's software assets safe from malicious actors.

Evaluating code for maintainability is essential for long-term sustainability.

Assessing factors such as code readability, modularity and ease of debugging ensures that the AI-generated code remains manageable and adaptable over time.

This promotes efficient collaboration among development teams and minimizes the technical debt that can accumulate from poorly maintained code.

Track and Measure Developer Productivity Gains

Measuring the impact of [developer productivity](#) in the context of AI coding assistants is essential for enterprises considering the private deployment of these models. As organizations integrate AI-driven tools to enhance coding efficiency, the identification and quantification of relevant metrics become paramount.

[Key productivity metrics](#) include the reduction in coding errors, time saved in code development and an increase in the number of successful deployments over time. To accurately gauge the impact of AI coding assistants, it's vital to establish a baseline of [developer performance](#) before you start using these tools.

Additionally, your organization will gain more long-term benefits from AI-based tools by measuring the learning curve for developers as they adapt to AI assistants and the subsequent improvement in code quality and complexity handling. It's also important to track the AI tool's ability to understand and execute project-specific

requirements, which affect developer reliance on AI and the tool's overall effectiveness. These metrics can give you a comprehensive view of how the AI coding assistant is enhancing developer efficiency and project outcomes.

Also, evaluate how well these tools integrate with current development platforms and tools, as well as how scalable they are within the development pipeline.

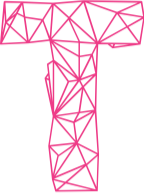
Focusing on these metrics allows organizations to make informed decisions about how and whether to adopt AI coding assistants companywide. Tracking and measuring these metrics ensures that the investment in AI coding assistants is delivering measurable and tangible improvements in overall developer productivity, project timelines and code quality.

To summarize, follow these best practices for integrating AI coding assistants into your SDLC:

- Safeguard privacy and data security through access controls and encryption.
- Adhere to license compliance to prevent copyright risks.
- Pick AI models trained on permissive licenses to eliminate legal exposure.
- Select an AI coding tool built on a model that includes IP indemnification to protect your company from intellectual property liability.
- Perform regular code reviews to ensure coding standards, security and maintainability.
- Identify, track and measure key metrics that define organization-wide developer productivity.

These practices collectively enhance efficiency and quality while mitigating legal and security risks in software development.

Conclusion

 This ebook explored the landscape of generative AI and its profound impact on developer productivity within the context of coding assistants. As we conclude our journey, we must emphasize the need for a balanced approach that harnesses GenAI's benefits while vigilantly mitigating potential risks.

Generative AI represents a paradigm shift for software development. Its ability to automate repetitive coding tasks, suggest improvements and even generate code from natural language instructions has ushered in a new era of efficiency and collaboration. It empowers developers to focus on innovation, problem-solving and creativity, while relegating routine coding to the capable hands of AI.

Throughout this ebook, we've underscored the importance of approaching generative AI with caution and with a deep understanding of the challenges this technology presents. One of these challenges is the potential introduction of security vulnerabilities through AI-generated code. Ensuring the integrity and safety of software should remain paramount, requiring continuous monitoring and refinement of AI coding assistants.

Equally crucial is the risk of overreliance on AI. While AI coding assistants are powerful allies in software development, they should complement — not replace — the fundamental coding skills of developers. Nurturing these skills and viewing AI as a collaborative partner will be vital to maintaining the quality and resilience of our codebases.

Privacy and data security are integral aspects of the AI coding assistant landscape. As we embrace this technology, it is our responsibility to safeguard sensitive data, uphold transparency in its usage and mitigate potential biases. Upholding ethical principles and practices will ensure that GenAI's advantages outweigh its potential risks.

Carry what you've learned about generative AI and the SDLC with optimism and responsibility. The future promises a symbiotic and collaborative relationship between human developers and AI coding assistants — one where innovation and efficiency can thrive.

This ebook has aimed to provide a comprehensive understanding of the intricacies and potentials of generative AI in the realm of coding. May it empower you and your organization to make informed decisions, to leverage AI responsibly and to continue pushing the boundaries of what is possible in software development.

About the Author



Janakiram (Jani) MSV is a practicing architect, research analyst and adviser to Silicon Valley startups. He focuses on the convergence of modern infrastructure powered by Kubernetes and machine intelligence driven by the Internet of Things, edge computing and AI. Before becoming an entrepreneur, he spent over a decade working as a product manager and technology evangelist at Microsoft Corporation and Amazon Web Services. Janakiram regularly writes for Forbes, InfoWorld and The New Stack, covering the latest from the technology industry. He is a keynote speaker for internal sales conferences, product launches and user conferences hosted by technology companies of all sizes.

Disclosure

The following companies mentioned in this ebook are sponsors of The New Stack:
Amazon Web Services, Google, Microsoft and Tabnine.

