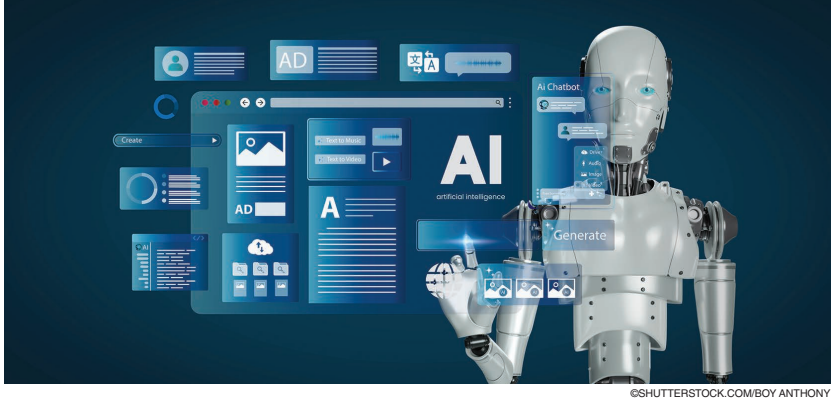
The Fine Balance Between Helping With Your Job and Taking It

AI Code Assistants Come to the Fore

Cleidson Ronald Botelho de Souza, Universidade Federal do Pará

Gema Rodríguez-Pérez, **Manaal Basha**, **Dongwook Yoon**, and **Ivan Beschastnikh**, University of British Columbia

// AI code generation tools are reshaping the software engineering landscape. We provide recommendations for practitioners interested in these tools based on narratives we have collected regarding two AI code generation tools, GitHub Copilot and Tabnine. //



©SHUTTERSTOCK.COM/BOY ANTHONY

Digital Object Identifier 10.1109/MS.2024.3357787
Date of publication 29 January 2024; date of current version 10 October 2024.

PROGRAMMING IS A nontrivial human activity. Until very recently, the idea of automating code generation was considered a pipe dream (putting aside the advances in program synthesis, which is a topic at least 10 years old).[1] This radically changed with the first generation of artificial intelligence (AI) code assistants, all released within the past few years. These include Tabnine (released 2018), GitHub Copilot (released 2021), ChatGPT (2022), and Amazon CodeWhisperer (released 2022); all of these are described as "assistants" that are intended to help increase developer productivity.

Introduction

These tools are trained on hundreds of millions of lines of source code. They work by translating a natural language prompt into a corresponding implementation. For example, prompting Copilot for "a function that computes the *i*th element in the Fibonacci sequence" will result in a function (in the programming language of the project) that implements precisely this logic. These tools may seem like a novelty. But, large companies like Google have already reported astonishing results—a Google blog post reports that "3% of new code (measured in characters) is now generated by accepting ML completion suggestions."[2] Similarly, Cisco reported a 50% increase in productivity, albeit in a specific scenario.[3]

Meanwhile, the fast-paced development of AI has generated some concern about the impact of this technology on humans and its implications for education,[4] job losses,[5] and other human aspects. This has fueled both public and scientific debate.[6]

But, how are these tools perceived by developers? And, why do some people advocate for their use, while others

argue for avoiding them? In this article, we aim to answer these questions by considering the public perception of these tools based on discussions on Twitter. We provide some quantitative results from our study and then detail four concrete narratives that we believe are representative. In short, we read many Twitter discussions so you do not have to. Let's dig in!

Distilling Narratives From Twitter Discourses

We have distilled 39k tweets about Copilot and Tabnine into a sample of 331 tweets. We focused on Copilot and Tabnine because they were the first two popular code generation tools with the largest volume of related tweets. Our 331 tweets had a wide reach and were representative of the narratives generated by users of these tools. We analyzed this sample to understand the promises and concerns of these tools as reported by their users. Before detailing the methodology behind our study, it is important to mention why we used Twitter. Previous studies have shown that developers use Twitter to stay current with recent technological trends.[7] Specifically, Sharma and colleagues[8] report that the third most popular category discussed by software developers on Twitter is exactly New Releases, that is, "tweets announcing the release of a new software version, tool, etc." [The first one is Article and Multimedia Sharing (tweets sharing articles, blogs, tutorials, or videos related to software development), and the second is Technical Discussions (tweets discussing some technical issues related to software development).] Indeed, Copilot averaged 1,097 tweets per month before its release on VS Code; after that, it averaged 1,175 tweets per month, which is a 7% increase in tweets. Tabnine averaged 246 tweets per month before its release on VS Code; after it averaged 329 tweets per month, which is a 34% increase in tweets. Since we are interested in understanding software developers' reactions to the first generation of AI code assistants, Twitter data proved valuable because they often included early adopters of these assistants as well as of other technologies (https://blog.twitter.com/en_us/a/2016/new-research-8-ways-early-tech-adopters-use-twitter). Twitter is a starting point for this line of research and leaves room for future studies of other platforms, such as StackOverflow and Reddit.

We used quantitative and qualitative approaches to analyze the public English discourse to build the narratives around AI code generation tools. We first retrieved the tweets of interest using the Twitter application programming interface, querying it with the keywords "tabnine" and "github copilot." (These were the tools publicly available when we collected the data in November 2022. Other tools were available either in beta or had recently been launched.) We limited our search to retrieve tweets published only after the public releases of Tabnine (November 2018) and GitHub Copilot (June 2021). We obtained 39,361 tweets about GitHub Copilot (25,516) and Tabnine (13,845). We identified tweets in 81 different languages, including English (24,692), Spanish (1,899), German (479), etc. Out of the 39,361 tweets, 64.29% were posted by users who had associated location information in their profile. Figure 1 presents a map with the locations of these users. The map indicates that most tweets were from North America, Europe, and India. The countries with the most tweets were Pakistan, the United States, India, the United Kingdom, and Canada (in decreasing order).

We first filtered out 14,669 retweets and non-English tweets. Given that English tweets are the majority of the data and that English is regarded as the *lingua franca* of computing (https://en.wikipedia.org/wiki/English_in_computing), we assume the dataset sufficiently applies to the general public. Then, we identified the most significant tweets and qualitatively analyzed the narratives behind them. We consider a tweet to be significant if it belongs to the 1% of tweets that have the largest number of retweets, the largest number of replies, or the largest number of likes. In short, by significant,

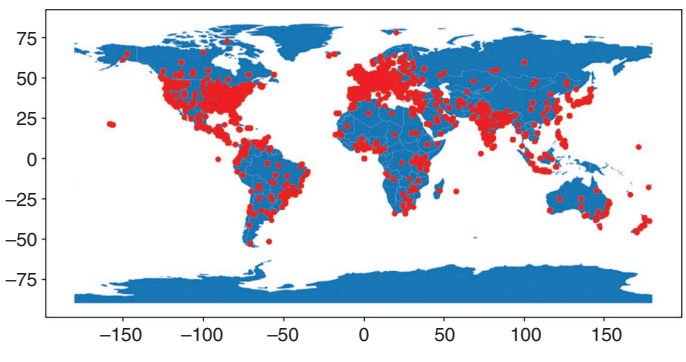


FIGURE 1. A map with locations of Twitter users for those users who had their location available in our dataset.

we mean tweets that have reached a large audience. This yielded 255 tweets about Copilot and 138 tweets about Tabnine. Table 1 overviews our final dataset.

The resulting codebook is a list of themes associated with each tweet. For instance, there are tweets describing performance issues with the AI-based code generation tools and tweets mentioning the low quality of the code suggested by the tools. Therefore, we organized these two codes (performance and quality) as examples of a larger category, called *Concerns*. Similarly, we identified *Promises* as another category that encompasses the codes associated with the benefits of the tools, including usefulness, efficiency, integration with other software development tools, etc.

In addition, we identified the *Goal* of each tweet. In this case, tweets might be informative, provide recommendations, be funny, or be an advertisement about one of the tools. For instance, the following tweet recommends those writing PowerShell scripts to adopt GitHub Copilot: "If you write #PowerShell scripts, you must check out #GitHub Copilot in #VSCode." Tweets marked as advertisements were those posted by the companies that build the AI tools and the people who work for those companies. Note that the categories are orthogonal; that is, one tweet might be informative and, at the same time, describe a concern with a particular AI-based code generation tool. In fact, most tweets belonged to more than one category.

After creating our codebook, one of the authors coded the 393 tweets. We removed from our sample 57 tweets whose goal was to advertise the tools and five tweets that were not related to Tabnine or GitHub Copilot. In the end, we qualitatively analyzed 336 tweets. (Our final codebook with categories and codes, our final dataset, and the script we used to retrieve the data are available online at https://osf.io/xerz2/?view_only=ac60108ce22241b1bb9b51f40c781d86.) Out of these 393 tweets, only 30 occurred after Twitter was acquired. This indicates that the change in Twitter's ownership had at most a minor impact on the narratives we identified.

Narratives

By creating the codebook and associating codes to tweets, we were able to determine common narratives within the dataset while ensuring a level of consistency in our coding. Using the 31 unique codes from the codebook, we found that the themes revealed a mixed sentiment toward AI code generation tools. Developers appreciated their productivity benefits and broad applicability, but they also expressed concerns about code quality, security, and ethical implications. These tools were seen as potentially transformative for software development practices, with the fear of AI replacing human developers considered unwarranted. We then used our codebook and the narratives to assemble recommendations for developers in the section "Recommendations for Practitioners."

Table 1. Descriptive statistics of our final tweets dataset.

		Copilot	Tabnine
Users	Number of unique users	4,634	1,890
	Top 5 countries	Pakistan, United States, India, Great Britain, Canada	United States, India, Great Britain, Germany, Canada
	Occupations of top 15 users	IT services, bloggers, developers, freelancers, academics, software practitioners, cofounders	IT services, bloggers, developers, developer relations, freelancers, software practitioners, cofounders
Tweets	Number of tweets	255	138
	Minimum number of likes	1	
	Mean number of likes	137	
	Maximum number of likes	6247	
	Minimum number of retweets	2	
	Mean number of retweets	21	
	Maximum number of retweets	1788	
	Minimum number of comments	0	
	Mean number of comments	6	
	Maximum number of comments	186	

Promises: Useful Code Suggestions

In general, several tweets mentioned that the tools provide suggestions that are very accurate and useful (*"really good at predicting useful completions", "crazy good", "Pretty scary how accurate it is at predicting your code, especially after a while, it gets better at predicting your code."*). These tweets often also reflected how *impressed* users are with the capabilities of the studied AI tools. For instance:

> I was so surprised how good it [Tool] is
>
> I've been trying [Tool] an auto-completion tool for VS Code that blew my socks off. Sometimes it seems like it's just reading my mind
>
> Been trying out the [Tool] with C# and @xunit in VS2022. It really is pretty mind blowing.

A related promise is associated with the broad applicability of AI-based code generation tools. For instance, we found tweets mentioning different programming languages (Java, Flutter, Dart, Python, and Rust) and frameworks (Drupal, React.js, etc.).

Beyond Code Suggestions: New Work Practices

Given the quality of the suggestions and their applicability, the AI tools *promise* to impact developers' work in different ways. First of all, they allow developers to save time during development (however, our methodology does not allow us to make any claims about the impact of these tools on developers' productivity).[9] (*"I feel like I've been able to write faster", "it really saves me tons of time.", "Saves me tons of typing"., "the piece of software that most improved my productivity in the last months."," a noticeable reduction in workload."*) AI tools also influence developers' work styles since some users reported one important side benefit of using these tools: better code documentation. This is illustrated with the following tweets:

> I've noticed that a nice secondary consequence of using [Tool] is that it's causing me to leave better comments because of the need to make sure the AI understands what I'm trying to do.
>
> By far the greatest benefit of using [Tool] so far is I now don't have to be forced to document my code. I actively *want* to write great comments, because when I do, I get the dopamine hit of a good [Tool] suggestion.

Last but not least, we found tweets discussing how AI-based code generation tools can facilitate the *learning* of programming languages and frameworks. The following tweet illustrates this point:

> I'm using [Tool] like a private tutor to learn Haskell. Here it is telling me how to translate Python dicts to Haskell's maps. And of course it knows how to build a Markdown table. Pretty close to magic

The user who posted the previous tweet has a Ph.D. degree and has also previously worked for Google and Meta. But, we found supportive tweets about AI-based code generation tools from those with little programming experience, too. For example, we identified tweets with the hashtag #100daysofcode, which is used by someone who accepted the challenge to learn to code, or improve one's code skills, by programming and tweeting during 100 days (https://www.100daysofcode.com). In this context, the following tweet is the beginning of a thread about how an AI tool could be used by beginner developers:

> Experienced devs like [Tool], but it is even more important for beginners!
>
> People learning to code often have ideas they want to implement, but spend a huge amount of time on simple things: reading data from files or visualizing it.
>
> [Tool] can help

Technical and Legal Concerns

In parallel to the narratives about the promises of AI-based code generation tools, there was a narrative about the *concerns* that arise when using these tools. These are technical and legal ones.

Technical concerns are about the quality of the code suggestions by the AI tools because either they are incorrect (*"should never keep the output verbatim or trust it"," this* [Tool] *completion ends where a\n should be escaping is hard","don't forget* [Tool], *an ML model that can't code"*) or insecure (*"I'd love it if you didn't suggest vulnerable code snippets like the* [example]", *"Careful!* [Tool] *doesn't always generate the most secure code."*). Whenever posting about the incorrect suggested code snippets, users often posted screenshots or short videos capturing their use of the tool.

There is one major legal concern discussed in the tweets. As explained, AI-based code generation tools are trained with code from millions of open source projects. However, we found tweets mentioning the possible unethical aspects of these tools, for instance, because *"People's*

source code … is feed into these machines then turned into a paid service without these original creators even agreeing to this let alone getting compensated."

The following tweet makes a parallel between computer-generated images by systems like DALL-E and AI code generation tools:

> This most likely violates licenses of artworks similar to how code was violated by [Tool.] [Twitterhandle] is doing a class action lawsuit against them for violating licenses, do with that information what you want.

The previous tweet also mentions a lawsuit against a major technology company, which is a popular concern identified in the tweets. Some developers even "tested" AI-based tools to find out whether their code was inappropriately suggested by these tools. For instance, the following tweet even included screenshots of such a test:

> [Tool] with "public code" blocked, emits large chunks of my copyrighted code, with no attribution, no LGPL license. (…) My code on left, [Tool] on right. Not OK.

Both technical and legal concerns led many Twitter users to either abandon or not adopt AI-based tools. For instance, a developer reported the following about the technical aspects of the tool he tried:

> So I finally got around to trying [Tool.] My overall experience: it can be remarkable at times, but for the most part, it slowed me down and made my work more stressful. I felt a significant sense of relief when I uninstalled it.

Legal concerns, however, not only led users to abandon the tools, but also led them to discourage *others* to adopt these tools.

> Any employer who allows their employees to use [Tool] should be aware they might be sued for copyright intrusion - and anybody putting code on [Tool] should be aware [Tool] will ignore your license and try to steal it to make $$$. Shameful.

According to a Twitter user, this legal concern was exacerbated by a U.S.-centric view of tool developers; that is, we observed complaints about U.S. companies and their narrow view of privacy, contrasting it with the European Union (EU) and its privacy law, the General Data Protection Regulation (GDPR):

> The internet isn't just US, why does everyone forget the EU exists. gdpr exists for a very good reason

AI Impact on Software Development

Another narrative we identified in the tweets was about the future of software development work given the impact and evolution of AI in software development tools. This narrative spans three aspects.

First, many tweets mentioned the integration of voice interaction with IDEs so that a developer can speak her intentions and the IDE, with the AI-based code tool, would generate the requested code. Additional tweets noticed the opportunity to increase accessibility and inclusion in software development, as exemplified in the following tweet:

> This [voice-based interaction with code generation] will be an amazing #Accessibility tool for my limited mobility friends AND any of you with carpel tunnel, broken wrists, etc.

We note that most of the tweets about voice-based interaction were inspired by the beta launch of a feature in this context.

The second aspect concerns the future of AI-based tools. In this case, we identified tweets that speculated about potential new development tools to be built inspired by AI, for instance, supporting pull requests and code reviews, the command-line interface, scripts, etc. As an experienced developer tweeted:

> We can't be far off AI being able to automate tedious refactorings of giant codebases. It's a shame we mostly went through the Python 2-3 migration a few years before it would be feasible to have some slightly more advanced version of [Tool] do the whole thing for us

Finally, the last aspect includes AI anxiety[6]: concerns about being replaced by automated AI tools. The following tweet expresses this idea:

> Man … seeing how [Tool] automates coding it really IS a question if coders are even going to be needed at all in a few years: P
>
> It's really scary how good that shit is getting.

Recommendations for Practitioners

Software engineering is a continuously changing field. Rather than perceiving AI code assistants as a threat or a chore (another new technology to install/learn/use), we believe practitioners should see these tools as an opportunity. While we found a narrative criticizing the quality of AI

suggestions, there is a contrasting narrative that captures how users were also impressed by the suggestions. Previous research[1,10] on AI code assistants indicates that even bad suggestions can be beneficial as they provide a "starting point" for developers to complete their tasks.

At this point, it is unclear why there is a difference between the good and bad suggestions, that is, whether that difference is related to the professionals using the tools, the quality of the prompts, the programming language used, a combination of these elements, or something else. We recommend that users experiment with these tools to see for themselves if these AI tools are useful *to them*, given their industry, individual use case, and satisfaction with the generated suggestions. In short, *give your AI coding assistant a try.*

It is also important to manage expectations instead of blindly assuming that because the tool worked well for someone else, it will also work for me. In other words, *decide it for yourself.* Finally, one should not discount the fast-paced development of AI technologies, which suggests that tools rapidly improve and might prove useful after just a couple more months, so *be patient with your AI coding assistant.*

Some of the narratives we identified suggest that AI-based tools will make software development more inclusive and accessible, especially through voice-based interfaces. New AI-based tools are already being created, extending the tools we consider in this article, as well as tools that address other aspects of the software development landscape; see, for instance, the work of Sawant and Devanbu[11] and Pradel and Chandra.[12] AI assistants also have the potential to influence the learning process of software developers who constantly need to learn new technologies. Arguably, in simple projects, these tools might allow a single software engineer to develop the entire project without the need to hire a specialized engineer with expertise in, for instance, a particular type of database. In this case, our recommendation is *to be on the lookout for new AI coding assistants.*

An issue with existing AI assistants is that there is no way to distinguish generated code from human-written code.[13] Yet, retaining traceability is important: which assistant was used to generate the code, which prompt was used, was the generated code inspected and/or modified by a human, etc.? This information, if currently maintained, is tracked manually. But, without this information, the provenance of the codebase quickly deteriorates. We recommend that practitioners annotate their code with comments containing traceability information; that is, *document your AI coding sessions.*

Besides the tools coming out of large companies, there are already open source alternatives like GPT-Code-Clippy (https://github.com/CodedotAl/gpt-code-clippy). We expect more such alternatives in the future. These open source tools are also likely to be easier to review and use in commercial contexts and are less likely to face legal concerns.

Finally, it is all too easy to entertain the notion of AI assistants that come to replace human engineers. AI assistants are indeed highly sophisticated tools, and they stand to become increasingly important in our profession. However, engineering robust software requires complex human-in-the-loop tasks. So, we consider the replacement fear to be unwarranted. So, put your fears aside: *make your AI coding assistant your friend* and see what this amazing technology has to offer!

AI code generation tools have great potential to impact the software development process. We extracted tweets about two of these tools, GitHub Copilot and Tabnine, to identify the different narratives being discussed about these tools. We extracted 39k tweets made from June 2021 to December 2022, filtered them, and analyzed the 331 tweets that have reached the largest audience, measured by the number of likes, retweets, and comments. Our analysis yielded different narratives describing the promises of these tools, the technical and legal concerns around their usage, and the potential impact of AI on the software development process. A limitation of this study is our sample size of the tweets, which can affect the generalizability and comprehensiveness of our recommendations. We concluded our article with recommendations for practitioners about how to go about adopting these tools to benefit their careers.

We are currently analyzing other sources of information (StackOverflow, Reddit, etc.) to find out whether the narratives around AI-based code assistants on these platforms are similar to the ones we identified on Twitter.

Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada for Discovery Grant RGPIN-2014-04870 and funding reference GR023171. The first author would like to thank the National Council for Scientific and Technological Development for

Grants 420406/2023-9 and 442779/2023-2. This work involved human subjects or animals in its research. The authors confirm that all human/animal subject research procedures and protocols are exempt from review board approval.

References

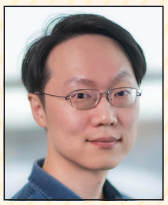
1. P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Proc. Extended Abstracts CHI Conf. Human Factors Comput. Syst.*, New York, NY, USA: Association for Computing Machinery, 2022, pp. 1–7, doi: 10.1145/3491101.3519665.
2. "ML-enhanced code completion improves developer productivity." Google Research. Accessed: Dec. 12, 2022. [Online]. Available: https://ai.googleblog.com/2022/07/ml-enhanced-code-completion-improves.html,
3. "Learn how tabnine enables 'pair programming' with AI." CISCO. Accessed: Dec. 12, 2022. [Online]. Available: https://blogs.cisco.com/developer/tabninepairprogramming01
4. Y. N. Harari, "Reboot for the AI revolution," *Nature*, vol. 550, no. 7676, pp. 324–327, Oct. 2017, doi: 10.1038/550324a.
5. J. Manyika et al., "Jobs lost, jobs gained: What the future of work will mean for jobs, skills, and wages," McKinsey Global Inst., Washington, DC, USA, Tech. Rep., 2017. Accessed: Dec. 13, 2022. [Online]. Available: https://www.mckinsey.com/featured-insights/future-of-work/jobs-lost-jobs-gained-what-the-future-of-work-will-mean-for-jobs-skills-and-wages
6. J. Li and J.-S. Huang, "Dimensions of artificial intelligence anxiety based on the integrated fear acquisition theory," *Technol. Soc.*, vol. 63, Nov. 2020, Art. no. 101410, doi: 10.1016/j.techsoc.2020.101410. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0160791X20300476
7. L. Singer, F. Figueira Filho, and M.-A. Storey, "Software engineering at the speed of light: How developers stay current using Twitter," in *Proc. 36th Int. Conf. Softw. Eng., (ICSE)*, New York, NY, USA: Association for Computing

ABOUT THE AUTHORS

CLEIDSON RONALD BOTELHO DE SOUZA is with Universidade Federal do Pará, Belém 66075110, Brazil. His research interests include collaorative and human aspects of software engineering and AI coding assistants. de Souza received his Ph.D. in information and computer sciences from the University of California, Irvine. Contact him at cleidson.desouza@acm.org.

GEMA RODRÍGUEZ-PÉREZ is with the University of British Columbia, Kelowna, BC V1V 1V7, Canada. Her research interests include software engineering, mining software repositories, and human aspects of software engineering. Rodríguez-Pérez received her Ph.D. in information technology and communications from the Universidad Rey Juan Carlos. Contact her at gema.rodriguezperez@ubc.ca.

MANAAL BASHA is with the University of British Columbia, Kelowna, BC V1V 1V7, Canada. Her research interests include natural language processing, large language models, and human aspects of software engineering. Basha received her B.Sc. specialization in mathematics and economics from the University of British Columbia. Contact her at manaals@mail.ubc.ca.



DONGWOOK YOON is with the University of British Columbia, Vancouver, BC V6T 1Z4, Canada. His research interests include human-computer interaction, human-AI interaction, and computer-supported cooperative work and social computing. Yoon received his Ph.D. in information science from Cornell University. Contact him at yoon@cs.ubc.ca.

IVAN BESCHASTNIKH is with the University of British Columbia, Vancouver, BC V6T 1Z4, Canada. His research interests include software engineering and distributed systems. Beschastnikh received his Ph.D. in computer science from the University of Washington. Contact him at bestchai@cs.ubc.ca.

Machinery, 2014, pp. 211–221, doi: 10.1145/2568225.2568305.
8. A. Sharma, Y. Tian, and D. Lo, "What's hot in software engineering Twitter space?" in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, 2015., pp. 541–545, doi: 10.1109/ICSM.2015.7332510.
9. N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, and J. Butler, "The space of developer productivity: There's more to it than you think," *Queue*, vol. 19, no. 1, pp. 20–48, Mar. 2021, doi: 10.1145/3454122.3454124.
10. J. D. Weisz et al., "Perfection not required? Human-AI partnerships in code translation," in *Proc. 26th Int. Conf. Intell. User Interfaces (IUI)*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 402–412, doi: 10.1145/3397481.3450656.
11. A. A. Sawant and P. Devanbu, "Naturally!: How breakthroughs in natural language processing can dramatically help developers," *IEEE Softw.*, vol. 38, no. 5, pp. 118–123, Sep./Oct. 2021, doi: 10.1109/MS.2021.3086338.
12. M. Pradel and S. Chandra, "Neural software analysis," *Commun. ACM*, vol. 65, no. 1, pp. 86–96, Dec. 2021, doi: 10.1145/3460348.
13. "Apparently I'm a robot." Twitter. Accessed: Dec. 13, 2022. [Online]. Available: https://mobile.twitter.com/JanelleCShane/status/1601729685385535488

Digital Object Identifier 10.1109/MS.2024.3459168