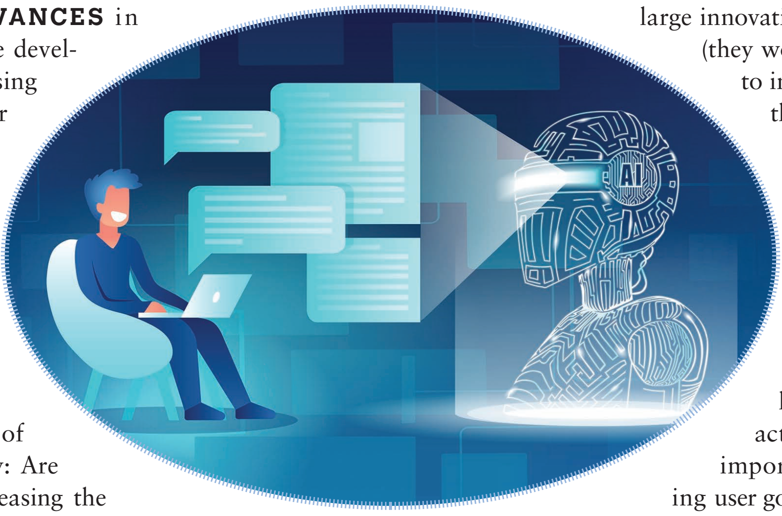
Editor: Ciera Jaspan
Google
ciera@google.com

Editor: Collin Green
Google
colling@google.com

# What Do Developers Want From AI?

Sarah D'Angelo, Ambar Murillo, Satish Chandra, and Andrew Macvean



©SHUTTERSTOCK.COM/FGC

**THE RECENT ADVANCES** in AI have resulted in the development of an increasing number of developer tools enhanced with AI (e.g., DuetAI, CoPilot, and ChatGPT for coding tasks). With this growth, there has been a lot of research on the impact of these enhancements from the perspective of developer productivity: Are AI enhancements increasing the speed at which developers write code? Do they improve the quality of the code written?[1] Do they help developers find more creative solutions? However, there has been far less discussion of where and how developers want to interact with AI in their tools. If we do not address these questions, we risk focusing too much on the technology and its capabilities and not enough on identifying promising opportunities. As we have emphasized in this column before, our team takes a human-centered approach to understanding developer productivity, and accordingly, we began our explorations into this space from the developer's perspective. Where do developers want AI in their workflows, and what do they anticipate its effects to be?

*Digital Object Identifier 10.1109/MS.2024.3363538*
*Date of current version: 11 April 2024*

## Introduction

Understanding what humans want and need from technology is the foundation of user experience research. However, this approach is sometimes questioned, especially when it comes to large technical innovations. The often misattributed quote "If I had asked people what they wanted, they would have said faster horses" is used to suggest that users may not be able to imagine large innovative technological leaps (they wouldn't have been able to imagine the car). While there is no evidence that this was ever said by Henry Ford,[2,3] it remains a popular justification of going directly to the user to better understand what they want. But if you take a closer look at the quote, it actually does reveal the importance of understanding user goals: a car is faster than a horse. It is a way to get from point A to B, faster. So if we focus on the user goal in that quote, we realize that it's not about the solution (e.g., the horse or the car), it's about the fundamental user need and goal: getting somewhere faster. In the apocryphal Ford quote, the car is a large technical innovation that meets a user's need for fast transportation. If we accept that AI is a large technical innovation that can meet needs for developers, it makes sense to explicitly investigate what needs (and preferences) developers have that AI can meet. So with this in mind, what do developers want from AI? To begin unpacking this question, we started where we typically start: with the developers.

## Efficient Horses

In our many interviews and user studies with developers who had varied experiences with AI-enhanced developer tooling, they expressed a desire for AI-enhanced developer tooling to save them time and energy by helping them do their job more efficiently. This calls us back to the faster horses quote: developers do not want AI to fundamentally change their workflows (right now), they want AI to help them do what they are already doing, getting from point A to point B. Notably, they are not only focused on speed: efficiency is more than just speed, it's also about making things easier and at least maintaining, if not improving, quality.

We also heard from developers that they want AI to support simpler tasks and reduce toil, allowing the developer to focus more of their energy on the complex problem solving and creative aspects of their jobs. As one developer put it, "Automating menial and repetitive tasks could help me focus more on the use case and the problem at hand." Similarly, another developer expressed how AI could help them feel more creative by saving time on simpler tasks: "It feels much more impactful if I can spend less time on the finer details. I think AI powered tools will help me be more creative and think about the bigger picture and long-term goals." This is because right now, engineers want to stay in control, and that is okay. It is also consistent with perspectives shared by engineers outside of Google.[4]

To put it in context, even today, we are only beginning to deploy driverless cars and in most cases still want human supervision in autonomous vehicles. This is also true in AI-enhanced developer tools. Right now, when it comes to AI in developer tooling, developers want to stay in the driver's seat. They want to maintain control and still decide how to get from A to B, even if AI can help them on the way. For example, one participant in our user studies said, "I like to use AI with 'trust, but verify.' You can look at the suggestion and say, okay, [the AI] has some confidence that it's correct, but I need to be that final barrier." Another developer reflected, "I don't think I personally would ever trust it 100% like we don't trust human written code 100%, we're like okay, you know, it needs a code review, it needs to be looked over by someone else."

This is good because we are still in the early days of AI, so we still want someone "behind the wheel." The task for us is to ensure developers feel in control when working with AI-enhanced tooling and, above all, to make sure that AI is working for developers, supporting efficiency, while letting developers solve the hard problems that they find rewarding.

This is not to say that AI enhancements in developer tooling will stay only at the level of automating simpler tasks or that developers will always want complete control. What we want to highlight is that developers want to keep doing the most rewarding aspects of their workflows and want AI tooling to help expedite the less rewarding aspects. And what falls into each category will likely change over time, also in response to developments in AI.

## The Horse to Car Journey for AI Developer Tooling

If we lean into the advent and advances of the automobile as a metaphor for the integration of AI into developer tooling, it raises important considerations for the future based on technological innovations of the past, giving us new insights into how we think about what developers want from AI in developer tooling.

We looked to the history of car technology[5] to gather inspiration for how we might think about the trajectory of AI in developer tooling. This revealed three categories of enhancements (Table 1).

When it comes to AI in developer tooling today, we're mostly enhancing existing capabilities and extending them. For example, AI-powered code completion enhances developers'

Table 1. Three categories of AI enhancements.

| Enhancement Type | Car Example | AI Example |
|---|---|---|
| *Enhancing existing human capabilities*: These are features that help humans do what they are already capable of but in an easier or more convenient fashion. | Power steering or antilock brakes | Code completion |
| *Extending human capabilities*: These are features that go beyond what humans are capable of but still depend on a more active human in the loop. | Reversing cameras or blind spot warnings | Code review suggestions, chatbots |
| *Delegating human capabilities*: These are features that replace human capabilities. | Lane-keeping, automatic braking, enhanced cruise control (distance keeping from leading cars) | Automatic rollbacks of breaking changes, automatic deletion of dead code, test generation |

ability to write code quickly, while leveraging large language models to find answers extends developers' capabilities to search for information. If we look at the evolution of car technology as a similar journey, there were many advances that made cars easier to drive, faster, and more comfortable, highlighting that in our AI journey, there is a lot of innovation to be had before we get to self-driving cars. When we think about how to build AI developer tools for developers, we should ask ourselves, how might we leverage AI to make software development more secure, higher quality, easier to learn, and efficient? Each of these played a pivotal role in making automobiles as pervasive as they are today, and we expect the same to be true for AI.

Looking to the future, when we ask engineers what they want from AI in the long term, engineers want AI to transform the way they work (e.g., elevate their thinking and help them solve complex problems).[4] For example, developers we talked to expressed excitement for expanding capabilities of AI to do more. One developer said, "It would be nice to have a J.A.R.V.I.S. type of AI that is a coding partner… It feels more like: feeding a design doc into an AI and having it generate the library. I see AI, in that case, as a programming companion." (J.A.R.V.I.S. stands for "Just a Rather Very Intelligent System"; see https://en.wikipedia.org/wiki/J.A.R.V.I.S.) Similarly, another developer mentioned, "When the suggestions and the assistance can get ahead of where I'm thinking and allow me to think faster, that to me is interesting, it's not the typing speed that is really the bottleneck for most software developers, it's the ability to really think things through." These quotes highlight the opportunities for AI to go beyond enhancing and extending human capabilities into delegating them. As we think about the future, how might we anticipate these use cases and plan for them? What is the optimal balance of system autonomy and user control?

It's important to acknowledge that other pivotal innovations were happening to improve car technology that did not alter human capabilities, for example, the introduction of the electric ignition.[3] Examples of these innovations in the AI space include model quality improvements and changes to the underlying technologies (e.g., the introduction of Transformer).[6] These are critical aspects of the "horse to car" journey that we are not focusing on in this discussion.

## Opportunities for AI to Support Developers

So, now that we've situated ourselves in the horse to car journey, what do developers want AI to do more specifically? As we've said before, we don't need to look too far. Ask developers. What are their pain points with current workflows? What tasks are error prone and tedious? We can think of AI as a tool that we can use to enhance and extend human capabilities, particularly in areas where we know developers have already expressed pain points.

At Google, every quarter we ask a third of Google developers what hinders their productivity. The top hindrance is consistently technical debt, and the following two most common hindrances are interesting opportunities for AI: 1. poor or missing documentation and 2. learning a new platform, infrastructure, framework, or technology (see Figure 1).

At Google we are leaning in to supporting these pain points by investing in AI to support both the inner and

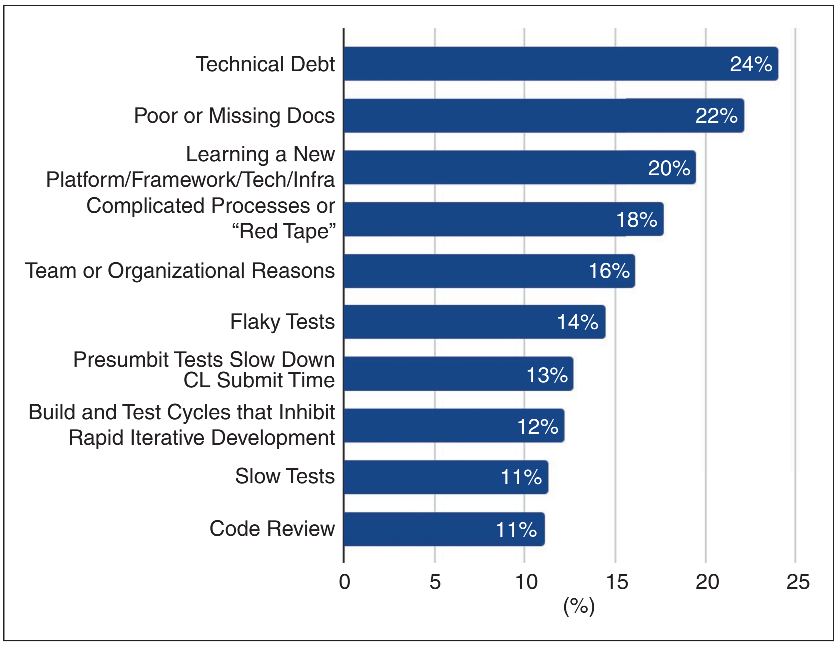


**FIGURE 1.** The top 10 hindrances to developer productivity at Google in Q4 2023 (the question asks engineers to select their top three hindrances and includes more options than shown here).

outer loops of software development,[7] not just writing code. While AI could be applied to all aspects of the development lifecycle, at Google we are driven by what developers tell us are their pain points as a starting point for our AI investments. For example, to address the pain point of learning new platforms and frameworks, we are investing in helping engineers find answers to their questions. For example, DUET AI[8] provides chatbot and troubleshooting support to developers. Additionally, we are investing in AI support for reviewing and testing code[9] to address pain points with those tasks. Asking developers about what hinders their productivity helps us identify new directions and take a holistic approach in AI in development workflows.

We also ask developers how they feel about AI in their workflows. While a majority of developers express positive impacts of AI and are in various stages of trusting it to assist their workflows, others said they do not want AI in their workflows. This is not new to AI; there were holdouts to cars as well. It is important to have skepticism with new technology: research has shown that engineers using AI-powered tools were more likely to write insecure code during security-related tasks than those without.[10,11] Adding AI to developer tooling isn't without risk, and not all developers are enthusiastic about it. (Cars are not perfectly safe, and not everyone has a driver's license.) It's important for us to understand developers' resistance to AI and identify opportunities to make improvements and meet developers where they are.

At Google, we are focusing on supporting the entire development lifecycle, understanding resistance to AI, and thinking about what AI means for product development more broadly.

## Beyond the Car Itself

Now that we have an idea of what AI can do for developers, we have to take a step back. To achieve the full benefits of cars, we did not just simply make them available, we had to create infrastructure to allow for easy adoption and reasonable evolution (e.g., highways, speed limits, and regulations). We also had to support people in transitioning from horses to cars. The same principles apply for AI. As we move toward more AI support in developer workflows, how can we anticipate structural changes that need to be made? Do we need common infrastructure for AI-powered tools? And how can we work together across industries to ensure safe, fair, and equitable access to AI in developer tools?

This takes us beyond Google and will involve the entire software engineering community. With the rapid innovation and development of AI specifically in developer tooling, these questions are increasingly important and urgent.

The evolution of AI is a pivotal moment in history, but it's not the first time we have experienced technological advances that have changed how humans work. Looking back at past examples, such as the advent and advances of automobiles, can give us a different perspective and remind us of the importance of focusing on our developers' goals, as well as highlighting that when a new technology appears, it can add value in both intended and unexpected ways. The transition from horses to cars is just one metaphor, but there are many others that can provide valuable lessons. For example, AI is not the first major advancement that has affected software engineering. Engineers were skeptical of "automatic coding,"[12] but the compiler allowed developers to codify best practices and automate them, which in turn allowed them to work at a higher level of abstraction. The addition of AI in developer tooling presents a similar opportunity.

As we think about the future of software with AI, it's important to consider what developers' fundamental goals are (regardless of what they imagine them to look like) and how they want AI to support them. At Google, we are focusing on supporting the entire development lifecycle, understanding resistance to AI, and thinking about what AI means for product development more broadly. We hope this article has given you a new lens on AI in developer tools and sparked ideas to the questions we have posed.

## ABOUT THE AUTHORS

**SARAH D'ANGELO** is a user experience researcher on the Engineering Productivity Research Team, Google, Auckland 1010, New Zealand. Contact her at sdangelo@google.com.

**AMBAR MURILLO** is a user experience researcher on the Core Developer Team, Google, 80636 Munich, Germany. Contact her at ambarm@google.com.

**SATISH CHANDRA** is the leader of the Developer AI Team, Google, Sunnyvale, CA 94089 USA. Contact him at schandra@acm.org.

**ANDREW MACVEAN** is the head of UX research in the Developer Organization, Google, Seattle, WA 98103 USA. Contact him at amacvean@google.com.

## References

1. "Early LLM-based tools for enterprise information workers likely provide meaningful boosts to productivity." Microsoft. Accessed: Feb. 1, 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/early-llm-based-tools-for-enterprise-information-workers-likely-provide-meaningful-boosts-to-productivity/
2. "Henry Ford, innovation, and that 'Faster Horse' quote," *Harvard Bus. Rev.*, Aug. 29, 2011. Accessed: Feb. 1, 2024. [Online]. Available: https://hbr.org/2011/08/henry-ford-never-said-the-fast
3. "User research and design: Three old chestnuts cracked," Foolproof, London, U.K., 2017. [Online]. Available: https://www.foolproof.co.uk/journal/user-research-and-design-three-old-chestnuts-cracked/
4. "A developer's second brain: Reducing complexity through partnership with AI." GitHub. Accessed: Feb. 1, 2024. [Online]. Available: https://github.blog/2024-01-17-a-developers-second-brain-reducing-complexity-through-partnership-with-ai/
5. "The history of car technology." Jardine Motors Groups. Accessed: Feb. 1, 2024. [Online]. Available: https://news.jardinemotors.co.uk/lifestyle/the-history-of-car-technology
6. "Transformer: A novel neural network architecture for language understanding." Google Research. Accessed: Feb. 1, 2024. [Online]. Available: https://blog.research.google/2017/08/transformer-novel-neural-network.html
7. J. Bader, S. Seohyun Kim, F. Sifei Luan, S. Chandra, and E. Meijer, "AI in software engineering at Facebook," *IEEE Softw.*, vol. 38, no. 4, pp. 52–61, Jul./Aug. 2021, doi: 10.1109/MS.2021.3061664.
8. "AI-assisted application development." Google Cloud. Accessed: Feb. 1, 2024. [Online]. Available: https://cloud.google.com/duet-ai
9. "Resolving code review comments with ML." Google Research. Accessed: Feb. 1, 2024. [Online]. Available: https://blog.research.google/2023/05/resolving-code-review-comments-with-ml.html
10. H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub copilot's code contributions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2022, pp. 754–768, doi: 10.1109/SP46214.2022.9833571.
11. N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do users write more insecure code with AI assistants?" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 2785–2799, doi: 10.1145/3576915.3623157. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3576915.3623157
12. J. W. Backus et al., "The FORTRAN automatic coding system," presented at the Western Joint Comput. Conf., Techn. Rel. (IRE-AIEE-ACM (Western)), New York, NY, USA: Association for Computing Machinery, Feb. 26–28, 1957, pp. 188–198, doi: 10.1145/1455567.1455599.