Emerging Tech: Generative AI Code Assistants Are Becoming Essential to Developer Experience

Published 11 May 2023 - ID G00790320 - 11 min read

By Analyst(s): Radu Miclaus, Arun Chandrasekaran, Ray Valdes, Mark Driver, Eric Goodness

Initiatives: Emerging Technologies and Trends Impact on Products and Services

Generative AI code generation has the potential to revolutionize software development workflow and the developer experience. Product leaders must capitalize on generative AI that can be used to scale their product development and enhance the experience of the developer offering target persona.

Additional Perspectives

- Invest Implications: Emerging Tech: Generative AI Code Assistants Are Becoming Essential to Developer Experience
(15 May 2023)

Overview

Key Findings

- As the code assistant accelerate the coding tasks, developers become “x” times multipliers through orchestration of software development across multiple languages/frameworks and platforms. This allows the “pair” to scale the development processes.

- Developers are increasingly expecting the tools they use to have augmented generative AI code assistants within the experience.

- Code generation is also assisting in the ability to put together generative workflows, especially for no-code environments that need to string multiple generative coding tasks together.

- The use of AI code generation tools are not replacing the quality assurance (QA) processes and security controls that are needed for robust and secure product development, as well as for mitigation of inherited risks from using generative methods for code.

Recommendations

Product leaders looking to accelerate the entire software development life cycle for their own developers and developer personas should take these steps:

- Work with your engineering leader to explore ways for accelerating the product development and testing cycles by adopting AI code assistants-enabled integrated development environments (IDEs), adding support for multiple development frameworks and languages. The adoption cycle should not shortcut the examination of the legal risks of generative AI tools and should prioritize tools trained on permissive licenses and curated sources.
- Integrate augmented IDE services that allow for an accelerated developer experience within the workflow your platforms provide and the coding platforms you support.
- Expand your target user base by adding the capability to build low-code and no-code workflows within the offerings that enables citizen technologists to prompt the platforms and accelerate their development of components.
- Build awareness of the risks that come with the use of AI code assistants, and plan to mitigate them by selecting vendors that address these risks specifically.

Strategic Planning Assumptions

By 2025, 80% of the product development life cycle will make use of generative AI code generation, with developers acting as validators and orchestrators of back-end and front-end components and integrations.

By 2028, the combination of humans and AI assistants working in tandem could reduce the time to complete coding tasks by 30%.

Analysis

> **"The hottest new programming language is English."**
>
> *— Andrej Karpathy, Twitter thread*

What Is Powering the Generative Code Revolution?

An Evolution of Code Generation Capabilities

Researchers at OpenAI introduced Codex in July 2021, a code-specific AI model based on third-generation Generative Pre-trained Transformer (GPT-3), which is trained against millions of publicly accessible code repositories. Codex was innovative as it used a GPT-3 foundation that was refined against existing codebases. According to ArXiv's Evaluating Large Language Models Trained on Code the output correctness was measured with the HumanEval method to show major improvements of the Codex model over the unrefined GPT-3 model.

A month later, August 2021, the Codex model was made available for practical use for programmers as GitHub Copilot, starting a race in the area of AI-augmented programming. As the GPT framework evolves with GPT-4, Codex itself as a task-specific model may be replaced by the new GPT framework versions evolving as a more versatile and general-purpose model. Other players in the market have released their AI code assistants, built either on the GPT framework or proprietary models and refinement, and we will be discussing some of the players below.

Vendors That Offer Code Assistants Using Generative AI

Several technology providers are focusing on both providing an immersive experience of AI code assistants for developers and making developers more productive. The following are examples of vendors and/or offerings:

- AI Code Reviewer
- aiXcoder
- Amazon CodeWhisperer
- AskCodi
- Bard by Google
- BigCode
- Bito by Metatext
- Blackbox
- ChatGDB by OpenAI
- Codacy

- CodeComplete, currently in private beta
- Codeium by Exafuction
- CodeSnippets.ai
- CodeSquire
- Codewand
- CodeWP
- Codiga
- Debuild
- Duet AI by Google
- GitHub Copilot
- Microsoft's Visual Studio Intellicode
- Mintlify
- MutableAI
- Replit's Ghostwriter
- Safurai
- Salesforce's CodeGen
- Tabnine
- Warp

Examples of programming languages supported by AI code assistants include C#, Go, Java, JavaScript, Perl, PHP, Python, Ruby, Rust, SQL, Swift and TypeScript. Others will likely emerge as the need and popularity shift in the developer communities.

As we will discuss in the three critical insights below, the concept of assistants will extend in the low-code and no-code space in the medium term. Consequently, citizen developers can build more complex workflows based on complementary code frameworks via text or voice prompts as opposed to visual programming.

Developers are experiencing an exciting evolution of how they complete development work through augmentation of their experience with generative code assistants. The following lists some examples of tasks where generative assistants can help developers:

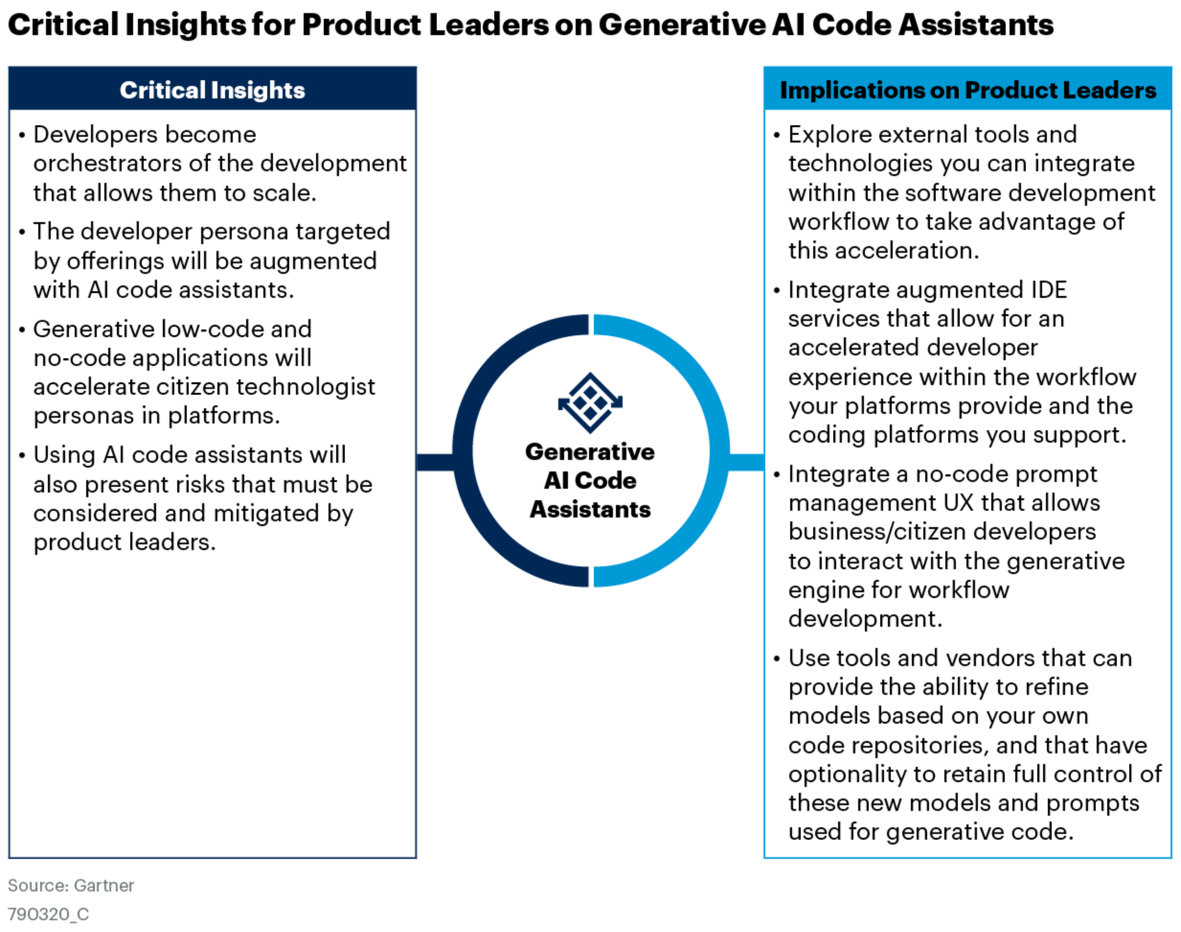
- Generate boilerplate code
- Generate regular expression (regex)
- Rewrite code using correct style/optimal code
- Rewrite code in different languages (code translation)
- Refactor legacy code
- Write code to use an existing third-party API
- Generate service implementation code based on an API specification
- Explain/comment/document the code
- Write test cases
- Check vulnerabilities
- Automate packaging features with security layers for testing and delivery
- Monitor and provide recommendations for improvements as applications mature in usage
- Implement algorithmic logic

The excitement and opportunity in this space are also tempered by the fear of full coder replacement by generative code agents. The likely reality in the short term to medium term is that the developers will likely scale their productivity, and also remain vital as a supervisory and directional role on tasks like quality assurance, security and process design. Developers will likely have to engage in new learning curves to both elevate their skills to higher-level design and orchestrations of systems and workflows and to build smarter systems of automation and generation of features that we will discuss further in this document.

Product leaders that both work with development teams and also possibly target developers in their offerings have to plan for the potential of AI coding assistants both for their own software development tasks as well as the development tasks that the target audience for their offerings engage in. In many ways, these two aspects can become complementary in terms of refinement and feedback loop based on the needs of both internal and targeted external developers.

Figure 1 shows the critical insights for product leaders.

Figure 1: Critical Insights for Product Leaders on Generative AI Code Assistants



Critical Insights for Product Leaders on Generative AI Code Assistants

Critical Insights

- Developers become orchestrators of the development that allows them to scale.
- The developer persona targeted by offerings will be augmented with AI code assistants.
- Generative low-code and no-code applications will accelerate citizen technologist personas in platforms.
- Using AI code assistants will also present risks that must be considered and mitigated by product leaders.

Generative AI Code Assistants

Implications on Product Leaders

- Explore external tools and technologies you can integrate within the software development workflow to take advantage of this acceleration.
- Integrate augmented IDE services that allow for an accelerated developer experience within the workflow your platforms provide and the coding platforms you support.
- Integrate a no-code prompt management UX that allows business/citizen developers to interact with the generative engine for workflow development.
- Use tools and vendors that can provide the ability to refine models based on your own code repositories, and that have optionality to retain full control of these new models and prompts used for generative code.

Source: Gartner
790320_C

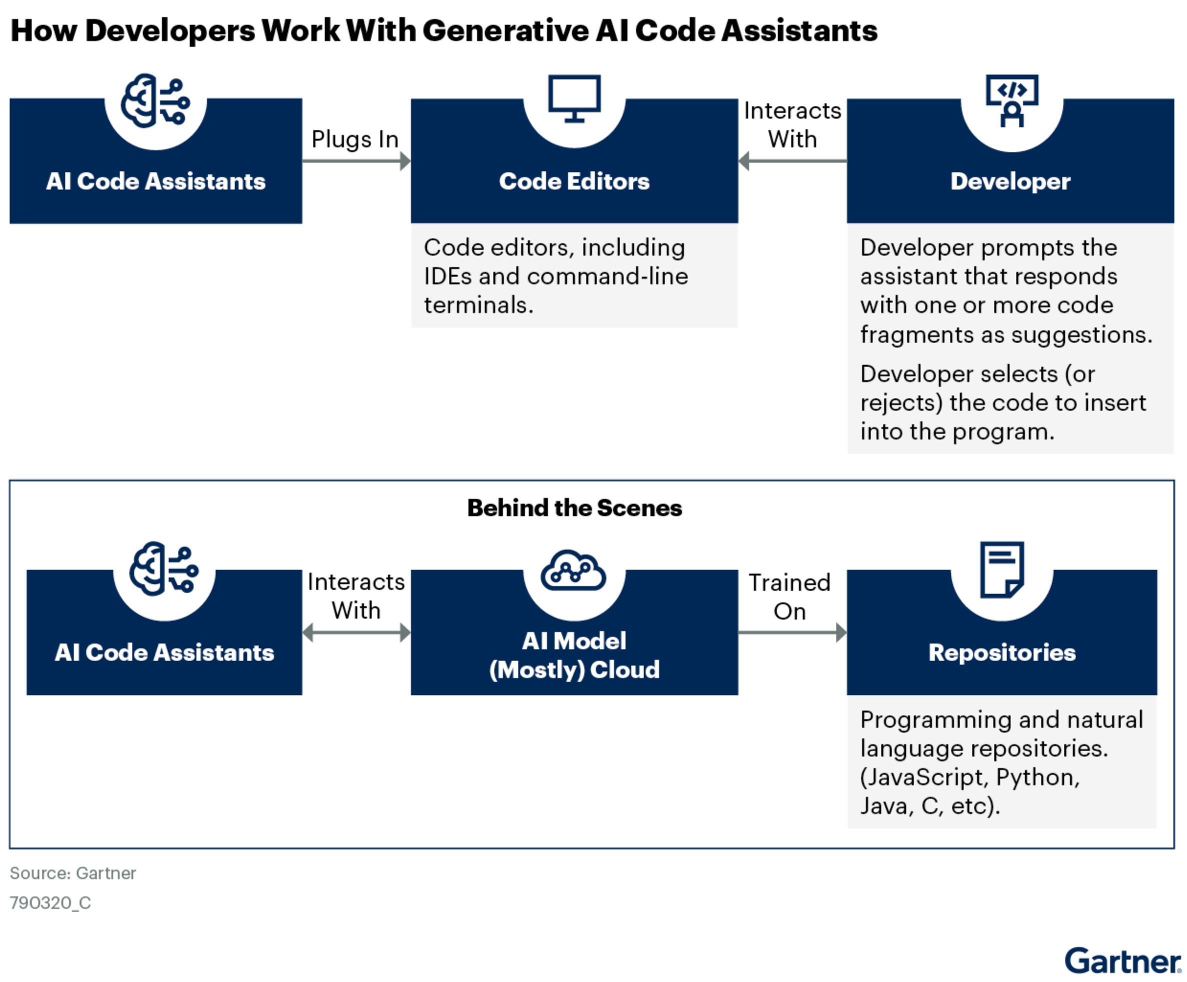
Critical Insight: Developers Become Orchestrators of the Development That Allows Them to Scale

AI code assistants will act as two important drivers for technology and service providers. One vector will be productivity. The engineering teams will be able to scale their productivity and, hence, their ability to iterate and improve features at a faster pace. In the near future, developers will increasingly act as orchestrators of coding tasks, with code assistants completing a large majority of tasks. The second driver is competitive pressure. AI code assistants will considerably lower the barriers to entry in software development, which means new entrants in the competitive space will add to the pressure on innovation pace and margin of existing players.

According to ArXiv's The Impact of AI on Developer Productivity: Evidence from GitHub Copilot, "Recruited software developers were asked to implement an HTTP server in JavaScript as quickly as possible. The treatment group, **with access to the AI pair programmer, completed the task 55.8% faster** than the control group." While scaling for common tasks will be possible in the short term, scaling in custom tasks is still a question mark that time and experimentation will resolve.

Figure 2 shows how developers work with AI code assistants.

Figure 2: How Developers Work With Generative AI Code Assistants


How Developers Work With Generative AI Code Assistants
AI Code Assistants
Plugs In
Code Editors
Code editors, including IDEs and command-line terminals.
Interacts With
Developer
Developer prompts the assistant that responds with one or more code fragments as suggestions.
Developer selects (or rejects) the code to insert into the program.
Behind the Scenes
AI Code Assistants
Interacts With
AI Model (Mostly) Cloud
Trained On
Repositories
Programming and natural language repositories. (JavaScript, Python, Java, C, etc).
Source: Gartner
790320_C
Gartner

Implications

Product teams that do not adopt code assistants within their software life cycle will be left behind in terms of ability to both execute and deliver against the fast-moving competitive landscape.

Recommendations

Product leaders responsible for product teams innovation and productivity should:

- Work with engineering leaders to evaluate external tools and technologies (providers of AI code assistants) that the development teams can integrate within the software development workflow to take advantage of this acceleration.
- Focus on faster minimum viable product (MVP) delivery that focuses on development scalability and maintainability to validate the use cases for generative AI code assistants within their processes.

- Work with engineering leaders to instill a culture of continuous experimentation in the development teams focused not only on both system and process design, but also on task orchestration for AI code assistants.
- Build discipline around documentation, and share the best practices for prompting and designing among internal developers as well as external developer personas for the offerings.

Critical Insight: The Developer Persona Targeted by Offerings Will Be Augmented With AI Code Assistants

When delivering software offerings that target developer personas for either back-end services, front-end components or both, product leaders must capitalize on the changing expectation for what the developer experience needs to be. Augmented IDEs with code assistants will replace basic code editors in any coding and development activity within software offerings, becoming table stakes in the short term.

Examples of offerings targeting both UI users and coders as multipersona applications range across the entire stack — from infrastructure management (infrastructure as code), to data management, analytics and AI development, all the way to business application front-end development. Interactive workflows that include coding tasks are common in the design of offerings for full flexibility, and the coder experience will benefit from the acceleration provided by AI code assistants.

Implications

The developer target personas will expect a superior coding experience in the context of the applications and platforms they use. If the platform offers neither native nor integration options with vetted AI code assistant services, the targeted developer persona either will choose competitors that offer that option or will take their development effort outside of the designated platforms offered.

Recommendations

Product leaders looking to provide a competitive experience for the developer target personas for their software should take these steps:

- Integrate augmented IDE services and support plug-ins that allow for an accelerated developer experience within the workflow your platforms provide and the coding platforms you support.

- Use trials and Freemiums to show the value of the coding environments and how they integrate within the use cases of their platforms.

Critical Insights: Generative Low-Code and No-Code Applications Will Accelerate Business/Citizen Technologist Personas in Platforms

Advancing into generative processes and workflows will be a natural progression from task-based code generation. Process metadata will be the baseline for training and guiding generative processes that orchestrate blocks of generative code tasks. This application of generative AI will fuel the productivity wave for low-code and no-code citizen developers. They will be able to use text-to-process generative assistants that generate processes and workflows with multiple code tasks (likely utilizing multiple coding frameworks).

Examples will be experiences that enable citizen developers to prompt generative assistants through text or voice to design and build full applications that combine both front-end and back-end services. Examples of voice-to-text-to-process are already coming up for building basic functional web applications and will continue to progress in more complex tasks. In ArXiv's ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application, the authors provide examples of "how OpenAI's ChatGPT can be used in a few-shot setting to convert natural language instructions into an executable robot action sequence." The emerging technologies for guided or autonomous generative agents, like Auto-GPT or AgentGPT, will also offer interesting and expanding use cases for process/workflow/application development.

Vendors that will build generative experiences that democratize the creation of simple to complex applications for enterprises, will be able to create and extract value by scaling the business value enterprises can achieve. The quality (capability and sophistication) of the generated processes and individual tasks will evolve, as large language models (LLMs) increase their optionality for refinement on privately held code and metadata repositories that vendors use in the context of the offerings themselves.

Implications

Supporting the developer experience is just the beginning. Supporting the low-code and no-code builder experience will scale the value that vendors can produce and extract from enterprises. The combination of process/workflow and code generation for back-end and front-end services will enable the generation of no-code and low-code experiences for citizen developers and drive adoption and stickiness.

Recommendations

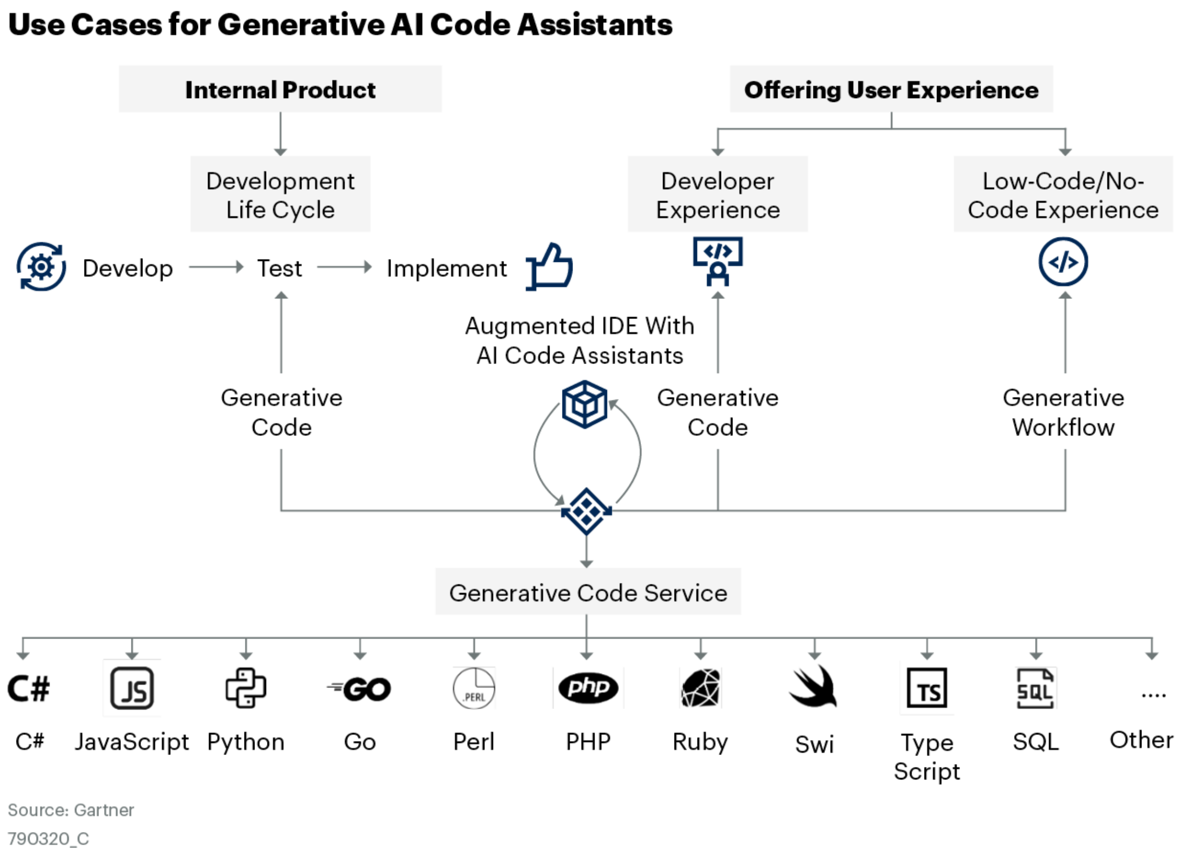
- Integrate a no-code prompt management UX that allows business/citizen technologists (noncoders) to interact with the generative engine for workflow development.

- Activate the process/workflow metadata and code structures that the offerings' platforms are built on in order to be used to refine existing foundational models. This will increase the accuracy of the generative output (code and workflow elements) within the parameters and usage patterns allowed by the offering platforms.

- Start experimenting with emerging guided or autonomous generative agent technology that push a heuristic learning and automation into workflows and sequential tasks.

Figure 3 brings together the three areas of focus for product leaders, including their own product teams, developer target personas for their offerings, and citizen developer target personas for low-code and no-code offerings.

Figure 3: Use Cases for Generative AI Code



Evidence

Evaluating Large Language Models Trained on Code, ArXiv.

The 1000X Developer (a16z podcast with Replit), powered by Simplecast.

OpenAI Codex, OpenAI.

Your AI Pair Programmer (GitHub Copilot), Github.

Amazon CodeWhisperer (AI Code Generator), Amazon Web Services (AWS).

Intro to Ghostwriter, Replit.

Significant-Gratis/Auto-GPT, GitHub.

Bard Now Helps You Code, Google.

ChatGPT Users Report Seeing Other People's Conversation Histories, PCMag.

Meet the Post-AI Developer: More Creative, More Business-Focused, ZDNET.

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Quick Answer: Should Software Engineering Teams Use ChatGPT to Generate Code?

Innovation Insight for Generative AI

Emerging Tech: Generative AI Needs Focus on Accuracy and Veracity to Ensure Widespread B2B Adoption

Quick Answer: How Can You Manage Trust, Risk and Security for ChatGPT Usage in Your Enterprise?

Innovation Insight for ML-Powered Coding Assistants

Quick Answer: Will Machine-Learning-Generated Code Replace Developers?