THE RISE OF AI AGENTS: RESHAPING SOFTWARE DEVELOPMENT

BY DEEPAK SINGH, VICE PRESIDENT OF NEXT-GENERATION DEVELOPER EXPERIENCE FOR AMAZON WEB SERVICES

From the use of modular components to the well-defined rules and syntax of programming languages, the very way we build applications makes software development an ideal use case for generative AI, and it is no surprise that software development is one of the first areas being transformed by generative AI.

While the industry has made great strides in a short stretch of time, we have only scratched the surface of what is possible with generative AI-powered coding assistants. What started as the ability to simply predict the next line of code is quickly evolving into something entirely new. In the future, AI-powered agents will drive the majority of software development, and we are just starting to see this shift take shape.

However, generative AI is as much a problem of science and technology as it is a problem of human interaction. While agents may take the tedium out of software development, a developer's role is only becoming more vital as they look to orchestrate these agents and bring products to life.

A Quick History of Generative AI-powered Assistants

While generative AI-powered assistants came about recently in the grand scheme of things, we have already experienced several generations of these tools. The first generation of generative AI-powered assistants for software development was autocomplete for code. Many people today have had the experience of writing an email or text only to have an AI system guess the next word they want to type. While these predictions can be a hit or miss for everyday communications, they can be immensely helpful when writing code. These models are trained on a broad set of patterns and encompass a deep understanding of how codes work that exceeds what any one person typically knows. These systems were the first generation of coding assistants, allowing developers to generate lines or entire blocks of code based on what they were typing.

That was a great start, but there was still more to be done. The second generation focused on giving developers the ability to chat directly with the models. Once a developer could access these models in their integrated development environment (IDE), the next logical step was to broaden how they could use them by enabling direct conversation with the models.

Instead of using prompts, developers could pose questions (such as, “How does this section of code work?” or “What is the syntax for declaring a variable in Python?”) and receive responses just like they would if they were talking with a colleague over Slack. This kind of chat-based coding is exciting because it showed the potential of these tools to not only generate code, but reason through a problem on behalf of the user. While the developer still has to drive every interaction and wait for a response, that reasoning capability remains extremely useful and helped set the stage for the shift taking place right now.

The third generation of assistants—and the one that will truly reshape how software gets built—uses AI-powered agents to do much of the heavy lifting. So, what is so game changing about AI agents? Agents are *goal-seeking*, and they can accomplish these goals almost entirely on their own.

While the use cases in generation one and two were one-way interactions, AI agents are more like collaborators or team members. A developer sets the goal, but the agent is left to reason through a developer’s request, share a plan based on the specifications, and execute it. Developers can iterate on the details of the plan, but much of the grunt work is left to the agent. For example, a developer working at an e-commerce company could ask an agent to specify the desired outcome: “Write a feature that allows customers to save specific items to view later.” The agent would then generate a step-by-step implementation plan. After the developer approves the change, the agent would handle the rest, connecting multiple steps to create the code, test it, and make necessary changes across the entire code base. With AI agents, every developer has access to their own team of engineers who can do everything from upgrading applications from one language version to the next to building entirely new features.

This could save teams months of undifferentiated work, and it is only the beginning for agents. In the future, agents will handle more of the software development lifecycle, freeing developers to focus on where they can have the most impact.

Welcome to Spec-Driven Development

It all sounds great–but will AI agents sideline thousands of talented software developers? Not at all. If anything, the role of developers will become even more vital as they look to guide these agents from idea to production.

For decades, some have mistakenly characterized developer talent as the ability to write code in arcane (and constantly changing) programming languages. Now, being an expert on a specific language may not be the most critical skill. Instead, developers will need much more experience in systems thinking and design. The ability to foster a deep understanding of a problem and translate that into a specification that a machine can understand will become a critical skill.

Depending on the problem, these specifications may be defined by a simple prompt or developed collaboratively as part of a larger plan with an agent. While working backwards from a problem to figure out where you're going isn't a new concept, the challenge becomes much more interesting in an era of generative AI. With an eager AI agent just a prompt away, developers will need to be much more deliberate in articulating what they want and how they want it done to maximize an agent's potential.

Perfecting the "Last Mile"

In transportation planning, the "last mile" is typically the last leg of the journey before something reaches its final destination. While the last mile may be the shortest part of the trip, it's often the most complex – chock-full of obstacles, twists, and turns that make reaching the end difficult.

Similarly, an AI agent may help developers swiftly move through the middle of the development process, but the hardest part comes at the end. Only a keen-eyed developer can properly determine if the end product produced by the agent actually meets the developer's original goal.

Developers must ask themselves: "Is this the application I want?" If the agent's product meets a developer's exact needs, they can start thinking about how they make it even better. If it falls short, the developer has a host of options available. Some developers may want to dive deep into the code, making their own tweaks and optimizations to make an application truly stand out. Others may consult more agents and work through problems iteratively, chipping away to make their vision a reality. There really is no wrong way to do it. And, with agents eliminating so much of the undifferentiated work in the middle, developers will actually have the time to perfect their own last mile to deliver something truly remarkable.

Moving Forward: Illuminate and Clarify

At Amazon, our principal engineers (some of our most senior and experienced technical employees) follow a tenet called "Illuminate and Clarify." The core of that principle is about distilling complexity, boiling a problem down to its essence, and driving a shared consensus for how to solve it.

The hallmark of software development in the age of AI agents will be very much the same. Because, ultimately, software development is about so much more than code. It's about building systems that do what users want to accomplish.

###