10 Years Later: Revisiting How Developers Search for Code

KATHRYN T. STOLEE, North Carolina State University, USA
TOBIAS WELP, Google, Germany
CAITLIN SADOWSKI, Unaffiliated, USA
SEBASTIAN ELBAUM, University of Virginia, USA

Code search is an integral part of a developer's workflow. In 2015, researchers published a paper reflecting on the code search practices at Google of 27 developers who used the internal Code Search tool. That paper had first-hand accounts for why those developers were using code search and highlighted how often and in what situations developers were searching for code. In the past decade, much has changed in the landscape of developer support. New languages have emerged, auto-complete in the IDE has gotten better, artificial intelligence (AI) for code generation has gained traction, Q&A forums have increased in popularity, and code repositories are larger than ever. It is worth considering whether those observations from almost a decade ago have stood the test of time.

In this work, inspired by the prior survey about the Code Search tool, we run a series of three surveys with 1,945 total responses and report overall Code Search usage statistics for over 100,000 users. Unlike the prior work, in our surveys, we include explicit success criteria to understand when code search is meeting users' needs, and when it is not. We find that Code Search users continue to use the tool frequently and the frequency has not changed despite the introduction of AI-enhanced development support. Users continue to turn to Code Search to find examples, but the frequency of example-seeking behavior has decreased. More often than before, users access the tool to learn about and explore code. This has implications for future code search support in software development.

CCS Concepts: • **Software and its engineering** → *Software development methods*; **Development frameworks and environments**; *Software libraries and repositories.*

Additional Key Words and Phrases: Code search, developer surveys, development practices in industry

ACM Reference Format:
Kathryn T. Stolee, Tobias Welp, Caitlin Sadowski, and Sebastian Elbaum. 2025. 10 Years Later: Revisiting How Developers Search for Code. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE055 (July 2025), 21 pages. https://doi.org/10.1145/3715774

1 Introduction

Research has shown that searching for code is integral for developers in their workflow, and the number of publications in this area has been increasing [10]. One study at Google, conducted by a subset of the authors of this paper, found that developers search for code using the internal Code Search tool, which indexes all the company's source code, an average of 12 times a day (median of 6) [33]. Even outside Google where developers do not have bespoke search tools such as this at their disposal, they still turn to performing code searches in general-purpose search engines [21, 32, 35, 39] on a daily or weekly basis.

Authors' Contact Information: Kathryn T. Stolee, North Carolina State University, Raleigh, USA, ktstolee@ncsu.edu; Tobias Welp, Google, Munich, Germany, twelp@google.com; Caitlin Sadowski, Unaffiliated, Mountain View, CA, USA, supertriceratops@gmail.com; Sebastian Elbaum, University of Virginia, Charlottesville, USA, selbaum@virginia.edu.

This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2025 Copyright held by the owner/author(s).
ACM 2994-970X/2025/7-ARTFSE055
https://doi.org/10.1145/3715774

Over the past decade, however, the nature of development has shifted. Developers have access to extensive Q&A forums (e.g., StackOverflow) and to programs that help newcomers have a smoother experience contributing [11]; these may *reduce* the need for code search support. Developers have access to massive quantities of source code that can be reused and remixed without having to write everything from scratch [4]; this may *increase* the need for code search support. Some tasks that were previously solved with web search can now be solved with AI, such as finding examples to reuse [37] or understanding what code does [28], which may *reduce* the need for code search support. Yet, other code search tasks still *require* previous code search tools, such as finding a particular location in source code. These changes mentioned merely scratch the surface, and it leads us to ask, **is code search as critical or relevant for developers today as it was ten years ago?**

In this work, we aim to assess the degree to which code search is being used today, and additionally, if and how its usage has changed with the introduction of new languages, IDEs, and AI tools to better support development tasks. We focus on users of the Code Search tool at Google, referred to as *developers* for simplicity, as the primary user group of the tool is those who write and review source code at Google.

The contributions of this work are:

- A logs analysis inspired by prior work that looks at Code Search usage frequency. While prior work captured 15 days of data for 27 users, we capture 30 months of usage for over 100,000 users (Section 2.2).
- Surveys of developers with 1,945 responses about their Code Search behavior (RQ1), based on prior work [33].
- A deep exploration of the role of Code Search in finding code examples (RQ2).
- Clarifications on the role Code Search plays during code review (RQ3).
- Analysis of developer satisfaction during code review based on an explicit signal of Code Search success (RQ4).

Through the this research, we gained following insights into how the code search process is conducted today at Google:

- Developers still use Code Search to regularly support their development activities, with an average of 7 queries per user per weekday (averaged over 3 months). Code Search query frequency per user per workday has remained consistent over a 30-month period, despite changes in tooling, languages, and development support from AI (Section 2.2).
- The motivations behind performing Code Search queries (e.g., to find examples, to learn why something is failing, to read code) have remained largely consistent compared to 10 years ago. (RQ1)
- Of those seeking code examples, most wanted the most common way of using the API in production code. Free-text responses suggested that *where* the example was located and *who* wrote it are considerations in deciding whether the example is acceptable. (RQ2)
- During the code review process, we found that issuing Code Search queries is common. While authoring code changes is more associated with example-seeking behavior or trying to understand why something is failing, reviewing code changes is more associated with exploration activities in Code Search. (RQ3)
- Code Search users are very satisfied with the tool, though those looking for examples are somewhat less satisfied. Code Search users looking for a code location were most likely to be completely satisfied. (RQ4)

These results have implications for where and how to better support developers through Code Search, and also through AI. The rest of the paper is organized as follows: Section 2 presents

background information on development at Google, including a logs analysis on the prevalence of Code Search among developers. The study RQs and survey design considerations are in Section 3, followed by results and threats to validity in Section 4. The discussion, including future work and lessons learned, is in Section 5. Related work comparing prior code search studies to this one is in Section 6, followed by the conclusion in Section 7.

2 Background on Google Code Search

This section focuses on development practices at Google and includes a logs analysis about Code Search frequency.

2.1 Development and Tooling

Code Search tooling at Google enables developers to effectively search and browse through Google's source code. The standalone tool, which was studied previously [33], is enabled by an index that organizes the vast code base in a way that allows for efficient information retrieval. Users interact with the tool through a search bar for issuing queries. Search results contain snapshots of relevant code and links that point to the specific code locations. A search bar that accesses the same Code Search backend is also embedded in the IDE.

Developers specify their queries using plain text search terms and/or regular expressions. Additional search flags are available to limit the search scope to specific directories, file names, or languages. As with web search, the user interface provides suggestions while the developers formulate their queries. Customizable layers in the UI show additional information depending on user needs, such as runtime information or when the code was last modified [38].

Code review at Google is also performed using bespoke internal tools. The process involves the submission and subsequent review of a change list (CL), which is analogous to a pull request (PR). For example, developers create CLs when writing new features or fixing bugs. Code reviewers work on, or evaluate, CLs when reviewing proposed code changes prior to merging it into a code base or requesting changes. In this paper, as we were focused on those at Google who write code, we adopt the CL terminology.

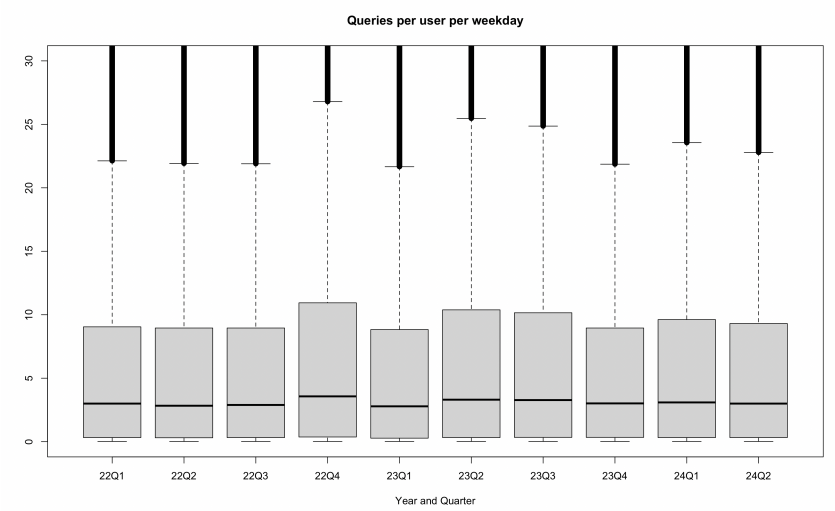
2.2 Code Search Tool Usage

We analyzed the Code Search logs at Google for 30 months, from January 2022 to July 2024, following a procedure similar to the one described in the previous Code Search study performed at Google [33]. In the last quarter of 2022, AI started gaining momentum in code generation [13, 24]. Tools such as Copilot [12] and ChatGPT [29] were available and undergoing consistent improvements. In the code search research community, discussions were emerging about the impact of such tools on the future of code search tooling and adoption [6]. As AI was becoming more and more a part of developer workflows, we wondered if code search as it was, is still relevant.

To understand the impact of AI tools on code search behavior, we turned to the Code Search tool logs at Google. The tool is used by over 100,000 developers on a quarterly basis. There were 77k monthly users in January 2022 and this slowly ticked up to 96k monthly users in August 2022 and remained between 89k and 96k monthly users for the next 2 years, through August 2024.

We wanted to capture all the search query events for users, similar to the prior work, which "collected search events and interactions with the code search tool." [33] Presently, the most common search interfaces are the standalone tool, which accounts for 65% of the human-written search queries, and the embedded IDE search bar, which accounts for 35% of the human-written search queries. We considered queries from both sources, as they both interact with the same backend.

Figure 1 reports boxplots of quarterly distributions for the number of queries per user per day, and the table shows the annualized distributions of per-user per-day queries. For example, the



	2022	2023	2024
Min	0.0154	0.0154	0.0154
Quartile 1	0.3231	0.3231	0.3231
Median	3.0462	3.0923	3.0462
Mean	7.0048	7.1747	7.1458
Quartile 3	9.4462	9.5538	9.4615
Max	861.1538	1258.3231	619.9538

Fig. 1. Queries per user per workday is on the y-axis, and time periods are on the x-axis. In the boxplot, the number of queries per user per day are averaged over each 3-month quarter, from Quarter 1 2022 (22Q1) until Quarter 2 2024 (24Q2) (That is, January 2022 until June 2024). In the table, the min, quartile-1, median, mean, quartile-3, and max queries per user per day are computed for each calendar year.

mean queries per user per day per quarter in 2022 was 7.00, compared to 7.17 in 2023. Over the analyzed time period, the average number of queries per user per workday in a quarter[1] is 7.1053, and relatively stable, as shown in the plots. This is lower than the average of 12 queries (median of 6) per user per day that was reported in prior work [33], however, details and trends matter.

Looking closer, in the prior work, the study was conducted on 27 developers who volunteered to be studied. These developers were mostly known to the researchers, and it is possible that they volunteered because they knew they would be doing code search intensive activities over the next two weeks. This new data, by contrast, includes all developers and non-developers who issued even one query to Code Search in a quarter, which would make their per-day usage over the quarter rather low. The timeframe for the previous study was approximately 2 weeks, compared to approximately 13 weeks in a quarter for our logs data. As not all developers are working on code every day, and the Code Search logs also captures non-developers who used the Code Search, say, once that quarter, we should expect our numbers here to be lower than reported by the original researchers. It is also possible the developers in the prior work were simply power users of Code Search. The 12 queries per user per day reported previously is aligned with the 81st percentile in the current dataset.

[1]Our query is imperfect in that it ignores holidays and assumes everyone observes the same weekend days of Saturday and Sunday.

During 2022, generative AI started gaining traction as a development assistant. And yet, based on the code search query distributions, the trend for usage of the tool per user per day has remained flat. This underscores the importance of Code Search *independent of AI* to support software development. Even with AI support in the development pipeline, Code Search remains critical infrastructure.

Summary: The Code Search tool is used frequently across a large and diverse set of developers at Google, with an average of 7 queries per user per day. The number of queries to this tool, which does not have AI support, has remained flat over the past 30 months of AI innovation in development workflows, highlighting its continued importance for developers.

3 Study

To understand code search behavior, we posed four research questions and designed three surveys to address the questions. At the time this work was conducted, Large Language Models (LLMs) were not part of the developer workflow. In fact, the Code Search tool had no AI support. And so, the RQs and associated survey questions do not include questions about AI usage. The surveys were deployed in the standalone Code Search tool, as described in Section 3.3.

3.1 Research Questions

Toward the goal of understanding Code Search usage, RQ1 is a replication of prior work [33]; we compare the results against observations from that prior work. RQ2, RQ3, and RQ4 are new research questions studied in this work to address questions that emerged in the prior study.

RQ1 : Why do developers search for code?

Prior work [33] posed the question, "*Why do programmers search?*" The authors used an open-ended question about search intent and used a card sorting process to qualitatively identify common topics. Here, we take those topics that emerged and turn them into options in a survey question. This way, we can collect the search intent at scale, covering more developers and development contexts, with the goal of testing the *generalizability* of the original findings to more developers and the *robustness* of the findings after ten years.

RQ2 : What are developers looking for when they want code examples?

In the prior work, it was found that developers frequently sought examples of how to use a specific API, as this category of results to their RQ1 reflected 33.5% of the queries [33]. As code examples can serve many purposes (e.g., to learn from, to reuse), in this work, we dive deeper into the needs of developers who are seeking examples.

RQ3 : What role does Code Search play during code review?

The role of Code Search during code review is largely missing from the code review literature (e.g., code review is never mentioned in a recent survey on code review techniques and studies [10]), and yet, according to the prior work [33], over half the developers were working on a code change task at the time of their Code Search query. In this work, we dive further into the role of Code Search during code change authoring and code change reviewing.

RQ4 : In what contexts is Code Search falling short of developer needs?

One challenging part about research in code search in general is knowing when a query or search session is successful. As the prior work [33] observed, the fact that a result is clicked is not necessarily indicative of success, and the fact that a result is not clicked is not necessarily indicative of a lack of success. This limits our ability to use logs alone when trying to discern search success. Therefore, in this work, we design the surveys to explicitly ask whether their search journey

(1) **What is the main reason you are using Code Search right now?**
  (a) I am exploring or reading code for understanding (e.g., understand what it does)
  (b) I am looking for a specific code location (e.g., where a class is instantiated)
  (c) I need example code for how to do something (e.g., how to use an API, discover an API)
  (d) I want to know who or when a particular file was modified
  (e) I want to know why something is failing, or the side effects of a proposed change
  (f) [Other]
(2) **What do you want to do with the code once you've found it?**
  (a) Continue exploring / learning about the code
  (b) Copy/paste code into my IDE, or work on a CL as an author
  (c) Fix or diagnose bugs or production issues
  (d) Work on a CL as a reviewer
  (e) Tell a teammate about what I've found
  (f) Work on a design document or other form of documentation
  (g) [Other]
(3) **Did you get what you wanted from Code Search today?**
  (a) Yes, completely
  (b) Yes, partially
  (c) I'm still working on it
  (d) No

Fig. 2. Code search *Intent* survey questions for RQ1 & RQ4. Question 3 was only asked to a subset of the respondents after results were found to be overwhelmingly consistent.

is successful. Then, we correlate success with different code search intentions and activities to understand when search is supported well, and when it is not.

3.2 Survey Design

Here, we explain the design of each survey and its questions.

3.2.1 RQ1 Survey: Intent. The *Intent* survey is shown in Figure 2. To understand why developers are searching for code, we asked the question, *"What is the main reason you are using Code Search right now?"* The responses for the first question come directly from the *Categories* in the qualitative analysis in prior work [33]. The responses for question 2 are also inspired by the prior work, where participants could respond to a survey question, *"What are you doing?"* After conversations with developers, we modified the responses about working on a CL (see Section 2.1) to separately include working on a CL as an author (option b) or as a reviewer (option d). The third question aimed to address a limitation of prior work in terms of understanding search success: we explicitly asked participants if their search activity was successful. This final question is used to address RQ4.

3.2.2 RQ2 Survey: Examples. Based on the high frequency of cases where developers use Code Search to find examples, we dug deeper to uncover the types of examples they seek. The *Examples* survey is shown in Figure 3. First, we confirm the developer is looking for an example, in question 1. To differentiate between exploratory tool use and focused use, we ask whether they knew which API they want in the example (options (b) and (c)). Question 2 aims to determine if the developer wants a standalone example (which could be created from common patterns [3, 43]) or a production code example (which come from a code repository [18, 26]). As recent research has demonstrated value in creating standalone example templates from production code [3], we made this a multi-select

(1) **Are you looking for an example in Code Search?** (1/) {single select answer}
  (a) No, I'm not looking for an example [exit survey]
  (b) Yes, and I know which API I want
  (c) Yes, but I need to discover an API for a task
  (d) Yes, I'll describe the example
(2) **Ideally, what kind of example do you want?** (2/4) {multi-select answer}
  (a) Standalone examples for how to use an API (e.g., as if the example is from a tutorial or YAQS [14])
  (b) Production code that calls the API (e.g., the example is production code)
  (c) [Other]
(3) **Ideally, what diversity of examples would be most helpful?** (3/4) {multi-select answer}
  (a) Multiple options for how to call an API or solve a problem (e.g., a diverse set of examples)
  (b) The most common way to use an API or solve a problem (e.g., a single best practice example)
  (c) [Other]
(4) **Did you get what you wanted from Code Search today?** (4/4) (same as #3 in Figure 2)

Fig. 3. Code search *Examples* survey Questions for RQ2. The final question is used for RQ4.

(1) **Are you working on a CL?**
  (a) Yes, as an author
  (b) Yes, as a reviewer
  (c) No, I'm not working on a CL [exit survey]
  (d) [Other]
(2) **What is the main reason you are using Code Search right now?**
  (a) I am exploring or **reading** code for understanding (e.g., understand what it does)
  (b) I am looking for a specific code **location** (e.g., where a class is instantiated)
  (c) I need to **discover** how to do something, and would like an example to work from
  (d) I need example code for how to use a **specific API**
  (e) I want to know **who or when** a particular file was modified
  (f) I want to know why something is **failing**, or the side effects of a proposed change
  (g) [**Other**]
(3) **Did you get what you wanted from Code Search today?** (same as #3 in Figure 2)

Fig. 4. Code search *Code Review* survey to answer RQ3. The final question is used for RQ4.

question. Question 3 focuses on how many examples the participants would want to see: multiple (option a), or the single most common way (option b). Finally, question 4 asks about the success of their search experience to partially address RQ4.

3.2.3 RQ3 Survey: Code Review. A relatively common workflow for developers is using Code Search during the code review process when writing new code or when reviewing code. Therefore, we found, developers often switch between a code review tool and Code Search. To understand the challenges with those workflows, we designed the *Code Review* survey in Figure 4. Question 1 asks about whether the user is working on a CL; users who are performing a relevant task are routed to question 2, which asks about why they are using Code Search. This is similar to Question 1 in the *Intent* survey (Figure 2), except option c about examples is split into two, option d and option e,

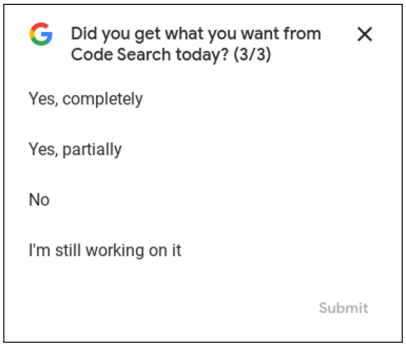


Fig. 5. Survey question from the *Intent* survey, as it appeared to users.

based on whether they are looking to discover an API or use or know which API they want to use. The final question is about satisfaction with their Code Search activity to partially address RQ4.

3.2.4 RQ4 Survey: Success. All surveys were designed to include an explicit success question about whether the developers were satisfied with their use of Code Search that day, like the one shown in Figure 5. This question was included in the original design of the *Intent* surveys but was removed later because the results were so consistently high, we did not want to burden the developers with unnecessary questions (see Section 5.4 for more on this). As it was found that satisfaction with example-seeking behavior was lower than satisfaction with other intents (see Section 4.4), we kept this question for the *Examples* survey. Similarly, to see if satisfaction differed based on whether a developer was authoring or reviewing a code change in a code review process, we kept the question in the *Code Review* survey.

3.3 Survey Deployment

We used the Google's HaTS [27] infrastructure to manage the study deployment. The infrastructure can be configured to display a survey associated with a particular trigger. For any specific trigger, HaTS allows a small, configurable percentage of users receive the survey. The survey questions appeared one at a time in a pop-up on the lower-right portion of the Code Search tool; an example is shown in Figure 5. Here, the question is posed, *"Did you get what you want from Code Search today?"* followed by numbers *(3/3)* indicating this is the third question of three in the survey. Participation was voluntary; participants could exit the survey at any time using the X button.

We ran all surveys between July 2022 and January 2023. A survey trigger raised a flag every time a Code Search user issued a query and then clicked a search result. Of all flags, HaTS randomly sampled 1% for surveying, subject to some constraints: users can only be surveyed once per week, and the participant has not opted out of the surveys. The survey did not deploy for queries issued in the IDE. The same survey trigger was used for all three surveys. The survey was distributed globally without restrictions based on location. Response rates for the surveys are reported in Table 1, with a range of 2.8% to 7.9%. No demographic information was collected.

3.4 Target Population

We studied Google developers who use the Code Search tool, a majority of these are developers. For the surveys, we sampled from the population of users of the standalone Code Search tool, which is a subset of the population described in Section 2.2 (recall the standalone tool accounted for 65% of

Table 1. Survey response rate and duration summaries. Responses includes partial and complete surveys.

Survey	Duration	Responses	Response Rate
Intent	78 days	1286	7.2%
Examples	10 days	188	2.8%
Code Review	8 days	471	7.9%

Table 2. Survey Response Numbers. RQ1 had 1,286 responses. RQ2 had 188 responses, but 117 were irrelevant due to non-example-seeking responses. RQ3 had 471 responses; but 127 were irrelevant. RQ4 is answered by the complete responses only (note: only some of the *Intent* surveys included a question about search success).

		# Responses			
RQ	**Iteration**	**Partial**	**Complete**	**Irrelevant**	**Total**
RQ1	*Intent*	364	922	0	1286
RQ2	*Examples*	15	56	117	188
RQ3	*Code Review*	52	292	127	471
RQ4	*Intent*	0	235	0	235
	Examples	0	56	0	56
	Code Review	0	292	0	292
	Totals	0	583	0	583

the human-written queries to the Code Search backend). The standalone Code Search tool supports approximately 100,000 users on a quarterly basis, and that is our target population.

The number of responses per survey are shown in Table 1. Due to the infrastructure we used for deployment and restrictions on surveying developers, we have reason to assume that these responses are from approximately the same number of developers. For every 10,000 times the trigger was hit, at most 100 developers got the survey. Of those, 2-8 developers submitted a response (based on the response rates). For this reason, and because Google aims to not over-survey the developers, it is likely our data set has only a minimal number of repeated responses from a single developer. However, because of anonymity requirements, we are not able to verify this.

3.5 Data

We received 1,945 survey responses. The mapping of surveys to RQs, is shown in Table 2, along with counts of the responses. We differentiate between complete responses (those that answer all questions in the survey) and partial responses (those that answer at least the first survey question, but not all survey questions). This is because RQ1 is answered by the first two questions in the *Intent* surveys, whereas RQ4 requires the final question from each survey. Therefore, RQ1 can use partial responses while RQ4 cannot. *Irrelevant* responses are those that select an option that leads to an *[exit survey]*, such as option (a) in question 1 for the *Examples* survey (Figure 3) or option (3) in question 1 for the *Code Review* survey (Figure 4). In all, the *Intent* survey received 1,286 responses, the Examples survey received 188 responses, and the Review survey received 471 responses.

4 Results

4.1 RQ1: Code Search Intent

Virtually all developers at Google use Code Search habitually and issue dozens of search queries per week. They search for code for a variety of reasons. The results we collected for RQ1 are from the *Intent* survey and represent the Code Search behaviors of 1,286 developers. A comparison of these results to prior work is in Section 5.1.

Table 3. RQ1: *What is the main reason you are using Code Search right now?* Responses from the current study reflect one response each from 1,286 developers. Responses from the prior work reflect multiple responses across 27 developers.

	Current Study		Prior [33]	
Response	**Responses**	**%**	**Responses**	**%**
What: I am exploring or reading code for understanding (e.g., understand what it does)	472	37%	67	26%
Where: I am looking for a specific code location (e.g., where a class is instantiated)	226	18%	41	16%
How: I need example code for how to do something (e.g., how to use an API, discover an API)	362	28%	87	34%
Who/When: I want to know who or when a particular file was modified	49	4%	22	8%
Why: I want to know why something is failing, or the side effects of a proposed change	120	9%	42	16%
Other / None of the above	57	4%	0	0%
Total	1286	100%	259	100%

Table 4. RQ1: *What do you want to do with the code once you've found it?* Responses from the current study reflect one response each from 981 developers. Responses from the prior work reflect multiple responses across 27 developers.

	Current Study		Prior Study [33]	
Response	**Responses**	**%**	**Responses**	**%**
Continue exploring / learning about the code	233	24%	47	12%
Copy/paste code into my IDE, or work on a CL as an author	306	31%	159	39%
Fix or diagnose bugs or production issues	167	17%	90	22%
Tell a teammate about what I've found	96	10%	–	–
Work on a CL as a reviewer	50	5%	62	15%
Work on a design document or other form of documentation	87	9%	24	6%
Other	42	4%	25	6%
Total	981	100%	407	100%

4.1.1 Code Search Purpose. Table 3 shows answers to the question, *"What is the main reason you are using Code Search right now?"*, alongside the results from prior work, compared in Section 5.1. Overall, in the current study, we observe over the largest percentage of developers, 37%, were seeking to *explore or read code for understanding*. In this way, Code Search is being used to assist with comprehension of the code. The second-largest category is looking for *example code*, with 28% of the responses; we explore this intention more deeply in RQ2. For 18% of the searches, developers were looking for a specific code location. Approximately 9% of developers were seeking an answer to a *why* question about the code, such as *why something is failing*.

Looking at the *[Other]* responses (57, 4%), many developers wrote-in specifics about their tasks. Some descriptions included reading `README` files, reviewing tests to assess current coverage, or looking for appropriate imports. Overall, these examples tended to be rather specific to the developer's context and represent very focused tasks, versus exploratory tasks.

4.1.2 Code Search Next Steps. In response to the question, *"What do you want to do with the code once you've found it?"* we received 981 responses. Results are shown in Table 4, alongside results from the prior work (see Section 5.1).

The top response category was *copy/paste into my IDE, or work on a CL as an author*, representing 31% of the responses. This suggests a tight integration in the workflow between Code Search as a standalone tool and the IDE. This suggests that many developers keep search activity in a separate window from development activity, instead of using the Code Search panel within the IDE.

The next-largest category is to *continue exploring*, suggesting that the survey was triggered in the middle of their activities with the Code Search tool and the task at hand was not done yet. Approximately 17% (167) of the responses were aiming to *fix or diagnose bugs or production issues*, demonstrating the utility of Code Search across a wide range of software development activities.

RQ1 Summary: Developer use Code Search to regularly support their development activities. Exploring or reading code is the most common purpose for search, and the most common downstream intention is to copy/paste the code into their IDE or work on a CL as an author.

4.2 RQ2: Code Search and Examples

Prior work showed that approximately 1/3 of the time developers were using Code Search, they were looking for examples [33]. The results of RQ1 show that the frequency of using code for example-seeking behavior is still high at 28%. In the *Examples* survey, we first asked, *"Are you looking for an example in Code Search?"* For those seeking examples, we received 56 complete response and 15 partial responses, totaling 71. The results for the first three questions are shown in Table 5.

Among those looking for examples, 40 (56%) knew which API they wanted, representing a majority of those seeking examples. Approximately a third (23, 32%) were looking to discover an API for a task whereas the remaining 8 had other explanations for the examples they were seeking. These examples fell into two broad categories: some were looking for an example within a specific location, for example, *"Finding other use cases within my directory to understand proper usage."* Others were looking for example *files* they wanted to copy, for example, *"Looking for an example of a [elided] file containing [elided] in [location]"*.

Table 5. RQ2: Results from all questions in the *Examples* survey.

Are you looking for an example?		
Yes, and I know what API I want	40	56%
Yes, but I need to discover an API	23	32%
Yes, I'll describe the example	8	11%
Total	71	100%

Ideally, what kind of example do you want?		
Standalone examples for how to use an API	25	44%
Production code that calls the API	39	68%
Other	5	9%
Total	57	100%

Ideally, what diversity of examples would be most helpful?		
Multiple options	27	48%
The most common way to use an API	38	68%
Other	1	2%
Total	56	100%

Of those who were looking for examples, 57 (80%) continued on to answer the second question about the kind of example they want. In a multiple-select question, 25 (44%) indicated they want a standalone example for how to use an API whereas a majority (39, 68%) wanted production code examples. Included in these numbers are the 12 participants who said they wanted both.

In terms of example diversity, 27 participants (48%) said they wanted multiple options for how to use an API whereas a majority said they want to see the most common way of using the API (38, 68%). Included in these numbers are the 10 responses who wanted both. If we remove those 10 from the categories, we see that half the participants (28, 50%) want *only* the most common uses. For the participant who answered *other*, they were seeking specifically examples authored by themselves.

RQ2 Summary: Of those seeking examples, most knew what they were looking for and desired to find the most common way of using the API in production code. Free-text responses suggested that where the example was located and who wrote it were considerations in their search journey.

4.3 RQ3: Code Search and Code Review

As shown in Section 4.1.2, developers frequently use Code Search while working on a CL as an author (31%) or reviewer (5%). Of the 471 respondents to the first question in the *Code Review* survey, 295 answered *yes* indicating that they were working on a CL and continued to answer the second question. Of those, 242 (82%) were working on a CL as an author, 44 (15%) were working in a CL as a reviewer, and 9 (3%) were doing other work. These data are shown in the *Sum* column of Table 6. While developers use Code Search during code review, far more often Code Search is used during code authoring.

To understand more about Code Search tool usage code authoring and code review, we do a two-way analysis of the first two questions in the *Code Review* survey, shown with question 1 in the rows and question 2 in the columns of Table 6. The percentages for each cell are row percentages. That is, there were 44 participants who were reviewing a CL (*Yes, as a reviewer*). Of these, 19 (43%) were *Reading* code for understanding and 12 (27%) were looking for a particular *Location* in code.

When working on a CL *as a reviewer*, most of the Code Search behavior is centered around reading for comprehension or looking for a specific location in code (70% total) versus example-seeking behavior (9% + 16% = 25%). When authoring CLs, developers are more often reading or looking for a code location (49% total) or example-seeking (27% + 11 % = 38%). We see that example seeking is more common during authoring than reviewing, which confirms our expectations.

RQ3 Summary: Reviewing CLs is more associated with exploration activities in Code Search, while authoring CLs is more associated with example-seeking behavior.

Table 6. RQ3: *Are you working on a CL?* (rows) vs. *What is the main reason you are using Code Search right now?* (columns). Numbers are reported in raw and in row percentages. For example, of those who were working on CLs as an author (242), 73 were exploring or *reading* code, representing 30%. (295 total Responses)

			Examples					
	Reading	Location	Discover	Specific API	Who or when	Failing	Other	Sum
Yes, as author	73 (30%)	45 (19%)	66 (27%)	26 (11%)	7 (3%)	17 (7%)	8 (3%)	242 (82%)
Yes, as reviewer	19 (43%)	12 (27%)	4 (9%)	7 (16%)	1 (2%)	0 (0%)	1 (2%)	44 (15%)
Other	4 (44%)	2 (22%)	–	–	1 (11%)	2 (22%)	–	9 (3%)
Total	96 (33%)	59 (20%)	70 (24%)	33 (11%)	9 (3%)	19 (6%)	9 (3%)	295 (100%)

Table 7. RQ4: *Did you get what you wanted out of Code Search today?* Responses to this question from all surveys. Column percentages add to 100%.

Response	Intent	Example	Review	Total
Yes, completely	115 (49%)	17 (30%)	151 (52%)	283 (49%)
Yes, partially	37 (16%)	13 (23%)	58 (20%)	108 (19%)
Still working	78 (33%)	25 (45%)	79 (27%)	182 (31%)
No	5 (2%)	1 (2%)	4 (1%)	10 (2%)
Total	235 (100%)	56 (100%)	292 (100%)	583 (100%)

4.4 RQ4: Code Search Satisfaction

Overall, developers were satisfied with the Code Search tool. Table 7 shows the responses to the satisfaction questions from each of the surveys. In all, we have 583 survey responses about satisfaction with Code Search. Of these, 283 (49%) were completely satisfied and 108 (19%) were partially satisfied. The survey was triggered after a result was clicked, and this happened to be in the middle of their task for 182 respondents (31%) who indicated they were still working and did not know yet if they were satisfied. Only 10 (2%) respondents were unsatisfied.

From the *Intent* surveys, only 5 of the 235 cases, developers reported 'no' to being satisfied with Code Search. Looking deeper into the data, in two cases, developers said they were searching because "*I need example code for how to do something*" (per question 1 of the survey). In two more cases, the developers responded that they were searching because "*I want to know why something is failing, or the side effects of a proposed change*". The last case was a developer who was dissatisfied with having their workflow interrupted by surveys. In the *Examples* survey, only 1 in 56 of the responses from people searching for examples were dissatisfied with Code Search. Overall, there is extremely high satisfaction. Even if example-seeking behavior corresponds to a lack of satisfaction, it is so small that it would see Code Search is performing well in that capacity.

In Table 7, the column *Examples* represents responses from the *Examples* survey. By and large, participants were satisfied with Code Search when it comes to example seeking. Only one participant out of 56 said they were not satisfied with Code Search. A majority (30, 53%) were either completely or partially satisfied. Given that the trigger for this survey was after a single search and result click, we expected (and received) many responses indicating their journey with Code Search was incomplete; 25 (45%) reported to still be working. Still, those seeing examples are somewhat less often completely satisfied with Code Search and more likely to be partially satisfied or to be still working on their task.

For the *Code Review* survey, seen in the *Review* column, the distribution looks similar to the *Intent* survey. When we look at satisfaction based on whether a developer is working on the CL as a review versus as an author, there were no notable differences in satisfaction.

We further investigated whether satisfaction was different for any of the intents declared in question 1 of the *Intent* survey. While most of the results were rather unremarkable, when the main reason for search is to find a code *location*, satisfaction with Code Search was the highest. Developers were completely satisfied 56% of the time they were looking for a location, or else, they were still working on the task at hand (31%). This suggests that for this purpose behind Code Search, the tool is working quite well.

RQ4 Summary: Code Search users are very satisfied with the tool, though those looking for examples are somewhat less satisfied than others. Code Search users looking for a code location were most likely to be completely satisfied.

4.5 Threats to Validity

There are a few threats to validity for this study. First of all, a threat to external validity is that this study focuses on one particular Code Search tool at one particular company (Google). The Code Search experience at Google is unique in that most developers at Google are working within (and hence using it to search over) an extremely large monolithic repository [19]. Developers at Google also use a custom developer tool stack [19] and so may behave differently than developers at other companies using a different set of tools. Replication is needed to understand if that is the case.

Another threat to external validity is that we are limited to developers who chose to fill in the survey, and since we did not collect demographic information (due to a limit on the number of questions possible in HaTS) we cannot tell if the responses are biased towards a particular demographic group. We also include answers from developers who only partially completed some surveys. These threats are tempered by the large number of developers who responded.

A threat to internal validity is the specific survey questions we used and the responses we received. Although our survey questions and response options are grounded in prior work and we refined the surveys throughout the course of the study, a different formulation of questions or responses could provide new perspectives not captured here.

Another internal threat is that we only surveyed queries to the standalone Code Search tool, which misses 35% of the queries to the Code Search backed that come from the IDE. This may create the illusion of a shift in search intents when comparing against the prior work.

Lastly, a threat to validity is the rapidly changing landscape of developer tools with increased focus on tooling powered by LLMs (discussed in Section 5). The surveys in this paper represent a pre-LLM context. The logs analysis, on the other hand, encompasses before and after the inclusion of AI in the developer workflow, and we did not find any signal indicating usage changes for Code Search. This may change in the future as AI becomes more sophisticated and entrenched in workflows. As we move into this new future the role of code search in software development is changing. Nonetheless, we feel that now is the time to review where we are so we can consider the implications of existing search behavior while designing this new wave of tooling.

5 Discussion

The discussion includes an in-depth look at our results compared to prior work, implications of this work for future research on code search and LLMs, and lessons learned.

5.1 Comparison against Prior Work

In the time between the prior work [33] and our survey, approximately ten years passed. RQ1 was asked in prior work [33] and also by us. In this section, we look at the differences between the prior results and our results for RQ1.

5.1.1 Code Search Purpose. Table 3 shows summaries of the answers to the question, *"What is the main reason you are using Code Search right now?"*. In most cases, the frequency of responses per category in our survey versus the prior work are close, but there are exceptions. Developers in our survey are more often using Code Search to explore or read code for understanding compared to the prior work. Using a test of two-proportions to determine if these percentages are significant, we find that $p_{what,prior} = 67/259$ and $p_{what,new} = 472/1286$ with *what* referring to the proportion of responses seeking exploring or reading code, *prior* referring to prior results [33] and *new* referring to the current study. The null hypothesis $H_0 : p_{what,prior} = p_{what,new}$ is rejected at $\alpha = 0.01$ with $p = 0.00109$. Therefore, we find a significant shift in the behavior, where more developers are going to Code Search to read or understand code than before.

We also find a significant decrease in the number of participants who want to know *why* something is failing. Previously, it was found that 16% were asking questions of this nature, versus 9% in the current survey. Using a test of two-proportions to determine if these percentages are significant, we find that $p_{why,prior} = 42/259$ and $p_{why,new} = 120/1286$. The null hypothesis $H_0 : p_{why,prior} = p_{why,new}$ is rejected at $\alpha = 0.01$ with $p = 0.00143$. Therefore, we find a significant shift in this behavior, likely because Code Search behavior related to debugging has moved within the IDE and was not observed with our survey.

Also of note is a lower percentage of users seeking examples using Code Search as compared to prior work; the percentage dropped 6% from 34% to 28%. This may reflect a change in the tools available to developers for examples (e.g., internal search tools, or documentation, may have improved). Using a test of two-proportions, we find that $p_{ex,prior} = 87/259$ and $p_{ex,new} = 362/1286$ with *ex* referring to the proportion of responses seeking examples. The null hypothesis $H_0 : p_{ex,prior} = p_{ex,new}$ is not rejected with $p = 0.09206$. Therefore, we conclude there has **not** been a significant shift in example-seeking behavior based on the two studies.

5.1.2 Code Search Next Steps. Table 4 shows summaries of answers to the question, *"What do you want to do with the code once you've found it?"*. While the question in prior work was multiple-select, only 13 surveys had multiple items selected, representing approximately 3% of the responses, so the authors felt that comparison is still valuable. We expand the number of responses to 407 so the percentages add to 100%.

There is a sizeable shift in the number of responses that report to be *exploring / learning about the code*, where the current survey has significantly more (test of two-proportions, $p < 0.001$). The goal of copy-pasting code or working on a CL as an author while using Code Search has gone down significantly (test of two-proportions, $p = 0.005$), as has working on a CL as a reviewer (test of two-proportions, $p < 0.001$). Together, these suggest a change in how the standalone Code Search tool is being used as more of a learning tool and less of a tool to assist with the day-to-day of writing and reviewing code. This shift, however, may not reflect overall Code Search behavior, but rather reflects the impact of a change in workflow. As we did not want to interrupt developers with surveys during active development, we did not capture search intent while developers were actively within the IDE. Additionally, as the newer IDEs have gained in popularity, developers who migrated from more simplistic IDE environments (e.g., pure text editors) to the internal tooling may reduce the need for copy and paste from Code Search into the IDE environment. Therefore, we can only speculate that those Code Search queries that desired to copy-paste the results, or that were working to debug code, moved from the standalone tool to the IDE. The impact here is that it creates the illusion of a shift in behavior when looking at surveys of only the standalone tool usage.

5.1.3 Comparison Summary. The results of this investigation into current Code Search usage are rather consistent with the prior work. We confirm that the prior work included a rather representative sample of Google developers from the perspective of intent. We observe a subtle shift toward using Code Search more for reading and exploring code; it is unclear if that shift is due to the 10-year gap between the studies, the increased number of participants, or something else. Changes in the search intents seem to be easily explained by changes to the developer workflows and therefore appear consistent overall.

5.2 Code Search and LLMs

During the time the study was conducted, LLMs such as ChatGPT [29] and Gemini [31] were just emerging as part of developers' workflow, primarily for assisting code completion [8, 16] and code generation [20, 25]. As discussed in Section 2.2, Code Search usage at Google has remained steady

despite the emergence of such technologies. Yet, their integration seems unavoidable. Identifying the most productive ways to integrate AI into the Code Search workflow is an open challenge.

A research agenda to address such challenge should aim to catalog and discern the use cases under which each technology excels. For example, we speculate that for tasks that require precise answers, with high recall, and guarantees of no hallucination (e.g., find production code containing a deprecated API call, find a specific location in source code), Code Search already offers a high degree of satisfaction (Section 4.4). Meanwhile, for more exploratory activities (e.g., find simple code examples, find other APIs to perform the task) where there is some tolerance for lower recall or lower precision, Code Search performs well, but there are opportunities to innovate.

Identifying ways in which these technologies can be synergistically combined is likely a fruitful pursuit. LLMs could be used to find bodies of relevant code that are further filtered by Code Search, for example, to improve the usability of a Code Search tool itself by enabling fuzzy matching or suggesting new places to explore in an existing code base. Other examples include refactoring a piece of production code to fit a given context or providing alternative paths to tasks that are currently done via a Code Search tool. When developers want standalone examples for how to use an API (44%, as in Table 3), generative AI may provide faster or simpler examples. Code Search could be used as an initial filter to find real examples, and those examples could be simplified or refactored through AI. If a developer wants to find an API for a task and the API is not known, an AI agent might be able to help. On the other hand, when developers want examples from production code (68%, Table 3), existing tools may be enough. One could imagine integrating generated and existing code results within the same search surface.

It is also reasonable to expect that these technologies, just like it is occurring with general search engines, will evolve towards having a single interface. Already, researchers are looking for which code search techniques are most effective for which search intentions [41]. Defining that interface and how it should interact with a code base and modern IDEs should also be part of that agenda.

5.3 Code Search Beyond the Code

The Code Search tool we study is focused explicitly on searching over code repositories at Google and no other data sources that may contain information about source code (e.g., queries to an internal Q&A site). In our survey, this means we miss some relevant searching behavior, such as searches performed in the IDE. We chose to focus on the Code Search tool explicitly as those most clearly represent intentional code searches. For search tools that interact with heterogeneous data sources (e.g., API documentation, designs, source code, requirements), it would be interesting to see what the breakdowns look like in terms of search intents and search satisfaction. For example, if people are searching for examples, are they more satisfied by what is presented in source code or in API documentation? Or what about summaries generated from the source code [17], are those better than the source code itself for learning? Is there a relationship between search satisfaction and document quality [36]? These questions remain open.

Beyond the existing data sources, as AI-generated code becomes more ubiquitous, the prompts used for that generation are important, and prompt engineering is becoming part of the developer workflow. Prompt and code search integration could take the form of new prompt languages for searching over existing code, or the need to search over the prompts themselves. One could imagine prompts themselves being yet another document in code search across heterogeneous data sources.

Correlating search intent and satisfaction with the types of resources accessed – API documentation, design documents, source code, test code, and Q&A forums, LLM prompts, for example – would perhaps shed light on the software development search process more generally. However, replicating our survey methodology may cause undue frustration to developers by breaking their flow, and the impact of such interruptions would need to be carefully monitored.

5.4 Lessons Learned from Studies of Developers at Scale

To conduct our study, we had access to thousands of developers operating under a rather uniform workflow and set of tools, working across a variety of projects, programming languages, and teams. Through the process, we learned some valuable lessons.

First, having at least one author that is part of the organization does not just open doors and facilitates the implementation of the study, but can help to design the study based on the organizational context. Further, the insider perspective can contribute to interpret the data and identify suspicious data points caused by implementation errors or nuances in telemetry.

Second, studies of this magnitude cannot be rushed. They require several iterations to mature, to get the surveys with the right constructs, and the delivery and retrieval mechanisms to work as expected. Incrementally building and deploying those surveys and mechanisms helped us enormously to gain internal credibility and reduce the impact of surprises. However, the extensive planning and iterative deployment could not overcome some unexpected events (e.g., invalidating tools updates, move of critical personnel), which often required restarting some phases of the study, further delaying the study for reasons beyond our control.

Third, a question asked thousands of times across a body of developers has a significant compounding cost in developers' time. This cost is magnified if that question is not well constructed causing misleading responses or fails to add value through its associated finding. In this study, for example, we balanced some loss of data by removing question 4 from the Intent surveys against the increase in developer effort to answer yet another survey question. While developers at Google have shown resilience to small interruptions in terms of their flow [5], we still wanted to minimize any interruptions caused by our work. Thus, studies at this scale must balance the luxury of having access to data at scale with the cost associated with such access.

6 Related Work

Here we describe recent related research on code search.

6.1 Studies on Code Search

Code search research is gaining in momentum [10], with many surveys on code search usage emerging. The methodologies include surveys, log analyses, or both.

The prior work targeting Google developers used both surveys and logs [33]. Through the logs analysis, it found that developers searched on average 12 times a day (median of 6). Other work using log analysis looks deeper into the query intent, finding that users frequently use code search tools to find code examples [30]. In a study with tens of thousands of users on the Lingma code search tool by Alibaba, researchers found a low average of 0.02 search events per developer per day using log analysis [23]. Follow up surveys, however, found the search rate to be higher, with most participants conducting 3–5 queries per day. Other research using code search log analysis has additionally focused on the content of queries, finding them to frequently include method names, structural patterns, and keywords [30].

The work that inspired ours [33] included surveys of developer intentions, which informed the design of our surveys, found that developers search for a variety of purposes (e.g., to find example code, to fix a problem, to learn about code). In a separate survey of 69 developers [34], researchers found that the most common motivations for code search were defect repair (20%), code reuse (15%), and program understanding (14%), which map to Question 2 of the survey in Figure 2, specifically options *c*, *b*, and *a*, respectively. Another survey of 235 software engineers found that reusing code snippets is a common motivation being code search on the web [40], and we explore similar motivations in the survey about code examples (Figure 3).

Code search success is tricky to measure, but researchers have tried. Result clicks might not be the most reliable [33], as often the preview of a code search result is enough to solve the problem at hand. Considering the Koders [2] tool, researchers measured success by whether code is downloaded, finding that 25% to 60% of code search queries were successful [1]. When considering the intention of code reuse, that success criteria might be somewhat accurate. For other use cases, such as debugging or learning about code, it would very likely be an underestimate of search success.

With a specific focus on API search, which takes an API name as input and returns representative code examples, researchers have achieved high developer satisfaction at scale [3, 15]. Exempla Gratis [3] is a tool that generated simplified examples for how to use APIs. When it was compared against code search results and hand-written examples from a popular programming website, the generated examples were preferred by professional developers in 97% of cases. API search using graph embeddings and clustering also received positive developer feedback, with 95% of developers finding them useful 95% and 38% finding them very useful [15].

6.2 AI and Code Search

AI-driven support in development workflows is rapidly advancing, and Google is no exception. Opportunities abound, including in the code comprehension space when developers need to learn a new platform, infrastructure, framework, or technology [9]. Research in academia and other companies is also focusing on this. For example, the Bing Developer Assistant tool from Microsoft searches and recommends code example mined from public software repositories (such as GitHub) and web pages (such as Stack Overflow) for users to improve their programming productivity [42]. Copilot from GitHub searches for code based on context signals from code comments [12].

Outside of industry, other research has compared different approaches to code search against different use cases, with the insight that there may not be one solution for all of the code search use cases. The researchers found that for reusing code, deep learning methods are more effective. For fixing bugs or learning about APIs, information retrieval methods are more useful [41]. By improving the mapping between the retrieved code and the query through deep learning and fuzzy matching [22], other researchers have achieved improvements. A combination of deep learning and information retrieval strategies has also been successful [7].

7 Conclusion

A decade ago, researchers highlighted how developers use Code Search in practice at Google. Fast forwarding to the present day, we both revisited this prior study to see what has changed and dug deeper into how developers use Code Search when seeking examples or performing reviews. We find that AI has not reduced the need for Code Search, at least, not yet. The standalone Code Search tool is still used frequently, especially when developers are exploring or reading code. Satisfaction with the tool is very high. When seeking examples in Code Search, developers typically know what API they are looking for and want to review production code. We find that Code Search is often focused on reading code for understanding. As AI tool support continues to emerge and increase in maturity, these conclusions should be revisited should the study be replicated again.

Currently, we emphasize that these results speak to the importance of the Code Search experience when developing software. The way that developers are using Code Search to explore and read code has implications for innovations in code summarization and comprehension. Even as LLMs are increasingly used to generate code examples, there are still many tasks, such as identifying code locations, that continue to fall to Code Search.

Data Availability

The logs and survey responses we summarize are subject to privacy constraints, including but not limited to focusing on logs that are associated with work purposes and not reporting out individual data without explicit permission to do so. Therefore, the raw data are not publicly available.

Acknowledgements

This work is funded in part by NSF SHF #1749936.

References

[1] Sushil Bajracharya and Cristina Lopes. 2009. Mining search topics from a code search engine usage log. In *2009 6th IEEE International Working Conference on Mining Software Repositories*. 111–120. https://doi.org/10.1109/MSR.2009.5069489

[2] Sushil Krishna Bajracharya and Cristina Videira Lopes. 2012. Analyzing and mining a code search engine usage log. *Empirical Software Engineering* 17 (2012), 424–466. https://doi.org/10.1007/s10664-010-9144-6

[3] Celeste Barnaby, Koushik Sen, Tianyi Zhang, Elena Glassman, and Satish Chandra. 2020. Exempla gratis (E.G.): code examples for free. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1353–1364. https://doi.org/10.1145/3368089.3417052

[4] Earl T. Barr, Yuriy Brun, Premkumar Devanbu, Mark Harman, and Federica Sarro. 2014. The plastic surgery hypothesis. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Hong Kong, China) *(FSE 2014)*. Association for Computing Machinery, New York, NY, USA, 306–317. https://doi.org/10.1145/2635868.2635898

[5] Adam Brown, Alison Chang, Ben Holtz, and Sarah D'Angelo. 2023. Developer Productivity for Humans, Part 6: Measuring Flow, Focus, and Friction for Developers. *IEEE Software* 40, 6 (2023), 16–21. http://doi.org/10.1109/MS.2023.3305718

[6] Satish Chandra, Michael Pradel, and Kathryn T. Stolee. 2024. Dagstuhl Seminar 24172: Code Search. (2024). https://doi.org/10.4230/DagRep.14.4.108

[7] Junkai Chen, Xing Hu, Zhenhao Li, Cuiyun Gao, Xin Xia, and David Lo. 2024. Code Search is All You Need? Improving Code Suggestions with Code Search. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 73, 13 pages. https://doi.org/10.1145/3597503.3639085

[8] Matteo Ciniselli, Nathan Cooper, Luca Pascarella, Antonio Mastropaolo, Emad Aghajani, Denys Poshyvanyk, Massimiliano Di Penta, and Gabriele Bavota. 2021. An empirical study on the usage of transformer models for code completion. *IEEE Transactions on Software Engineering* 48, 12 (2021), 4818–4837. https://doi.org/10.1109/TSE.2021.3128234

[9] Sarah D'Angelo, Ambar Murillo, Satish Chandra, and Andrew Macvean. 2024. What do developers want from AI? *IEEE* (2024). https://www.computer.org/csdl/magazine/so/2024/03/10493171/1VTvfCwygyk

[10] Luca Di Grazia and Michael Pradel. 2023. Code Search: A Survey of Techniques for Finding Code. *Comput. Surveys* 55, 11, Article 220 (feb 2023), 31 pages. https://doi.org/10.1145/3565971

[11] Denae Ford, Alisse Harkins, and Chris Parnin. 2017. Someone like me: How does peer parity influence participation of women on stack overflow?. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 239–243. https://doi.org/10.1109/VLHCC.2017.8103473

[12] Nat Friedman. 2021. Introducing GitHub Copilot: your AI pair programmer. (2021). https://github.blog/news-insights/product-news/introducing-github-copilot-ai-pair-programmer/

[13] Alexander Frömmgen, Jacob Austin, Peter Choy, Nimesh Ghelani, Lera Kharatyan, Gabriela Surita, Elena Khrapko, Pascal Lamblin, Pierre-Antoine Manzagol, Marcus Revaj, Maxim Tabachnyk, Daniel Tarlow, Kevin Villela, Dan Zheng, Satish Chandra, and Petros Maniatis. 2024. Resolving Code Review Comments with Machine Learning. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. https://doi.org/10.1145/3639477.3639746

[14] Google. [n. d.]. Yet Another Question System. ([n. d.]). A Google-internal question and answer site, similar to Stack Overflow..

[15] Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2020. CodeKernel: a graph kernel based approach to the selection of API usage examples. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering* (San Diego, California) *(ASE '19)*. IEEE Press, 590–601. https://doi.org/10.1109/ASE.2019.00061

[16] Qi Guo, Junming Cao, Xiaofei Xie, Shangqing Liu, Xiaohong Li, Bihuan Chen, and Xin Peng. 2024. Exploring the potential of chatgpt in automated code refinement: An empirical study. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13. https://doi.org/10.1145/3597503.3623306

[17] Emily Hill, Lori Pollock, and K Vijay-Shanker. 2011. Improving source code search with natural language phrasal representations of method signatures. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 524–527. https://doi.org/10.1109/ASE.2011.6100115

[18] Reid Holmes and Gail C. Murphy. 2005. Using structural context to recommend source code examples. In *Proceedings of the 27th International Conference on Software Engineering* (St. Louis, MO, USA) *(ICSE '05)*. Association for Computing Machinery, New York, NY, USA, 117–125. https://doi.org/10.1145/1062455.1062491

[19] Ciera Jaspan, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, and Emerson Murphy-Hill. 2018. Advantages and Disadvantages of a Monolithic Codebase. In *International Conference on Software Engineering, Software Engineering in Practice track (ICSE SEIP)*. https://doi.org/10.1145/3183519.3183550

[20] Kailun Jin, Chung-Yu Wang, Hung Viet Pham, and Hadi Hemmati. 2024. Can ChatGPT Support Developers? An Empirical Evaluation of Large Language Models for Code Generation. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 167–171. https://doi.org/10.1145/3643991.3645074

[21] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. 2013. What help do developers seek, when and how?. In *2013 20th working conference on reverse engineering (WCRE)*. IEEE, 142–151. http://doi.org/10.1109/WCRE.2013.6671289

[22] Chao Liu, Xin Xia, David Lo, Zhiwe Liu, Ahmed E. Hassan, and Shanping Li. 2021. CodeMatcher: Searching Code Based on Sequential Semantics of Important Query Words. *ACM Transactions on Softwware Engineering Methodology* 31, 1, Article 12 (sep 2021), 37 pages. https://doi.org/10.1145/3465403

[23] Chao Liu, Xindong Zhang, Hongyu Zhang, Zhiyuan Wan, Zhan Huang, and Meng Yan. 2024. An Empirical Study of Code Search in Intelligent Coding Assistant: Perceptions, Expectations, and Directions. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (Porto de Galinhas, Brazil) *(FSE 2024)*. Association for Computing Machinery, New York, NY, USA, 283–293. https://doi.org/10.1145/3663529.3663848

[24] Yue Liu, Thanh Le-Cong, Ratnadira Widyasari, Chakkrit Tantithamthavorn, Li Li, Xuan-Bach D Le, and David Lo. 2024. Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–26. https://doi.org/10.1145/3643674

[25] Zhijie Liu, Yutian Tang, Xiapu Luo, Yuming Zhou, and Liang Feng Zhang. 2024. No need to lift a finger anymore? assessing the quality of code generation by chatgpt. *IEEE Transactions on Software Engineering* (2024). https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3392499

[26] Sifei Luan, Di Yang, Celeste Barnaby, Koushik Sen, and Satish Chandra. 2019. Aroma: code recommendation via structural code search. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 152 (2019), 28 pages. https://doi.org/10.1145/3360578

[27] Hendrik Müller and Aaron Sedley. 2014. HaTS: large-scale in-product measurement of user attitudes & experiences with Happiness Tracking Surveys. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design* (Sydney, New South Wales, Australia) *(OzCHI '14)*. Association for Computing Machinery, New York, NY, USA, 308–315. https://doi.org/10.1145/2686612.2686656

[28] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 97, 13 pages. https://doi.org/10.1145/3597503.3639187

[29] OpenAI. 2022. Introducing ChatGPT. (2022). https://openai.com/index/chatgpt/

[30] Oleksandr Panchenko, Hasso Plattner, and Alexander Zeier. 2011. What do developers search for in source code and why. In *Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation* (Waikiki, Honolulu, HI, USA) *(SUITE '11)*. Association for Computing Machinery, New York, NY, USA, 33–36. https://doi.org/10.1145/1985429.1985438

[31] Sundar Pinchai and Demis Hassabis. 2023. Introducing Gemini: our largest and most capable AI model. (2023). https://blog.google/technology/ai/google-gemini-ai/

[32] Md Masudur Rahman, Jed Barson, Sydney Paul, Joshua Kayani, Federico Andrés Lois, Sebastián Fernandez Quezada, Christopher Parnin, Kathryn T. Stolee, and Baishakhi Ray. 2018. Evaluating how developers use general-purpose web-search for code retrieval. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 465–475. https://doi.org/10.1145/3196398.3196425

[33] Caitlin Sadowski, Kathryn T Stolee, and Sebastian Elbaum. 2015. How developers search for code: a case study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 191–201. https://doi.org/10.1145/2786805.2786855

[34] S.E. Sim, C.L.A. Clarke, and R.C. Holt. 1998. Archetypal source code searches: a survey of software developers and maintainers. In *Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No.98TB100242)*. 180–187. https://doi.org/10.1109/WPC.1998.693351

[35] Kathryn T Stolee, Sebastian Elbaum, and Daniel Dobos. 2014. Solving the search for source code. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 23, 3 (2014), 1–45. https://doi.org/10.1145/2581377

[36] Christoph Treude, Justin Middleton, and Thushari Atapattu. 2020. Beyond accuracy: assessing software documentation quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1509–1512. https://doi.org/10.1145/3368089.3417045

[37] Shangwen Wang, Bo Lin, Zhensu Sun, Ming Wen, Yepang Liu, Yan Lei, and Xiaoguang Mao. 2023. Two Birds with One Stone: Boosting Code Generation and Code Search via a Generative Adversarial Network. *Proc. ACM Program. Lang.* 7, OOPSLA2, Article 239 (oct 2023), 30 pages. https://doi.org/10.1145/3622815

[38] Hyrum Wright, Titus Delafayette Winters, and Tom Manshreck. 2020. *Software Engineering at Google.* O'Reilly.

[39] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E Hassan, and Zhenchang Xing. 2017. What do developers search for on the web? *Empirical Software Engineering* 22 (2017), 3149–3185. https://doi.org/10.1007/s10664-017-9514-4

[40] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E. Hassan, and Zhenchang Xing. 2017. What do developers search for on the web? *Empirical Software Engineering* 22, 6 (dec 2017), 3149–3185. https://doi.org/10.1007/s10664-017-9514-4

[41] Shuhan Yan, Hang Yu, Yuting Chen, Beijun Shen, and Lingxiao Jiang. 2020. Are the Code Snippets What We Are Searching for? A Benchmark and an Empirical Study on Code Search with Natural-Language Queries. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 344–354. http://doi.org/10.1109/SANER48275.2020.9054840

[42] Hongyu Zhang, Anuj Jain, Gaurav Khandelwal, Chandrashekhar Kaushik, Scott Ge, and Wenxiang Hu. 2016. Bing developer assistant: improving developer productivity by recommending sample code. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA) *(FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 956–961. https://doi.org/10.1145/2950290.2983955

[43] Hao Zhong, Tao Xie, Lu Zhang, Jian Pei, and Hong Mei. 2009. MAPO: Mining and recommending API usage patterns. In *ECOOP 2009–Object-Oriented Programming: 23rd European Conference, Genoa, Italy, July 6-10, 2009. Proceedings 23*. Springer, 318–343. https://doi.org/10.1007/978-3-642-03013-0_15

Received 2024-09-12; accepted 2025-01-14