InformationWeek: Will Generative AI Replace Developers?

Will Generative AI Replace Developers?

Developers continue to work against shrinking time to market demands, and the latest tool in the toolbox is generative AI (GenAI). From code generation to documentation and product marketing, the technology is making positive productivity impacts, albeit less than the current hype suggests. For example, global agricultural and construction company CNH Industrial is achieving a net developer productivity gain of 5% after debugging and security scans, though the benefits don't end there. The company is using generative AI in several ways.

At present, CNH has about 2,000 developers, 10% of which are actively using GenAI. The company launched its first customer-facing application for dealer technicians that allows them to query how to fix vehicles using a cell phone and app as opposed to sifting through 500page manuals and PDF files.

“We were able to go from talking to a dealer to a prototype in 30 days, from prototype to pilot in 75 days and from pilot to launch in 60 days, so all in all it was five and a half months from idea to the first iteration of the product,” says Marc Kermisch, global chief digital and innovation officer at CNH.

Farmers are now able to track their vehicle fleet and the agronomy data produced by those vehicles.

Kermisch's team has been experimenting with GitHub Copilot since summer 2023. So far, the highest utilization has been around traditional web-based technologies, including C#, Java, HTML and SQL. More specifically, they're using it for test case development, code commenting and repeatable procedure development. There's less adoption and efficacy when it comes to embedded systems utilizing C and C++ code.

The team is also using Microsoft Copilot and testing Google Gemini for things like writing job descriptions, press releases and employee-facing information.

Querying Vast Data Sources and More

Data orchestration platform provider Astronomer uses generative AI for code generation and enterprise chat. The company's customers expressed interest in code generation based on a prompt -- a capability that a couple of customers are already using successfully. Typically, those customers create their data pipeline with Python code and then deploy it to the Astronomer platform to run, but they want help writing those pipelines.

Astronomer has also created a chatbot that can be used to answer engineering questions across the company's public documentation page, Slack chat, Stack Overflow and other sources. The code for this chatbot has been open sourced along with the pipelines that feed it, according to Julian LaNeve, chief technology officer at Astronomer.

"I think the big difference came from the last 20%, making sure you're getting the answers right. It's very easy to get the first 80% using a commercial or open source LLMs, but getting that last 20% was really tricky, especially when keeping things up to date," says LaNeve. "Luckily, we're in the data pipeline business, which meant we could just go write pipelines to do it."

There's a reference implementation of that so others can understand how to build a similar enterprise chatbot, including making sure the data is up to date, what the pipeline should look like, what LLM they should use, and how to interact with it, as well as how to monitor response quality and cost.

New Tech, New Benchmarks

Digital services firm West Monroe has been using Github Copilot for arduous and repetitive tasks, and Nigel (a proprietary chat-bot built on top of the ChatGPT engine) for simpler things, such as integrating code with a well-documented third-party API. However, the LLM requires context, such as telling it to generate an integration given that the developer is using the .NET framework and particular libraries.

Sean McHale, a former software developer and partner at West Monroe, is focused on mergers and acquisitions. In that space, private equity firms interested in an acquisition want to understand which tools and technologies the target company is using, including how they're using GenAI in coding.

"We look at R&D as a whole -- R&D benchmarks, the SDLC methodology and how you're using various tools to complement your work," says McHale. "I'd say within the past year we've started seeing more companies using generative AI -- and we're still building out our benchmarks. From the perspective of our in-house developers, if we were suddenly did not have generative AI in development, it would be like going back to the Stone Age."

Internally, the use of generative AI has improved developer efficiency by 15% to 25%, McHale says, though GenAI is not used for all development.

Code Understanding Versus Code Generation

Joe Reeve, a software engineering manager at digital analytics company Amplitude says GenAI is useful for understanding code but warns that developers should be careful about code generation because GenAI can introduce coding errors. For example, when working on a hackathon project that involved collision detection, generative AI wrote about 20 lines of code based on the function name. Since it takes Reeve about 20 minutes to write the code himself, generative AI appeared to be a great time saver.

Two or three hours later, it was clear that the code wasn't detecting collisions properly, so Reeve had to spend three or four hours trying to identify the root case. It turned out that the AI-written code had inserted a less than symbol where a greater than symbol should have been.

"That's a really surprising error to have. Even in my debugging I came to it last because I was like there's no way that function is wrong because it [generated] the whole thing at once," says Reeve.

As one of the first users of Github Copilot, Reeve says the technology allows him to think at a much higher velocity.

“It definitely saves time, but it’s not typing speed. Instead, it allows me to think through higher level chunks [and] skip a lot of detailed thinking,” says Reeve.

Aleksey Didik, head of [the] engineering excellence program at EPAM Systems a global provider of digital engineering, cloud and AI-enabled transformation services, used generative AI to create a system in Python without knowing Python. (He’s actually a Java developer.)

“I wouldn't be able to do it without generative AI in the short term. It all depends on applicability, it all depends on the system,” says Didik. “[W]hen there is a lot of legacy code, the code base itself is very specific to the company, so the results are not as magical or immediate, but still, [generative AI] is a very powerful technology. If the latest updates to Gemini Pro, [introduces] 1 million tokens, it [will] increase the context the LLM can use to understand how to generate proper code.”

EPAM has been using generative AI since it was first introduced and they’re seeing productivity improvements at the individual level, but not so much at the team level. In fact, the company recently published a white paper, based on a client engagement, that explains what organizations should do to succeed with generative AI.

“If you're talking to engineers, the individual productivity and individual performance, yes, there is a boost, but how it converted to the team performance, product performance, company performance, and business performance is much more difficult,” Didik says. “Generative AI can generate elements of code, but you still need to have a whole SDLC [in place] to verify the correctness of the solution. We still need to verify it against the bigger scope of requirements against a more comprehensive set of user stories, for example.”

Despite the rapid adoption of generative AI inside and outside IT, it’s still early days, so organizational expectations should be set accordingly.