

# What do developers search for on the web?

Xin Xia<sup>1,2</sup> · Lingfeng Bao<sup>1</sup> · David Lo<sup>3</sup> ·  
Pavneet Singh Kochhar<sup>3</sup> · Ahmed E. Hassan<sup>4</sup> ·  
Zhenchang Xing<sup>5</sup>

Published online: 9 April 2017  
© Springer Science+Business Media New York 2017

**Abstract** Developers commonly make use of a web search engine such as Google to locate online resources to improve their productivity. A better understanding of what developers search for could help us understand their behaviors and the problems that they meet during the software development process. Unfortunately, we have a limited understanding of what developers frequently search for and of the search tasks that they often find challenging. To address this gap, we collected search queries from 60 developers, surveyed 235 software engineers from more than 21 countries across five continents. In particular, we asked

---

Communicated by: Emerson Murphy-Hill

---

✉ Lingfeng Bao  
lingfengbao@zju.edu.cn

Xin Xia  
xxia@zju.edu.cn; xxia02@cs.ubc.ca

David Lo  
davidlo@smu.edu.sg

Pavneet Singh Kochhar  
kochharps.2012@smu.edu.sg

Ahmed E. Hassan  
ahmed@cs.queensu.ca

Zhenchang Xing  
zhenchang.xing@anu.edu.au

- <sup>1</sup> College of Computer Science and Technology, Zhejiang University, Hangzhou, China
- <sup>2</sup> Department of Computer Science, University of British Columbia, Vancouver, BC, Canada
- <sup>3</sup> School of Information Systems, Singapore Management University, Singapore, Singapore
- <sup>4</sup> School of Computing, Queen's University, Kingston, ON, Canada
- <sup>5</sup> Research School of Computer Science, Australian National University, Canberra, Australia

our survey participants to rate the frequency and difficulty of 34 search tasks which are grouped along the following seven dimensions: general search, debugging and bug fixing, programming, third party code reuse, tools, database, and testing. We find that searching for explanations for unknown terminologies, explanations for exceptions/error messages (e.g., HTTP 404), reusable code snippets, solutions to common programming bugs, and suitable third-party libraries/services are the most frequent search tasks that developers perform, while searching for solutions to performance bugs, solutions to multi-threading bugs, public datasets to test newly developed algorithms or systems, reusable code snippets, best industrial practices, database optimization solutions, solutions to security bugs, and solutions to software configuration bugs are the most difficult search tasks that developers consider. Our study sheds light as to why practitioners often perform some of these tasks and why they find some of them to be challenging. We also discuss the implications of our findings to future research in several research areas, e.g., code search engines, domain-specific search engines, and automated generation and refinement of search queries.

**Keywords** Search task · Understanding · Empirical study · Survey

## 1 Introduction

In modern software development, developers often refer to web search engines as they seek pertinent information from large amounts of online resources such as online tutorials, technology blogs, API documents, social media posts, etc. Search engines such as Google,<sup>1</sup> Bing,<sup>2</sup> and Baidu<sup>3</sup> have become one of the most popular and important tools for developers to complete different types of software engineering tasks, e.g., fixing unexpected bugs/exceptions, and understanding unfamiliar code or APIs. Our previous study found that developers issue more than 20 search queries that are related to software development everyday (Bao et al. 2015a).

Developers may search for many things for different purposes during different phases of a project. During the requirement analysis phase, developers may search for explanations of terminologies, or relevant books and tutorials to help them understand business logic and gain necessary background knowledge. During the development phase, developers may search for suitable third party libraries, or the usage of specific APIs. During the maintenance phase, developers may search for solutions of bugs. It may be easy to get relevant results for some of the search tasks (e.g., search for explanations for unknown terminologies), but it may be hard for others (e.g., search for database optimization solutions). Moreover, developers might use different search engines such as Google and Bing, or some domain-specific search engines such as the search engine of Stack Overflow.

A better understanding of what developers search for could help researchers better understand some of the problems that are faced by developers throughout the software development process. For example, if we find that developers frequently search for solutions for a specific type of bugs (e.g., performance bugs), then we can deduce that such types of bugs maybe hard to fix, that there might not exist good tool support for such bugs, that developers are not familiar with such types of bugs, or that finding standard solutions for such

---

<sup>1</sup><http://www.google.com>

<sup>2</sup><http://www.bing.com>

<sup>3</sup><http://www.baidu.com>

bugs is difficult. Thus, the software engineering research community should design more tools to help developers avoid, detect, and possibly recommend fixes to these bugs. Notice that the types of bugs that appear frequently in a bug repository do not necessarily correlate to the bugs that developers frequently search. Some types of bugs are easy to fix (e.g., variable assignment error bugs) which do not require one to search for solutions online, although such bugs appear frequently in a bug repository (in many ways we believe that developers are likely to search for solution for rare bugs instead of commonly occurring types of bugs).

In the literature, a number of empirical studies have investigated how developers perform code search (Sim et al. 1998; Sadowski et al. 2015; Bajracharya and Lopes 2009, 2012). Sim et al. (1998) found that the most common reasons for code search include: defect repair, code reuse, program understanding, and feature addition. Sadowski et al. (2015) conducted a study at Google to learn how developers search for code, in what contexts are code search tools used, and what are the properties of search queries. Bajracharya and Lopes (2009, 2012) performed a topic modeling analysis on a year long usage log of Koders, one of the major commercial code search engines, to understand what users of code search engines are looking for. Prior work has focused primarily on searching for code. Yet, code search is only one of the search tasks that developers perform. Developers search for many other things. For example, developers may search for best industrial practices, or how to use a particular tool.

In this paper, we enumerate the frequency and difficulty of the different web search tasks that developers perform. We first collected 60 developers' web search queries for two weeks by using our ACTIVITYSPACE (Bao et al. 2015a, b), and analyzed them to understand common search tasks. Next, we interviewed 12 senior software engineers from two software companies. Though our analysis of the collected queries and interviews, we identified a set of 34 search tasks that are grouped into several categories: general search, debugging and bug fixing, programming, third party code reuse, tools, database, and testing. We then surveyed 235 software engineers across the globe (i.e., 21 countries) who worked in various small to large companies and organizations (including: Microsoft, Alibaba, and Baidu.) or were among the top contributors in GitHub. In our survey, we asked participants to rate the frequency and difficulty of our identified 34 search tasks and to provide their rationale for rating some search tasks as frequent or difficult. Our key contributions are:

1. We investigate the frequency and difficulty of many web search tasks that developers perform throughout the software development process. Our study includes an observational study of 60 developers, interviews of 12 senior software engineers from two companies, and a survey that is responded by 235 software engineers from 21 countries and coming from various software companies and open source projects.
2. We provide a ranked list of 34 search tasks. Such a ranked list helps researchers and practitioners understand the search tasks that are deemed as difficult/very difficult and which are performed often/very often. We find that participants show differing opinions towards the different search tasks. For some tasks, participants frequently perform them, or they find it difficult to find desired results; while for other tasks, participants rarely perform them, or they find it easy to find the desired results. We find that tasks such as searching for best industrial practice, solutions to configuration, security, and performance bugs, reusable code snippets, and database optimization solutions, are frequently performed and for which finding the desired results is difficult.
3. We highlight the rationale for the frequency/difficulty of certain search tasks. We also discuss the implications of our study on future research, including (1) code search engines, (2) domain-specific search engines, (3) automated generation and

refinement of search queries, (4) automated prediction of the quality of search results, (5) configuration, security, and performance bug fixing, and (6) knowledge sharing.

The remainder of this paper is structured as follows. In Section 2, we describe the methodology of our empirical study. In Section 3, we present the results of our study. In Section 4, we discuss the implications and the threats to validity of our study. In Section 5, we briefly mention related work. In Section 6, we conclude our study .

## 2 Research Methodology

Our study consists of an observational study, open-ended interviews, and a large scale validation survey. In the observational study (described in Section 2.1), we automatically collected search queries that were submitted by 60 developers to popular search engines, and manually analyzed the queries to identify some common search tasks. In the open ended interviews (described in Section 2.2), we interviewed 12 senior developers to get their insights into what developers search for during their development activities. At the end of the observational study and the interviews, we identified a collection of 34 search tasks. Next, we evaluated the frequency and difficulty of these search tasks by surveying a large number of software engineers from various backgrounds by means of an online survey (described in Section 2.3). Each survey respondent spent 10–25 minutes to rate how often they perform each of the 34 search tasks and how difficult it is for them to get the desired results for the search tasks. We also asked respondents to provide the rationale behind their ratings.

### 2.1 Observational Study

**Protocol** In the observational study, after seeking and obtaining explicit permissions from our participants, we installed our data collection tool ACTIVITYSPACE into participants' computer, and collected their behavior data for two weeks. ACTIVITYSPACE, proposed in our earlier work (Bao et al. 2015a), uses Windows Accessibility APIs to record developers' actions. Accessibility APIs are the standard interfaces built in modern desktop operating systems for assistive applications, such as screen readers, to access the low-level information of a user interface. Notice in our study, we collected the search queries from search engines and other online knowledge sites. The search engines include Google,<sup>4</sup> Baidu, and Bing. The other online knowledge sites include Stack Overflow, CSDN,<sup>5</sup> Quora,<sup>6</sup> and ZhiHu.<sup>7</sup> Thus, we did not distinguish the search queries from search engines to other online sites.

**Participant Selection** We sent emails to all of the developers in two IT companies in China, namely Insigma Global Service<sup>8</sup> and Hengtian.<sup>9</sup> Insigma Global Service is an

---

<sup>4</sup>Notice although Google is blocked in China, developers can use Google by using agent service such as Shadowsocks. See <https://shadowsocks.com/> for more details.

<sup>5</sup>CSDN is one of the largest technical blog site in China, see <http://www.csdn.net/> for more details.

<sup>6</sup><https://www.quora.com/>.

<sup>7</sup>Zhihu is one of the largest Q&A site in China, see <http://www.zhihu.com/> for more details.

<sup>8</sup><http://www.insigmaservice.com/>.

<sup>9</sup><http://www.hengtiansoft.com/>.

outsourcing company which has more than 500 employees, and it mainly does outsourcing projects for Chinese vendors (e.g., Alibaba, and Baidu). Hengtian is also an outsourcing company which has more than 2,000 employees, and it mainly does outsourcing projects for US and European corporations (e.g., State Street Bank, and Cisco). In total, 60 developers accepted our invitation. These 60 developers are part of different project teams in these two companies and have different roles. For example, some developers mainly use Java as their main programming language, while some mainly use .NET or C/C++. Some developers focus on development, while others focus on testing. Moreover, these 60 developers have different professional experience. The average number of years that these 60 developers worked in IT companies is  $4.2 \pm 2.5$  years. 35% of the developers are junior developers who worked less than 2 years. Including such junior developers into our study provides us with a wider and more diverse view regarding search tasks since such developers are most likely the major users of search engines.

**Data Analysis** In our observation study, we extracted the search queries that are used during development activities, and grouped these queries into different categories (i.e., search tasks). In total, we collected 12,051 queries. Among them, 8,102 queries were written in English, and 3,949 queries were written by using Chinese. Notice that 4,098 queries (34%) were extracted from online sites such as Stack Overflow, CSDN, Quaro, and ZhiHu. These queries are performed by entering search queries in the search box of each of these sites. From our analysis of these queries, we have the following observations:

- Developers are likely to use search engines to search for code or explanations of exceptions. For example, they enter the name of a thrown exception into Google. Yet they rarely use other sites for such searches. In our observational study, we only find 10 queries which copy code or exception into Stack Overflow or CSDN.
- Even though developers primarily use search engines for such searches, most of the answers are available on Stack Overflow, CSDN, or ZhiHu.
- Developers often use online sites to search for answers for general questions, e.g., they search for queries such as “how to use git” in CSDN, and “Java 1.8 new features” in Stack Overflow.

In this paper, we used a semi-automatic approach by applying Latent Dirichlet Allocation (LDA) – one topic modelling technique (Blei et al. 2003) to help us identify the topics among these queries. LDA automatically clusters terms that appear across the extracted queries into different topics, and each topic may represent a search task that we would like to investigate. In this paper, we set the number of topic as 50, and we mixed English and Chinese queries together when running LDA. For each query, we got its topic distribution array, and each value in the array represents the probability that the query belongs to a corresponding topic. And we classified a query into the topic with the highest probability. Next, the first and second author worked together to identify the search task that corresponds to each query, by analyzing the topics and their corresponding terms and queries. Specially, for each topic, we read the associated terms and queries, and we derived search tasks.

Table 1 shows 50 topics which were related to our search tasks and their corresponding top 5 key terms. The key terms from these topics give very useful hints to help us derive search tasks. For example, we find that developers often use search engines due to exceptions or errors that they encounter (Topic 1, 2, 3, 9, 20); Developers also search for code examples (Topic 5, 7) or libraries (10, 11); Sometimes, developers search for unknown

**Table 1** Top 5 key terms for the 20 relevant topics

Topic ID	Key terms
1	java exception configuration output project
2	java number linux error file
3	wordpress site php error css
4	select sql mysql custom data
5	java time current web transform
6	tutorial code asp development git
7	code java implementation access file
8	time linux set current custom
9	network android background connect error
10	web python library mvc json
11	python data library database according
12	file configuration linux group data
13	wordpress angularj com project page
14	angularj java http tutorial data
15	deep ai cnn neural network
16	currency collateral fund equity bond
17	eclips junit git ant license
18	cloudera manag server json js
19	git data get tutorial use
20	data android exception angular group
21	time content asp group data
22	class btrace com http group
23	tutorial code asp development git
24	url path current custom data
25	nodej post http data phpstorm
26	java android zhejiang group data
27	background css hadoop data database
28	wordpress custom article set open
29	english wordpress page hadoop data
30	file upload zhejiang group custom
31	js access eclips mysql angular
32	linux check site zhejiang current
33	page package asp upload group
34	screen mysql linux number group
35	wordpress article template phpstorm zhejiang
36	php output jquery mysql warn
37	group net xinji current angular
38	hangzhou net city zhejiang git
39	rsv bp utf baidu tn
40	university zhejiang line college city
41	default mysql password hadoop data
42	hadoop cloudera development group data
43	tomcat primefac eclips time configuration

**Table 1** (continued)

Topic ID	Key terms
44	cmd linux file path tar
45	version net check zhejiang group
46	webstorm php code svn git
47	springmvc translation angular js data
48	jdk openjdk java version difference
49	sqlserver set manag screen css
50	angularj time tutorial sqlserver git

terminologies (Topic 15) or industrial practices (Topic 18<sup>10</sup>). At the end of this phase, we end up with 28 intermediate search tasks.<sup>11</sup>

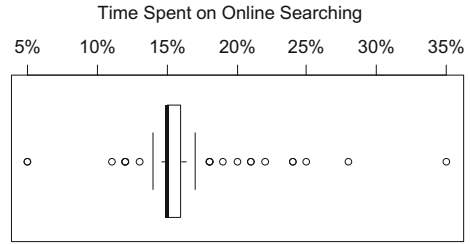
We also recorded the percentage of time that developers spent on online searching. To do so, for each query, we recorded the time when a developer enters a query into a search engine or online sites such as Stack Overflow and CSDN, and the time that he/she switch to another task, or closes the web browser, or opens a new website without clicking the websites that provided by the search engines. The difference of these two times is used as the time spent on online searching. We also recorded the effective working hours of each developer. Effective working hours refer to the time during which developers stay in front of their computer, doing tasks which are related to the project. We excluded the time that developers spend on personal activities (e.g., eating lunch/dinner), or meetings. To do so, we did not count the time if a developer did not have any mouse or keyboard actions for one hour or more. Figure 1 presents the percentage of time that developers spent on online searching. We notice that the median percentage is 15%, with a maximum value at 35%, and a minimum value at 5%. We ask the developer who spent only 5% of his time on search, and he told us that he is a project manager, and during that time, he was extremely busy to design the detailed project plan, thus he spent less time on searching. Moreover, the developer who spent 35% of her time on searching told us that she just graduated from university, and it is the first time that she joined a project team. Thus, she spent a considerable amount of time searching online for various background knowledge topics such as some introductions for techniques that are used in the project.

Figure 2 presents the distribution of the number of queries among the 60 developers. We notice that 16, 32, 10, and 2 developers perform less than 100, 100 to 500, 500 to 1,000, and more than 1,000 queries. Around half of the developers perform among 100 to 500 queries in our observational study. We record the data for two weeks, and suppose a developer would effectively work for 5 hours, then these 32 developers who perform 100–500 queries in the two weeks, they would perform 2 to 10 queries every day. For the two developers who perform more than 1,000 queries, we send emails to ask them for the reason. They told us that they are new employee, and thus they need to get up to speed on many new topics. Online searching could help them find the desired results fast and efficiency.

<sup>10</sup>Cloudera is a software company that provides Apache Hadoop-based software, support and services, and training to business customers (<https://www.cloudera.com/>).

<sup>11</sup>We identified another 6 search tasks in the open-ended interviews.

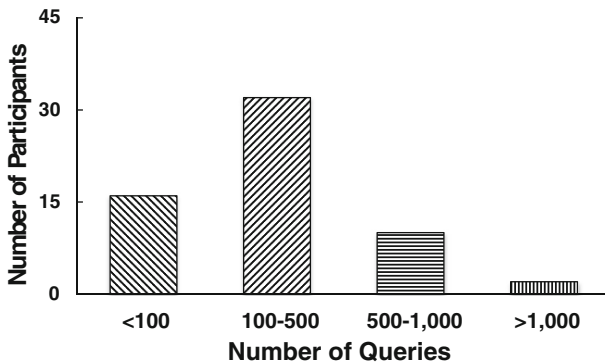
**Fig. 1** Percentage of time that developers spent on online searching



## 2.2 Open-Ended Interviews

**Protocol** The first author conducted face-to-face interviews with 12 senior developers and team leaders. We interviewed these senior developers and team leaders but not junior developers since they had more experience on software development and testing, and they met more problems during the development process and they were likely to search online more frequently. Each interview was around 30 minutes long. The interviews were semi-structured and divided into three parts. In the first part, we asked some demographic questions such as the experience that the interviewee has on software development/testing/project management. In the second part, we asked some open-ended questions such as what do they typically search for online during software development, and which search tasks are difficult (i.e., tasks for which it is hard to find relevant results). Specifically, we asked the following questions:

1. What do you typically search for online using web search engines or on Q&A sites such as Stack Overflow?
2. What kinds of search tasks or queries for which you find it difficult to find the desired results?
3. Suppose you are trying to fix a bug, which search queries would you try first, and what queries do you find it difficult to get the desired results?
4. Suppose you are writing code now, which search queries would you try first, and what queries do you find it difficult to get the desired results?
5. Suppose you are trying to use some third-party libraries or code, which search queries would you try first, and what queries do you find it difficult to get the desired results?



**Fig. 2** Distribution of the number of queries among the 60 studied developers



6. Do you use an IDE, or other tools such as Git and Bugzilla? If so, could you describe which search queries would you try first, and what queries do you find it difficult to get the desired results?
7. Do you use a database often? If so, could you describe which search queries would you try first, and what queries do you find it difficult to get the desired results?
8. Suppose you are performing testing now, e.g., unit testing, which search queries would you try first, and what queries do you find it difficult to get the desired results?

The purpose of this part is to allow the interviewees to speak freely about their search tasks without any bias. Notice these 8 questions focus on different areas, from code writing to testing. Notice it is possible that a participant says no to questions (3) to (8). To address this limitation, in the third part, we discuss with each interviewee the 28 search tasks that we identified in our observational study in an effort to expand the discussion to wide issues that are related to web search activities during the process of software development and maintenance.

**Participant Selection** We conducted interviews with senior developers and team leaders at Insigma Global Service and Hengtian. In total, 12 people were interviewed. Eight interviewees were male, and four were female. In the remainder of the paper, we denoted these 12 interviewees as P1 to P12. The average number of years these 12 interviewees worked in IT companies is 8.3 years and they have diverse experience on different types of projects. For example, P1 worked in different project teams from financial systems to cloud computing systems. P5 lead a large testing team which mainly performs outsourced testing for a commercial bank. P2 worked in an e-commerce project team, and his main responsibility was to maintain the e-commerce system and fix bugs. P8 lead a project team which focuses on language transformation, i.e., automated transformation of COBOL to Java. The diversity of the background and experience of the 12 interviewees helps improve the generalizability of our results.

**Data Analysis** After the interviews, we used a transcription service to transcribe the audio into text. We then read the text to identify additional search tasks that are mentioned by our interviewees. These additional search tasks did not appear in the search queries we collected in the observational study, and the identification process is as follows:

- P1 and P8 mentioned that they sometimes needed to confirm whether it is legal to use or re-use some open-source or commercial projects, or install some open-source or commercial tools, thus we added “search for laws or regulations about a technology” as an additional task.
- P1, P4, P8, P9, P11, and P12 all mentioned that it was hard to describe the details of performance and multi-threading bugs, and also there were not many solutions for performance and multi-threading bugs online. Thus, we added “search for solutions to performance bugs” and “search for solutions to multi-threading bugs” as two additional tasks.
- P5 and P9 who were tasked with standardizing processes across the whole company mentioned they needed to search for various standards such as code, requirement, and design standards. Thus, we added “search for standards” as an additional task.
- Only P5 mentioned that he frequently searched for guidance to avoid anti-patterns. Thus, we added “search for usage examples or guidance on how to avoid an anti-pattern” as an additional task.

**Table 2** List of identified search tasks

## General search

- |    |  |
|----|--|
| T1 | Search for explanation for unknown terminologies                                 |
| T2 | Search for background knowledge related to a project (e.g., financial knowledge) |
| T3 | Search for software developers of interest (e.g., well-known developers)         |
| T4 | Search for laws or regulations about a technology                                |
| T5 | Search for the description of a license  |
| T6 | Search for best industrial practices   |

## Debugging and bug fixing

- |     |  |
|-----|--|
| T7  | Search for explanations for exceptions/error messages (e.g., HTTP 404) |
| T8  | Search for solutions to common programming bugs                        |
| T9  | Search for solutions to software configuration bugs                    |
| T10 | Search for solutions to security bugs                                  |
| T11 | Search for solutions to performance bugs                               |
| T12 | Search for solutions to multi-threading bugs                           |

## Programming

- |     |  |
|-----|--|
| T13 | Search for usage examples or guidance on how to use a new programming language   |
| T14 | Search for usage examples and guidance on how to use a new feature of a programming language (e.g., how to use Lambda expressions in Java-1.8) |
| T15 | Search for standards (e.g., C++ standard)  |
| T16 | Search for usage examples or guidance on how to use a design pattern   |
| T17 | Search for examples or guidance on how to avoid an anti-pattern  |
| T18 | Search for pseudocode, code example, or principle of an algorithm (e.g., Dijkstra algorithm)   |

## Third party code reuse

- |     |  |
|-----|--|
| T19 | Search for reusable code snippets  |
| T20 | Search for suitable third-party libraries/services                                 |
| T21 | Search for usage examples or guidance on how to use third-party libraries/services |
| T22 | Search for configuration script examples of a build system tool                    |
| T23 | Search for HTML/CSS templates for front end development                            |

## Tools

- |     |  |
|-----|--|
| T24 | Search for usage examples and guidance on how to use operating system command line interfaces (e.g., by writing shell scripts) |
| T25 | Search for usage examples and guidance on how to use and customize IDEs (e.g., Eclipse)  |
| T26 | Search for usage examples and guidance on how to use version control systems   |
| T27 | Search for usage examples and guidance on how to use issue tracking systems (e.g., BugZilla)                                   |
| T28 | Search for usage examples and guidance on how to use code review systems (e.g., Gerrit)  |

## Database

- |     |   |
|-----|---|
| T29 | Search for usage examples or guidance on how to form SQL statements   |
| T30 | Search for usage examples or guidance on how to use a no-SQL database |
| T31 | Search for database optimization solutions                            |

## Testing

- |     |  |
|-----|--|
| T32 | Search for guidelines on testing methods (e.g., how to perform smoke testing)                      |
| T33 | Search for usage examples and guidance on how to use an automated testing tool (e.g., JUnit, etc.) |
| T34 | Search for public datasets to test a newly developed algorithm or system                           |

- Only P10, who is an algorithm developer, mentioned that he searched for public datasets to test new algorithms. Thus, we added “search for public datasets to test a newly developed algorithm or system” as an additional task.

We identified a final set of 34 different search tasks that we group into 7 different dimensions, i.e., general search, debugging and bug fixing, programming, third party code reuse, tools, database, and testing. Table 2 presents the list of final search tasks that we identified after the interviews. These search tasks were then feed into the third part of our study.

### 2.3 Validation Survey

**Protocol** We designed a survey to rank the 34 search tasks based on their perceived frequency and difficulty. Our survey has three parts:

1. In the first part of our survey, we asked demographic questions to understand the participants’ background (e.g., their number of years of professional experience).
2. We then present the tasks and ask our respondents to rank the frequency of them performing these tasks using one of the following ratings: *very often*, *often*, *sometimes*, *rare*, and *very rare*. We also ask respondents to rank the difficulty of getting the desired results for these tasks using one of the following ratings: *very difficult*, *difficult*, *neutral*, *easy*, and *very easy*. A participant has the option to specify that he/she prefers not to answer or that he/she does not understand our description of a particular task. We include this option to reduce the possibility of participants providing arbitrary answers.
3. Next, for each respondent, we randomly sampled two search tasks that he/she has ranked as often/very often, and two search tasks that he/she has ranked as difficult/very difficult, and ask the rationale of such ratings.

**Respondent Selection** Our goal is to get a sufficient number of software engineers from diverse backgrounds to rank and comment on the search tasks that we identified. We follow the following strategy to get respondents:

- First, we contacted professionals from various countries and IT companies and asked their help to disseminate our survey to some of their colleagues and friends. We sent emails to our contacts at Microsoft, Baidu, Alibaba, NetEase, Hengtian, IGS, and many other small to large companies in various countries encouraging them to complete the survey and disseminate it. This survey recruitment strategy helps us get respondents who are professional developers from diverse organizations and backgrounds in industry.
- Second, we mined the commit logs of projects that are hosted in GitHub using its REST APIs, and identified highly active practitioners who have contributed more than 1,000 commits. In total, we identified 1,100 email addresses and we sent invitations to these addresses, out of which 140 were not delivered, and 80 emails received automatic replies notifying the receiver’s absence. This recruitment strategy helps us get respondents who are active open source practitioners.

In total, we received 235 responses. These responses were made by respondents from 21 countries across five continents. The top two countries where the respondents reside are China and the United States. The number of years of professional experience of the 235 respondents varies from 0.3 years to 29 years, with an average of 5.89 years.

**Data Analysis** We collated the ratings that our respondents provide. We drop “I don’t understand” and “I prefer not to answer” ratings that form a small minority of all ratings (less than 3%). Next, we converted these ratings to Likert scores from 1 (very rare\very easy) to 5 (very often\very difficult). Next, we computed the average Likert score of each search task. A Likert score is commonly used in surveys when asking respondents to express their agreement or disagreement about a statement (Wuensch 2005). Furthermore, we extracted the comments that our survey respondents provided to explain the reason for a particular search task being performed frequently or it being hard to get the desired results.

### 3 Results

In this section, we describe how software engineers rate the 34 search skills that we grouped into seven dimensions, along with the rationale for performing such tasks frequently or for considering such tasks difficult. In the remainder of the paper, we use  $F$  and  $D$  to denote the comments which explain the rationale for frequently performing a specific search task, and why developers often find it difficult to get desired results for a task, respectively. We carefully read all the comments from the participants, and manually remove the comments which do not describe the rationale for considering a task as frequent or difficult. Next, we group the comments which have similar contents, and present only representative comments from each group.

#### 3.1 Overview

Table 3 presents the average Likert score for the 34 search tasks in terms of frequency and difficulty, and number of queries in our observational study. On average across the 34 tasks, the Likert scores  $3.14 \pm 0.54$  and  $2.73 \pm 0.55$  in terms of frequency and difficulty, respectively. The high standard deviation implies that our participants show differing opinions towards the different search tasks. For some tasks, participants frequently perform them (e.g., search for explanation for unknown terminologies), or participants consider them to be difficult to find the desired results (e.g., search for public datasets to test a newly developed algorithm or system); while for the other tasks, participants rarely perform them (e.g., search for laws or regulations about a technology), or participants consider them to be easy to find the desired results (e.g., search for explanation for unknown terminologies).

To further analyze the results, we apply Scott-Knott Effect Size Difference (ESD) test (Tantithamthavorn et al. 2017) to group the 34 search tasks into statistically distinct ranks according to their Likert scores in terms of frequency and difficulty. Scott-Knott ESD test is a variant of Scott-Knott test (Scott and Knott 1974). Following Tantithamthavorn et al.’s study (Tantithamthavorn et al. 2017), the Scott-Knott test assumed that the data is normally distributed, and it might create groups that are trivially different from one another. To address the limitations of Scott-Knott test, Tantithamthavorn et al. proposed the Scott-Knott Effect Size Difference (ESD) test to correct the non-normal distribution of an input dataset, and merge any two statistically distinct groups that have a negligible effect size into one group (Tantithamthavorn et al. 2017).

Notice that each participant is required to provide the rating on these 34 search tasks in terms of both frequency and difficulty. We input these ratings into the Scott-Knott ESD test. Tables 4 and 5 present the 34 search tasks as ranked according to the Scott-Knott ESD test in terms of frequency and difficulty, respectively. The search tasks in group 1 and 2 have

**Table 3** Median Likert score for the 34 search tasks in terms of frequency and difficulty, and number of queries in our observational study

ID	Task	Frequency	Difficulty	No. Queries
<b>General Search</b>				
T1	Search for explanations for unknown terminologies	4	2	803
T2	Search for background knowledge related to a project (e.g., financial knowledge)	4	2	923
T3	Search for software developers of interest (e.g., well-known developers)	3	2	12
T4	Search for laws or regulations about a technology	2	3	0
T5	Search for the description of a license	3	3	52
T6	Search for best industrial practices	4	4	827
<b>Debugging and Bug Fixing</b>				
T7	Search for explanations for exceptions/error messages (e.g., HTTP 404)	4	2	1,349
T8	Search for solutions to common programming bugs	4	2	1,034
T9	Search for solutions to software configuration bugs	4	4	1,402
T10	Search for solutions to security bugs	3	4	103
T11	Search for solutions to performance bugs	3	4	0
T12	Search for solutions to multi-threading bugs	3	4	0
<b>Programming</b>				
T13	Search for usage examples or guidance on how to use a new programming language	4	2	402
T14	Search for usage examples and guidance on how to use a new feature of a programming language (e.g., how to use Lambda expressions in Java-1.8)	4	2	201
T15	Search for standards (e.g., C++ standard)	3	2	0
T16	Search for usage examples or guidance on how to use a design pattern	3	3	62
T17	Search for examples or guidance on how to avoid an anti-pattern	2	3	0
T18	Search for pseudocode, code example, or principle of an algorithm (e.g., Dijkstra algorithm)	3	3	120
<b>Third Party Code Reuse</b>				
T19	Search for reusable code snippets	3	3	1,302
T20	Search for suitable third-party libraries/services	4	2	802
T21	Search for usage examples or guidance on how to use third-party libraries/services	4	3	423
T22	Search for configuration script examples of a build system tool	3	3	156
T23	Search for HTML/CSS templates for front end development	3	3	45
<b>Tools</b>				
T24	Search for usage examples and guidance on how to use operating system command line interfaces (e.g., by writing shell scripts)	4	2	723

**Table 3** (continued)

ID	Task	Frequency	Difficulty	No. Queries
T25	Search for usage examples and guidance on how to use and customize IDEs (e.g., Eclipse)	3	2	32
T26	Search for usage examples and guidance on how to use version control systems	3	2	28
T27	Search for usage examples and guidance on how to use issue tracking systems (e.g., BugZilla)	2	3	14
T28	Search for usage examples and guidance on how to use code review systems (e.g., Gerrit)	2	3	6
Database				
T29	Search for usage examples or guidance on how to form SQL statements	4	2	1,532
T30	Search for usage examples or guidance on how to use a no-SQL database	3	3	54
T31	Search for database optimization solutions	3	4	238
Testing				
T32	Search for guidelines on testing methods (e.g., how to perform smoke testing)	3	3	62
T33	Search for usage examples and guidance on how to use an automated testing tool (e.g., JUnit, etc.)	3	3	72
T34	Search for public datasets to test a newly developed algorithm or system	2	4	0

higher median Likert scores than the tasks in the other groups. For example, all the search tasks in group 1 in Table 4 have a median Likert score of 4 (often).

From Table 4, the search tasks that developers perform most frequently are (i.e., in groups 1 and 2):

- **T1:** Search for explanations for unknown terminologies
- **T7:** Search for explanations for exceptions/error messages (e.g., HTTP 404)
- **T19:** Search for reusable code snippets
- **T8:** Search for solutions to common programming bugs
- **T20:** Search for suitable third-party libraries/services
- **T13:** Search for usage examples or guidance on how to use a new programming language
- **T6:** Search for best industrial practices
- **T9:** Search for solutions to software configuration bugs
- **T14:** Search for usage examples and guidance on how to use a new feature of a programming language
- **T21:** Search for usage examples or guidance on how to use third-party libraries/services
- **T2:** Search for background knowledge related to a project
- **T24:** Search for usage examples and guidance on how to use operating system command line interfaces

From Table 5, the search tasks that developers consider most difficult are (i.e., in groups 1 and 2):

**Table 4** The 34 search tasks as ranked according to the Scott-Knott Effect Size Difference test in terms of frequency

Group	Task ID	Task Name
1	T1	Search for explanations for unknown terminologies
	T7	Search for explanations for exceptions/error messages (e.g., HTTP 404)
	T8	Search for solutions to common programming bugs
	T19	Search for reusable code snippets
	T20	Search for suitable third-party libraries/services
2	T2	Search for background knowledge related to a project (e.g., financial knowledge)
	T6	Search for best industrial practices
	T9	Search for solutions to software configuration bugs
	T13	Search for usage examples or guidance on how to use a new programming language
	T14	Search for usage examples and guidance on how to use a new feature of a programming language (e.g., how to use Lambda expressions in Java-1.8)
	T21	Search for usage examples or guidance on how to use third-party libraries/services
	T24	Search for usage examples and guidance on how to use operating system commandline interfaces (e.g., by writing shell scripts)
3	T11	Search for solutions to performance bugs
	T29	Search for usage examples or guidance on how to form SQL statements
4	T10	Search for solutions to security bugs
	T15	Search for standards (e.g., C++ standard)
	T18	Search for pseudocode, code example, or principle of an algorithm (e.g., Dijkstraalgorith)
	T26	Search for usage examples and guidance on how to use version control systems
	T31	Search for database optimization solutions
5	T12	Search for solutions to multi-threading bugs
	T16	Search for usage examples or guidance on how to use a design pattern
	T22	Search for configuration script examples of a build system tool
	T25	Search for usage examples and guidance on how to use and customize IDEs (e.g., Eclipse)
	T30	Search for usage examples or guidance on how to use a no-SQL database
	T32	Search for guidelines on testing methods (e.g., how to perform smoke testing)
	T33	Search for usage examples and guidance on how to use an automated testing tool (e.g., JUnit, etc.)
6	T3	Search for software developers of interest (e.g., well-known developers)
	T17	Search for examples or guidance on how to avoid an anti-pattern

**Table 4** (continued)

Group	Task ID	Task Name
	T23	Search for HTML/CSS templates for front end development
	T27	Search for usage examples and guidance on how to use issue tracking systems (e.g., BugZilla)
7	T34	Search for public datasets to test a newly developed algorithm or system

- **T11:** Search for solutions to performance bugs
- **T12:** Search for solutions to multi-threading bugs
- **T34:** Search for public datasets to test a newly developed algorithm or system
- **T19:** Search for reusable code snippets
- **T6:** Search for best industrial practices
- **T31:** Search for database optimization solutions
- **T10:** Search for solutions to security bugs
- **T9:** Search for solutions to software configuration bugs
- **T4:** Search for laws or regulations about a technology
- **T17:** Search for examples or guidance on how to avoid an anti-pattern

### 3.2 General Search

**Explanation for Unknown Terminologies (T1)** From our observational study, we noticed that many queries were related to this task, e.g., search for the definitions of blockchain, cloud computing, PaaS, and SaaS. From our interview, most of the respondents agreed that it is “one of the most common search tasks” (P1), however “current search engines provide good support to it” (P10), and “most of the contents can be found in Wikipedia<sup>12</sup> or Baidu Baike<sup>13</sup> in Chinn (P5 and P6). We did not receive any comments which explain why the task is difficult to perform, and the representative comments that we received which explain the rationale for performing the task frequently are as follows:

- ☞<sup>F</sup> “I have no formal training, many concepts have ambiguous names. There are too many ambiguous acronyms!”
- ☞<sup>F</sup> “Since it is an unknown terminology, I’ve got to know what is all about in order to tackle some tasks or just to be informed.”
- ☞<sup>F</sup> “I have an innate desire to fully understand whatever I touch or use. This includes jargon as well.”

**Background Knowledge Related to a Project (T2)** Background knowledge is “important to a developer to understand the purpose and the scope of a project especially for newcomers” (P3). Examples of this search task include: searching for the definitions of Candlestick chart, pension, and Stock exchange processes. It is “quite common to search for background knowledge when developing financial systems” (P11), and similar to T1, most

<sup>12</sup><https://www.wikipedia.org/>

<sup>13</sup><https://baike.baidu.com/>



**Table 5** The 34 search tasks as ranked according to the Scott-Knott Effect Size Difference test in terms of difficulty

Group	Task ID	Task name
1	T6	Search for best industrial practices
	T9	Search for solutions to software configuration bugs
	T10	Search for solutions to security bugs
	T11	Search for solutions to performance bugs
	T12	Search for solutions to multi-threading bugs
	T19	Search for reusable code snippets
	T31	Search for database optimization solutions
	T34	Search for public datasets to test a newly developed algorithm or system
2	T4	Search for laws or regulations about a technology
	T17	Search for examples or guidance on how to avoid an anti-pattern
3	T28	Search for usage examples and guidance on how to use code review systems (e.g., Gerrit)
4	T5	Search for the description of a license
	T16	Search for usage examples or guidance on how to use a design pattern
	T18	Search for pseudocode, code example, or principle of an algorithm (e.g., Dijkstraalgorithm)
	T21	Search for usage examples or guidance on how to use third-party libraries/services
	T22	Search for configuration script examples of a build system tool
	T23	Search for HTML/CSS templates for front end development
	T27	Search for usage examples and guidance on how to use issue tracking systems (e.g., BugZilla)
	T30	Search for usage examples or guidance on how to use a no-SQL database
5	T33	Search for usage examples and guidance on how to use an automated testing tool(e.g., JUnit, etc.)
	T2	Search for background knowledge related to a project (e.g., financial knowledge)
	T8	Search for solutions to common programming bugs
	T15	Search for standards (e.g., C++ standard)
	T20	Search for suitable third-party libraries/services
	T25	Search for usage examples and guidance on how to use and customize IDEs (e.g., Eclipse)
	T26	Search for usage examples and guidance on how to use version control systems
	6	T3
T13		Search for usage examples or guidance on how to use a new programming language
T14		Search for usage examples and guidance on how to use a new feature of a programming language (e.g., how to use Lambda expressions in Java-1.8)
T29		Search for usage examples or guidance on how to form SQL statements

of the search engines provide good support to it. We did not receive comments on the difficulty of the task, and the representative comments that we received on why some of our respondents perform this task frequently/very frequently are as follows:

- ☞<sup>F</sup> *“I usually want to know as much as I can, to help me in my work and to just know more in general. Also because it is fun.”*
- ☞<sup>F</sup> *“To understand the requirement of a domain-specific project better.”*
- ☞<sup>F</sup> *“Most projects aren’t very transparent; regulations are usually hard to read and/or ambiguously phrased.”*

**Software Developers of Interest (T3)** From our observational study, we only found 12 queries which search for developers of interest, and queries include searching for Richard Stallman, Linus Torvalds, and Wensong Zhang.<sup>14</sup> The median Likert scores of this task in terms of frequency and difficulty are 3 (sometimes) and 2 (easy) respectively. We did not receive any comments on the frequency and difficulty of the task.

**Laws or Regulations About a Technology (T4)** We did not find the queries related to laws or regulations about a technology. However, P1 and P8 who have been part of legal departments in the past pointed out that they performed this task sometimes when a company plans to re-use some open-source or commercial projects, or install some open-source or commercial tools, and they needed to ensure whether it is legal to use them across the company. The representative comments that we received on the difficulty of task T4 are as follows:



- ☞<sup>D</sup> *“I am not a lawyer and laws scare me. I don’t do it that often, but when I do I’m often just frustrated.”*
- ☞<sup>D</sup> *“I would like to understand easier what laws are in place regarding technology. But there is no single source of information.”*
- ☞<sup>D</sup> *“Because there is usually only \*\*\*\*\*<sup>15</sup> webs reporting about it and there is lot of bureaucrat language involved.”*
- ☞<sup>D</sup> *“Laws and regulations are complex and rarely distilled for consumption by non-lawyers, making searches for such item especially difficult.”*

From the above comments, developers consider this task as difficult because: (1) they are not professional lawyers, (2) there is no single information source on technology laws or regulations, and (3) a large amount of “legal Jargon” is used which often impairs understanding.







**Description of a License (T5)** The queries of this task included searching for the description of GNU GPL, LGPL, and BSD. Developers search for the description of a license “when they need to use an open-source project” (P12). However, it takes a great amount of time and effort to find the desired results, “since they have to ensure whether the licenses in the open-source projects cause conflicts to the existing projects or tools used in the company” (P4), and “the licenses may contain ambiguous wording” (P8). The representative comments that we received on the difficulty of this task are as follows:

<sup>14</sup>Wensong Zhang is the co-founder of the project Linux Virtual Server, see [https://en.wikipedia.org/wiki/Linux\\_Virtual\\_Server](https://en.wikipedia.org/wiki/Linux_Virtual_Server) for more details.

<sup>15</sup>An expletive was masked out.

-  *D* “I use CC licenses because they are easy to manage, but software licenses are complex and sometimes a license cannot be used in specific contexts.”
-  *D* “Most of the licenses I searched are incomplete. And the licenses are hard to understand.”





**Best Industrial Practices (T6)** From our observational study, examples of this search task include: “why OceanBase is successful”, “how to develop a good android app”, “how to perform performance testing for a mobile application”. Most of our interviewees mentioned that they frequently search for best industrial practices “when they complete the architecture design of a project” (P3, P4, and P5), or when they compare a proposed solution with state-of-the-art solutions (P1, P7, P8, and P9). However, finding the desired results for this task is really difficult since the returned results are either “incomplete” (P5), or “are not relevant to what they want” (P1). The representative comments that we received on the frequency and difficulty of this task are as follows:


-  *F* “To compare the current solution in our project with the state-of-the-art industrial practice.”
-  *F* “To understand the current industrial situation and development trend.”
-  *F* “Ensure I remain up to date with current methodologies.”
-  *D* “Competing solutions don’t often differ a lot. People who know 2+ solutions are rare, bloggers are rarer.”
-  *D* “It is hard to judge whether a so called “best industrial practice” is the best.”
-  *D* “Most of the best industrial practice are rarely shared, and often kept as the secret inside company.”

From the above comments, developers often search for best industrial practices to (1) compare different solutions, (2) understand industry trends, and (3) ensure that they use state-of-the-art technologies. However, it is often difficult to get the desired results, since (1) there are many incomplete solutions, (2) it is hard to judge which practice is a best practice, and (3) most best practices are not shared.

### 3.3 Debugging and Bug Fixing





**Explanations for Exceptions/Error Messages (T7)** A number of queries (i.e., 1,349 queries) that we collected were related to this task, examples of this search task include: “FileNotFoundException”, “java.util.concurrent. TimeoutException”, “System.Web.HttpException”. Our interviewees pointed out that it is common to search for explanations for exceptions, and “developers would directly copy and paste the exception thrown out by an IDE into a search engine’s query box” (P7), and “it is easy to find the desired results since most of the answers can be found in Stack Overflow or other Q&A sites” (P9). The representative comments that we received on the frequency and difficulty of this task are as follows:

-  *F* “I primarily program in C++, where error messages can sometimes be very difficult to understand.”
-  *F* “When an exception occurs, I want to know if it’s only me, or it’s a common developers problem, and how other developers solve it.”
-  *F* “Trying to debug edge cases, to remember details, to see cause in comments, etc.”
-  *F* “Quick way to diagnose unclear error messages and start working toward a solution.”





-  *D* “Usually, I’m looking for exceptions I can not debug, and as such exceptions are rare (usually I can solve them), there’s a high chance the bug is rare enough, so very few (or none) people stumbled upon it.”

From the above comments, developers often search for explanations for exceptions/error messages because: (1) exception messages sometimes are hard to understand, (2) they want to ensure whether an exception is common, and (3) find solutions to solve it. However, in some cases it is hard to get the desired results if the exception occurs rarely.

**Solutions to Common Programming Bugs (T8)** Similar to T7, there were a number of queries (i.e., 1,034 queries) related to finding solutions to common bugs, examples of this search task include: “c# cannot convert null to int”, “null pointer assignment error”, and “socket connect timeout”. The representative comments that we received on the frequency and difficulty of this task are as follows:






-  *F* “It’ll be much cheaper & faster to Google compared to figure it out by myself.”
-  *F* “When I stumble upon a bug that I cannot debug quick enough, I try to find some more info, so I know if somebody already got a similar problem and how they solved it.”
-  *F* “No point re-inventing the wheel if somebody has already solved the bug.”
-  *D* “Because usually there are no good answers, or not answering the same thing that I was looking for.”

**Solutions to Software Configuration Bugs (T9)** In our observational study, we noticed a number of queries (i.e., 1,402 queries) related to configuration bugs, examples of this search include: “maven configuration error”, “mvn -DmyVariable”, and “ubuntu php mysql AllowOverride”. Novice developers “often meet the configuration errors when deploy a project or install a software” (P6), and it is hard to find the desired results since there are “too many configuration parameters, it is hard to locate which one is problematic, and too many noise results online” (P4). The following are the representative comments that we received on the frequency and difficulty of this task:

-  *F* “Because I face these very often. Software configuration file formats are typically schemaless (e.g., YAML) so they’re very difficult to get correct without borrowing heavily from a known good template.”
-  *F* “Some applications that I integrate are configurable and that creates incompatibilities.”
-  *D* “Configuration bugs are often elusive and they are system-dependent. There are multiple solutions and they don’t always work, depending on the conditions.”
-  *D* “There seems to almost be a stigma about discussing and sharing configurations. I often feel like I am left to my own devices to figure out how to properly configure something when the default settings are not appropriate.”




From the above comments, developers search for solutions to configuration bugs frequently since (1) they often meet configuration problems, (2) the configuration file formats are typically schema-less, and (3) most of applications are configurable which causes incompatibilities problems. However, finding the desired results for this task is difficult because configuration bugs are often system-dependent, with many online solutions not always working.

**Solutions to Security Bugs (T10)** Although there were not that many queries (i.e., 103 queries) related to security bugs in our observational study, most of the queries were hard to resolve through searching since “it is hard to describe a security problem in several words” (P9). Examples of this search task include: “security exception java missing required permissions manifest”, “sql injection prevention”, and “vulnerabilities scanner”. The representative comments that we received on the frequency and difficulty of this task are as follows:


-  *“I am not an expert on solving security bugs, and I have to use Google to search for solutions.”*
-  *“Security bugs are one of the most common bugs I meet during my development process, however, the root cause of these bugs are hard to identify and thus I always search online.”*
-  *“For the things I’ve searched for they’ve been pretty niche.”*
-  *“Security bugs are hard to get right; it is seldom you’ll find a solution among the first several pages of google results that is the best solution for a security issue.”*
-  *“Many security bugs are complex and not well understood by the general programming populace, making finding correct solutions to such bug challenging.”*

From the above comments, developers frequently search for solutions to security bugs since (1) such bugs are common but the root causes are hard to identify, and (2) developers may not have enough expertise to fix them. However, developers argue that a search for a desired solution to a security bug is difficult because most of the solutions online are niche, and the best solution for a security bug is often ranked low in the results of search engines.

**Solutions to Performance Bugs (T11)** We did not find queries related to performance bugs in our observational study, however during our interview, some participants pointed out that although it is not difficult to detect performance bugs (e.g., the machine is suddenly quite slow), “it is really difficult to describe what is the problem, and which makes it difficult to find the desired results” (P1). And “most of the solutions found online are too vague, as they do not provide a detailed solution to a specific problem” (P12). The representative comments that we received on the frequency and difficulty of this task are as follows:

-  *“Performance is super important, and I always suffer from performance bugs, but not sure how to fix them.”*
-  *“It is hard to find general solutions for performance issues, it is usually much more detailed than that. People almost never blog or write tutorials about fixing performance bugs.”*
-  *“Very little is written about front end performance. And when I do find something, it’s buried in a TL;DR; (too long, didn’t read) article.”*

**Solutions to Multi-threading Bugs (T12)** Similar to T11, we did not find queries related to multi-threading bugs in our observational study. However, in our interviews, P8 and P9 who joined a flash memory database project considered searching for solutions to multi-threading bugs as difficult since such bugs were hard to reproduce and hard to describe. The representative comments that we received on the frequency and difficulty of this task are as follows:

-  *“They’re often more subtle than first appears and I want some insight into what to keep a look out for.”*

- ☞<sup>F</sup> *“Because Ruby (my main language) did not provide good support on multi-thread programming, so sometimes we got some weird bugs that aren’t easy to catch because of non thread safe third party libraries.”*
- ☞<sup>D</sup> *“Multi-threading bugs are very specific to the application’s architecture, and they are hard to reproduce. Search engines almost never help you solve them.”*
- ☞<sup>D</sup> *“Concurrency is an inherently difficult topic. One that is difficult to describe and one that is difficult to debug. It’s hard to search for solutions to these bugs mostly just because they are hard to track down and understand. Usually the error message is a red-herring.”*

From the above comments, developers frequently search for solutions to multi-thread bugs due to some programming languages not supporting multi-thread programming well, and multi-thread bugs being subtle. However, developers consider it difficult to get the desired results because multi-thread bugs are specific to the application’s architecture, are hard to reproduce, and/or are hard to track down and understand.

### 3.4 Programming

#### Usage Examples or Guidance on How to use a New Programming Language (T13)

Examples of this search task include: “python tutorial”, “Go Guideline”, and “Scala usage”. From our interviews, all the participants mentioned that they frequently search for the usage of a new programming language since “different projects use different programming languages” (P2), and “they also need to update their knowledge frequently” (P5). Still, it is easy to find considerable material online since “a lot of geeks like to share their experience” (P1). The representative comments that we received on the frequency of this task are as follows:

- ☞<sup>F</sup> *“Seeking best practice to avoid developing bad programming habits. And to master the programming language fast.”*
- ☞<sup>F</sup> *“It’s fun to try new languages but sometimes the syntax gets mixed up.”*
- ☞<sup>F</sup> *“Whenever I want to try a new programming language, I’m interested in how to use it for similar use case I would know in other languages.”*




#### Usage Examples and Guidance on How to use a New Feature of a Programming Language (T14)

Examples of this search task include: “Java 8 Advanced Features”, “Java LAMDA Expression”, and “.Net Remoting”. Our interviewees mentioned that they always perform this search when they are required to “migrate a project in an old version of a programming language to the newest version of the language” (P10), e.g., migrating a project in Visual Studio 2005 to Visual Studio 2013. The representative comments that we received on the frequency of this task are as follows:




- ☞<sup>F</sup> *“Because I mainly dabble in web technologies, the usage examples of new Javascript or CSS features are very well documented and used pretty broadly.”*
- ☞<sup>F</sup> *“Due to the evolution of our system, we have to learn how to use the new features. Searching for these new features online can help better understand them, and analyze the impact of them on our system. ”*

**Standards (T15)** From our observational study, we did not find queries related to standards. But two interviewees (P5 and P9) mentioned that they frequently search for standards since they are tasked with standardizing processes across the whole company, the standardization process covers code to design standards, as well as requirement standards. We did not



receive any comments on the difficulty of this task, and the following are the representative comments that we received on the frequency of the task:

-  *F* “I want to see how the standard defines things. I want to see how my programming language or protocol behaves.”
-  *F* “I prefer to understand a language or technical concepts from first principles. Thus reading the standard is usually the best way to learn a language.”
-  *F* “Standard compliance is very important for compatibility.”

**Usage Examples or Guidance on How to use a Design Pattern (T16)** Examples of this search task include: “Singleton Java code sample”, “simple factory pattern principal”, and “adapter pattern”. Some interviewees mentioned that design pattern are frequently used for legacy projects which require a considerable amount of refactoring effort. From our survey, respondents considered that they sometimes perform this task and finding the desired results for this task is relatively not difficult. The median Likert scores of this task in terms of frequency and difficulty are 3 and 3 respectively. The following are the representative comments that we received on the frequency of the task:

-  *F* “I search for usage example of design patterns so that I can use them well.”
-  *F* “I want to ensure I am doing things the right way, and compare various design patterns to decide which is the best in each situation.”
-  *F* “Looking for alternative thought processes to my own, “Devil’s advocate” for a second point of view on a problem.”



**Usage Examples or Guidance on How to Avoid an Anti-pattern (T17)** We did not find queries related to anti-patterns from our observational study, and only one interviewee (P5) mentioned that he frequently searches for guidance to avoid anti-patterns. The representative comments that we received on the difficulty of this task are the following:





-  *D* “Most of the search results on anti-pattern are hard to understand and useless.”
-  *D* “Since I don’t know much about anti-pattern, even if I find something, I am not sure whether I can apply it in my program.”

**Pseudocode, Code Example, or Principle of an Algorithm (T18)** Examples of this search include: “dynamic programming Java code”, “shortest distance graph”, and “.Net sorting”. We did not receive any comments for this task.

### 3.5 Third Party Code Reuse






**Reusable Code Snippets (T19)** In our observational study, we noticed 1,302 queries that are related to reusable code snippets, Examples of this search task include: “login page template”, “java crawler code”, and “decision tree implementation code in java”. From our survey, some participants expressed that they frequently search for code snippets “since they may forget how to call a sequence of an API to implement a functionality” (P4), and they considered it as a difficult task since “general search engines cannot understand the purpose of the query, and most of the time the participants just want a reference example” (P5). The representative comments that we received on the frequency and difficulty of this task are as follows:

-  *F* “To reduce the development time, especially when I cannot find a library.”
-  *F* “I want to reuse code so that I don’t reinvent the wheel.”

-  *D* “Most search engines ignore the symbols (e.g., ^, !, &, etc) that are crucial in code snippets since they’re not as important in written text, so getting exact matches can be a pain.”
-  *D* “There can be many snippets of code found on the web, but most are not reusable. Many pieces of code are badly written or follow bad practices, are not maintainable, and are not extensible. ”
-  *D* “Because it’s hard to explain in a sentence what the code snippet you’re looking for should do.”
-  *D* “There are too many possibilities for code snippets, it’s easier to just write what I need or find code I’ve already written myself (and can be confident about), than to find someone else’s snippet... And hard to be able to trust it. Also hard to find an exact match; usually I want/need something custom.”

From the above comments, developers often search for reusable code snippets to reduce their development time and effort. However, it is a difficult task because: (1) current search engines such as Google do not provide good support for code searching, (2) many snippets that are found on the web are not reusable and are often of low quality, (3) hard to express what developers want to search in several words, and (4) hard to trust the found online code.


**Suitable Third-Party Libraries/Services (T20)** In our observational study, we noticed 802 queries related to third-party libraries/services, Examples of this search task include: “machine learning library Java”, “algorithm library”, and “opencv”. Different from T19, although most of the respondents agreed that they often search for suitable third-party libraries/services, they can easily get the desired results. The median Likert scores of this task in terms of frequency and difficulty are 4 (often) and 2 (easy) respectively. The following are the representative comments that we received on the frequency and difficulty of this task:

-  *F* “Because using libraries a lot of time to do a big part of the work.”
-  *F* “To see how other people have solved problems, and as help in solving problems I have. Also because it is fun.”
-  *F* “Because it may save me a lot of time and I generally try to fight with NIH (not invented here) syndrome.”
-  *F* “The framework I use the most has a large addon ecosystem and chances that someone already created an addon for your use case are typically high.”
-  *D* “Too many libraries that do the same, hard to measure library quality.”




From the above comments, developers often search for third party libraries/services to save development time, and check how other developers solve similar problems. However, since there are too many libraries with similar functionalities, it is hard to measure the quality of these libraries.

### Usage Examples or Guidance on How to use Third-Party Libraries/Services (T21)

In our observational study, we noticed 423 queries related to guidance on third-party libraries/services. Examples of this search task include: “weka how to write a classifier”, “dom4j analyze XML”, and “log4J new user”. The representative comments that we received on the frequency of this task are the following:



-  *F* “The first thing I’m looking for when I use a new library is how all the pieces fit together so that I can understand the suitability of the third party library for my purpose.”



-  *F* “It can be difficult to find an entry point to third party libraries. Examples make it obvious and give you a place to expand outwards from.”
-  *F* “It’s hard to use a library without documentation. Very often, common libraries have very poor documentation so one needs to search for it. Also, even for project which do have good documentation, it’s often faster to use Google find the right part of the docs to read than using their own index.”
-  *F* “I find that there is frequently a good open-source library to help me build a new feature, so I tend to use them.”





**Configuration Script Examples of a Build System Tool (T22)** In our observational study, we notice 156 queries related to this task. Examples of this search task include: “ant pom.xml example”, “Makefile example”, and “.Net build system script”. We did not receive any comments for this task.

**HTML/CSS Templates for Front End Development (T23)** In our observational study, we noticed 45 queries related to this task. Examples of this search task include: “php forum template”, “css blog style template”, and “wordpress template”. The following are the representative comments that we received on the frequency and difficulty of this task:

-  *F* “Since there are a lot of frameworks for front end development, reuse them can improve the productivity. ”
-  *D* “Too many low quality HTML/CSS templates, which make it difficult to find the useful one.”

### 3.6 Tools




**Usage Examples and Guidance on How to use Operating System Command Line Interfaces (T24)** In our observational study, we noticed 723 queries related to this task. Examples of this search task include: “apt-get options”, “Windows ipconfig”, and “mkdir”. Most of the interviewees mentioned that they frequently search for usage examples of the operating system command line interfaces since “there are too many options for a command, and they cannot remember all of them” (P4 and P9). The following are the representative comments that we received on the frequency and difficulty of this task:

-  *F* “I have little experience with the shell and only use it for git and cmake primarily - for anything else, I have to look up how to accomplish my task.”
-  *F* “A lot of commands are rare enough that they’re hard to memorize, so searching is the usual method of learning/remembering how to use them (man is usually a bad explanation).”
-  *F* “I practically live in my terminal so I’m always finding/learning about new CLI tools that I can use to accomplish a task, hence I search for usage details.”
-  *D* “Because OS stuff is usually not targeted for beginners (except Stack Overflow answers).”


From the above comments, developers often search for the usage of operating system command line interfaces because: (1) they may have little experience on commands, (2) there are many rare commands that are hard to memorize, and (3) they search and learn new command line interface to help them during their development process. The main complaint on the difficulty to get desired results is that some tutorials or advice are not easily understood by beginners.

**Usage Examples and Guidance on How to use and Customize IDEs (T25)** In our observational study, we noticed 32 queries related to this task. Examples of this search task include: “Eclipse how to debugging”, “how to auto-complete code in Eclipse”, and “how to add a third-party JAR in Eclipse”. Our interviewees considered this task as a frequent task “especially for new developers who have never used Eclipse or Visual Studio before” (P5), and finding the results for this task was not difficult “since there are many technical blogs on this topic already” (P4). We did not receive any comments for this task.

**Usage Examples and Guidance on How to use Version Control Systems (T26)** In our observational study, we noticed 28 queries related to this task. Examples of this search task include: “Git log”, “SVN conflicts resolve”, and “git clone”. The representative comments that we received on the frequency of this task are the following:

-  *F* “Well, it’s hard to memorize all options of a version control tool.”
-  *F* “Git. Need I say more? That thing is impossible to use without constantly searching for recipes.”
-  *F* “Git’s command-line interface is hard to drive even when you understand the principles, if you don’t invest in a study of it, which I haven’t.”



**Usage examples and guidance on how to use issue tracking systems (T27)** In our observational study, we noticed 14 queries related to this task. Examples of this search task include: “How to assign severity in Bugzilla”, “Crawler Bugzilla bug report”, and “JIRA guideline”. We receive only one comment in support of the difficulty of this task:

-  *D* “Such info is usually buried inside issue tracker’s help system, which is often not indexed good enough.”

**Usage Examples and Guidance on How to use Code Review Systems (T28)** In our observational study, we noticed six queries related to this task. Examples of this search task include: “Gerrit usage”, “gerrit git pull”, and “approve code review Gerrit”. We did not receive any comments for this task.

### 3.7 Database






**Usage Examples or Guidance on How to Form SQL Statements (T29)** In our observational study, we noticed 923 queries related to this task. Examples of this search task include: “how to write store procedure in mysql”, “sql delete one column”, and “group statement sql”. Ten out of 12 interviewees mentioned that they frequently search for the formation of SQL statements since “it is hard to remember so many statements” (P9), and finding the results for this task is easy since “there are plenty of well-documented blogs or posts online” (P5). The representative comments that we received on the frequency of this task are the following:

-  *F* “SQL isn’t my strong point so I generally need help when I’m trying to formulate advanced queries.”
-  *F* “I use SQL databases in my work and the query syntax is a little tedious and hard to remember, so I end up googling around for it.”

**Usage Examples or Guidance on How to use a no-SQL Database (T30)** In our observational study, we noticed 54 queries related to this task. Examples of this search task

include: “mongoDB install”, “mongoDB select count”, and “mongoDB tutorial”. We did not receive any comments for this task.



**Database Optimization Solutions (T31)** In our observational study, we noticed 238 queries related to this task. Examples of this search task include: “mysqlcheck options”, “index mysql create optimization”, and “optimization strategies sql server”. Five out of the 12 interviewees considered this task as difficult since most of the database optimization solutions found online cannot be directly used to solve the problems that they face. The representative comments that we received on the frequency and difficulty of this task are as follows:

-  *F* “The database used in our project contains TB data, the search speed is too slow, thus we always search for optimization solution to increase the search speed in the database.”
-  *F* “It is important and common to optimize the database if you use distributed database (e.g., HBase) or real-time database.”
-  *D* “Most of database optimization solutions I want to search are often very specific, which make it difficult to get the one I want.”
-  *D* “Often I notice there are a number of noise during my search, a lot of article with the title “database optimization” actually have nothing about database optimization.”
-  *D* “Hard to describe what I want to search in several words by using Google.”

From the above comments, developers often search for database optimization solutions since their databases are too large and they must do the optimization, and some specific type of databases (e.g., distributed database and real-time database) need to be optimized often. However, it is difficult to get the desired results because: (1) the target amount of noise in the search results, (2) hard to formulate the search query about a database optimization solution, and (3) the solution for which they are searching is too specific which in turn leads to no results matching.


### 3.8 Testing



**Guidelines of Testing Methods (T32)** In our observational study, we noticed 62 queries related to this task. Examples of this search task include: “junit tutorial”, “test case generation”, and “smoke test”. The representative comments that we received on the frequency of this task are as follows:

-  *F* “I do not generally engage in this behaviour and so am not familiar with the terminology.”
-  *F* “Because the current test method used in my project are incomplete and inaccuracy.”

### Usage Examples and Guidance on How to use an Automated Testing Tool (T33)






In our observational study, we noticed 72 queries related to this task. Examples of this search task include: “WebDriver script”, “LoadRunner Replay”, and “Jmeter tutorial”. The representative comments that we received on the frequency of this task are as follows:

-  *F* “I have been using Jasmine and Protractor for Angular.js and need to learn some portions of it.”

-  *F* “I’m a test enthusiast. Since I code in many programming languages and my memory keeps deceiving me, I’m always searching for better ways to test my code.”
-  *F* “Best practice in this area are moving quickly; lots of new (free for OS projects) CI services are becoming available.”

From the above comments, developers often search for usage example and guidance on how to use an automated testing tool because: (1) they often use automated testing tool in practice, (2) they always search for better automated testing tools to ensure that their code is bug-free, and (3) they plan to integrate automated testing tools into their continuous integration (CI) process.

**Public Datasets to Test a Newly Developed Algorithm or System (T34)** We did not find a query related to this task in our observational study. However, during our interview, P10, who is an algorithm developer, pointed out that he frequently searches for public datasets to test new algorithms, but it was difficult to find the desired results since people are not likely to share their datasets and some datasets may leak private information.” The representative comments that we received on the difficulty of this task are the following:




-  *D* “Test datasets are often not linked to algorithm implementations.”
-  *D* “Finding good and useful datasets turned out to be cumbersome. I guess this is mostly because companies don’t like to share many information.”
-  *D* “Good public datasets are rare, period. Once you think you found one, it’s 1) incomplete, or 2) outdated/ unmaintained.”
-  *D* “Useful datasets are often very domain specific.”
-  *D* “A newly designed system by definition doesn’t fit an existing data set, so not only do you need to find that data, you need to find it in a way that is manageable to manipulate.”

From the above comments, we notice that developers find it difficult to find public datasets to test their newly developed algorithms or systems because: (1) good public datasets are rare, and most of them are incomplete or outdated, and (2) the datasets for which they are searching are domain specific.

## 4 Discussion

### 4.1 Implications

**Code Search Engines** From our survey, some respondents mentioned that current search engines such as Google do not provide good support for code search. Also, we found that search for reusable code snippets (T19) is both a frequent and difficult search task for developers. The following are some comments that we collected related to code search:

-  “Google obviously doesn’t always handle code well, e.g. underscores etc. It would be useful if it stopped autocorrecting that, or if there was something you can e.g. append to the URL.”
-  “Allow special characters in search queries! When I search for C++, don’t search for C with two spaces after it!”
-  “Allow me to search in all worldwide open source code by writing code expressions. Allow me to search in all code repositories together with one simple textbox. Allow

*me to search for weird symbols and operators treating them literally. Show good code search results prioritizing things that semantically make sense in code – if I search for a method name within a repository, show me the method definition, not the use.”*

- ☛ *“Improve tools for searching code online. All software hosting sites (i.e. Github/Bitbucket/grepcode/etc.) have very poor tools for searching code. Provide a way to jump directly from a stack trace to the matching code (same revision/file/line number) if the code is hosted online Provide tools for “jump to definition” in online code browsers. Basically make online code as easy to browse as within an IDE.”*

From the above comments, a developer-friendly web search engine should have the following functionalities: (1) support web search with software engineering (SE) related symbols and terminologies, (2) allow developers to specify that search results should originate from code repositories and SE related websites, (3) integrate search functionalities into IDEs, and (4) prioritize search results by considering their semantic meanings relative to the particular context in which a developer is working.

Past studies (e.g., Krugle 2014; Koders 2016; Lemos et al. 2007; Bajracharya et al. 2006; Linstead et al. 2009) also try to develop code search engines to help developers to improve their search efficiency. Our findings support these existing research studies, and it would be interesting to integrate these code search engines into some general search engines such as Google.

Notice our findings are also consistency with Sim et al.’s findings (Sim et al. 2011, 2012). Sim et al. found that code search engines worked better in searches for subsystems such as implementation and usage examples, but general search engines, e.g., Google, worked better on searches for blocks such as code lines or blocks (Sim et al. 2011). Notice Sim et al.’s findings are based on a controlled experiments of 36 participants, while our study is based on a large-scale empirical study of developers’ perceptions of their online searching activities.

Moreover, from the comments, we also found that IDEs has shortcoming on information seeking and there is a need to design next-generation IDEs that can help developers to do what they want without the need for constant switching between search engines and the IDE. Past studies (e.g., Ponzanelli et al. 2013; Rahman et al. 2014) also investigate how to integrate search engines or Stack Overflow into IDEs. Our findings support these recent research studies and the need for further advances in this direction.

**Domain-specific Search Engines** From our observational study, interview, and survey, we noticed that most of the developers use general search engines such as Google or Bing to search. However, general search engines fail in many instances to return the desired results, and they are not fulfilling all needs. For example, participants found it difficult to find public datasets to test a newly developed algorithm or system, since general search engine are not able to locate domain-specific datasets (T34). Thus, developing domain-specific search engines might be one solution to help address the limitations of general search engines. Such a domain-specific search engine should understand software engineering terminology and domain concepts, and be able to disambiguate query terms and web site contents based on these terminology and concepts. It should also automatically identify useful software information sites, and prioritize results from these sites while ignoring sites containing content that is irrelevant to software development. Of course, if relevant content does not exist online, no search engines can find it.

**Automated Generation and Refinement of Search Queries** For some search tasks (e.g., search for reusable code snippets (T19), and search for database optimization solutions (T31)), some respondents mentioned that it is often difficult to formulate their search in a few words. Hence, researchers should explore automated ways to interactively elicit search requirements (e.g., by asking some questions and allowing developers to provide simple answers), and/or automatically generate and refine search queries based on the context in which a developer is working (i.e., by monitoring the state of his/her IDE). Some related research tools have been proposed in the literature to reformulate search queries for text retrieval in software engineering (e.g., Haiduc et al. 2013), but more work is needed to build a solution that can effectively help developers with online searching. For example, Haiduc et al. propose Refoqus that can refine a user query based on the top-k (e.g.,  $k = 10$ ) documents that are retrieved by a query (Haiduc et al. 2013). However, it would be possible that all top-k documents are irrelevant to the query, and for such cases, there is a need to investigate other ways to refine user queries.

**Quality Prediction for Search Results** From our survey, a number of respondents mentioned that due to a number of low quality online posts and blogs (e.g. low quality code snippets (T19) and HTML/CSS template (T23)), it costed them much time to find their desired results. One way to cope with low quality online content is to indicate the quality of each search results when they are displayed. Some comments that we collected from our respondents on this point are as follows:

- ☛ *“Some weighing of the quality of search results. E.g. stackoverflow is full of low-quality answers.”*
- ☛ *“Quality analysis for libraries, something like Google’s PageRank but for libraries.”*
- ☛ *“Filtering out results that are known of low quality.”*
- ☛ *“Being able to determine if a specific search result actually contains a solution to a problem. Far too often you search for an error message and the top 5 hits all contain just a question on various forums, but not actual solutions.”*
- ☛ *“Another problem for error messages is that you often end up with lots of spam results, e.g., sites that simply copy questions from forums into blog posts. The forum might not have been indexed by Google, but the spam posts were. Thus, you end up finding the question, but not the answers.”*

**Configuration, Security, and Performance Bug Fixing** In our observational study, we noticed there are a number of queries related to configuration and security bugs (1,402 and 103 queries, respectively). From our survey, most of the respondents also mentioned that they often search for solutions to configuration, security, and performance bugs (T9, T10, and T11). But it is often difficult to get the desired results, for example, configuration bugs are often system-dependent, with many online solutions not always working. Our findings highlight the importance of these three types of bugs. Future research efforts should develop more tools to support the fixing of these three types of bugs, or develop a specific search engine which can index and allow developers to effectively search for solutions to such types of bugs.

**Knowledge Sharing and Community Building** Our survey shows that some developers like to share their programming experience (T13), but some respondents also mentioned that some knowledge is not very easy to find, e.g. good industry practice (T6) and public

dataset (T34). So, Some respondents highlighted the need for more developers to share their knowledge, experience, and know-how online, and more online communities in order to provide high-quality information and support related to various technical topics:

- ☛ *“I think the information available on the internet depends heavily on the community. I found the JavaScript developer community to be very beginner friendly. I’ve recently started to learn a bit Haskell and found the information on the internet to be very hard to understand.”*
- ☛ *“Encourage more people to share their excellent experience & solutions can help to improve the search efficiency.”*

Stack Overflow, which is one of the largest question and answering (Q&A) site focusing on software development, has provided a platform for many developers to share. From our observational study, we find that developers frequently use Stack Overflow to find the desired results, and 63% of the searches ended up with a visit to Stack Overflow. Thus, Stack Overflow is a great start but there exists many challenges ahead of us as a community.

## 4.2 Threats to Validity

**Internal Validity** It is possible that some of our survey respondents do not understand some of the descriptions of the 34 search tasks well enough. To reduce this threat to validity, we provided “I do not understand / I prefer not to answer” option in our survey, and we found that the number of respondents who choose this option to be small (less than 3%). We also translated our survey to Chinese to ensure that respondents from China can understand our survey well. We only have our survey in two languages (English and Chinese) since these two are the languages that are spoken by most people in the world<sup>16</sup>. It is also possible that we drew wrong conclusions about participant’s perceptions from their comments. To minimize this threat, we read the interview transcripts many times, and we also checked the survey results and the corresponding comments several times. Moreover, in our observational study, to reduce personal bias and errors in our manual analysis of the search queries, we applied a semi-automated approach which leverages LDA to automatically extract topics from the queries, then we manually analyzed the topics to derive search tasks.

Another internal threat relates to the tool *ActivitySpace* that we used to collect the search queries from developers, which may bias the generalizability of our 34 search tasks. To address this threat, we installed *ActivitySpace* into 60 developers’ desktop computers, and these 60 developers were of different professional experience, and worked on projects which cover a wide topics. Moreover, we collected and analyzed the queries from various search engines and online sites which are the main sites that developers would refer to when they search online, including Google, Baidu, Bing, Stack Overflow, CSDN, Quora, and Zhihu. In our paper, we did not capture what developers might ask during coding (Sillito et al. 2006), and developers may also use other search engines or online tools to perform search task.

**External Validity** To improve the generalizability of our findings, we performed an observational study with 60 developers tracking their web search behaviors during two weeks, interviewed 12 senior developers from two companies, and surveyed 235 respondents from more than 21 countries across five continents working for various companies (including

<sup>16</sup>[https://en.wikipedia.org/wiki/List\\_of\\_languages\\_by\\_total\\_number\\_of\\_speakers](https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers)

Microsoft, Baidu, Alibaba, NetEase, Hengtian, IGS and many other small to large companies) or contributing to open source projects hosted on GitHub. Still, our findings may not generalize to represent the perception of all software engineers. Moreover, we only considered 34 search tasks which are categorized into seven dimensions.

Another threat is that all the participants in observational study and interviews are from two Chinese companies, and thus probably have Chinese as their first language. This may hinder their search effectiveness, as most Q&A sites related to software are in English, and Google is blocked in China. However, from the 12,051 queries that we collected in the observational study, 8,102 queries were written using English, and 3,949 queries were written using Chinese. It is common for Chinese developers to use English Q&A sites such as Stack Overflow, and Google can be accessed using special settings. Moreover, to reduce this threats due to the participants being all from China in the observational study and interviews, we perform a large-scale survey study that involves a total of 235 developers across 21 different countries. In our survey, 82 developers are from North America (US and Canada), 51 are from Europe, 15 are from Australia and New Zealand, and 87 are from Asia (including 75 developers from China). The diversity of the respondents helps us combat this threat.

## 5 Related Work

There are many studies on code searching which is one specific search task that developers perform. Previous researchers have investigated developer code search practices by conducting surveys (Sim et al. 1998; Stolee et al. 2014). Sim et al. found that the most common motivators behind code search are defect repair, code reuse, program understanding, feature addition, and impact analysis (Sim et al. 1998). Stolee et al. found that 59% of the surveyed developers perform code search daily (Stolee et al. 2014). Search logs are also a good dataset to study the search behavior of developer. By analyzing search logs, Brandt et al. found that 48% of the queries contain just code, 38% contain just natural language, and 14% contain a mix of both (Brandt et al. 2009). Sadowski et al. conducted a study at Google by surveying developers and analyzing search logs to learn how developers search for code (Sadowski et al. 2015). Their study investigated the time when developers form a search, search scope, query properties and the search context. Bajracharya et al. conducted an analysis of the usage log of Koders (2016), a source code search engine, over a period of one year using topic modelling to understand what users of code search engines are looking for (Bajracharya and Lopes 2009, 2012). Treude et al. categorized the kinds of question that are asked on Stack Overflow and explore which questions are answered well and which ones remain unanswered (Treude et al. 2011). In this study, we investigate web search tasks that developers perform throughout the software development process. Developers search for not only code but also many other things, e.g., best industrial practices, and how to use a particular tool. Moreover, we extract and analyze search tasks not only from search engines such as Google, but also from other online sites such as Stack Overflow.

A number of researchers have performed user studies in controlled settings to better understand developers' search behavior when they create a new software program (Brandt et al. 2009), maintained existing software system (Li et al. 2013; Ko et al. 2006), test a piece of code (Lemos et al. 2007), and questions that asked during software evolution tasks (Sillito et al. 2006). For instance, Li et al. observed the development process of 24 developers and then divided the process into several search sessions (Li et al. 2013). They found that developers perform 4.3 and 6.6 web searches on average in each search session of two maintenance tasks respectively. Ko et al. found that developers start maintenance tasks



by searching for relevant code using their IDE (40/48 participants) or Google (8/48 participants) (Ko et al. 2006). Sillito et al. conducted two qualitative studies of programmers performing change tasks to medium-size and large-size programs, and they categorize 44 different kinds of questions that are asked by participants (Sillito et al. 2006).

Although these studies give us a deeper understanding of how developers search during software development, maintenance and testing, there are some shortcomings, such as, small number of participants, no professional developers participating in the study, and user study not being performed in a real working environment. Moreover, these studies have not investigated the frequency and difficulty of the specific 34 search tasks that we consider in our paper along with the rationale on the frequency and difficulty of these tasks.

A number of studies have been conducted to investigate the behaviors of common users of general-purpose web search engines (Jansen et al. 2000; Spink et al. 2002; Silverstein et al. 1999; Lee et al. 2005). Broder classified web queries according to their intents into three classes: *navigational*, *informational* and *transactional* (Broder 2002), while Rose and Levinson concluded three user goals in web search: *navigational*, *informational* and *resource* (Rose and Levinson 2004). Cutrell and Guan used eye tracking technology to explore the effects of changes in the presentation of search results (Cutrell and Guan 2007). Different from the above studies, our study focuses on web search activities that developers perform related to their software engineering tasks.

## 6 Conclusion

In this work, we surveyed 235 practitioners from diverse backgrounds to better understand what developers search for on the Web. We investigated a total of 34 search tasks which are grouped into seven dimensions, and asked the practitioners to rate the frequency and difficulty of these search tasks and to provide the rationale of their ratings. We summarized the practitioner responses and highlighted opportunities for future research to better support developers' online searching activities. Specifically, we highlighted the importance of (1) developing domain-specific search engines, (2) automated generation and refinement of search queries, (3) automated prediction of the quality of search results, (4) configuration, security, and performance bug fixing, and (5) knowledge sharing and community building. Moreover, our findings are consistent with previous studies on code search engines. We hope that our studies would inspire more studies on developing efficient and effective domain-specific search engines and code search engines, to help developer improve their productivity.

**Acknowledgments** The authors thank to all the developers who participated in this study. This research is supported by NSFC Program (No.61602403) and National Key Technology R&D Program of the Ministry of Science and Technology of China under grant 2015BAH17F01.

## References

- Krugle (2014) <http://opensearch.krugle.org/projects/>
- Koders (2016) <http://www.koders.com>
- Bajracharya S, Ngo T, Linstead E, Dou Y, Rigor P, Baldi P, Lopes C (2006) Sourcerer: a search engine for open source code supporting structure-based search. In: Proceedings of the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications, ACM, pp 681–682

- Bajracharya SK, Lopes CV (2009) Mining search topics from a code search engine usage log. In: Proceedings of the 6th international working conference on mining software repositories (MSR), IEEE
- Bajracharya SK, Lopes CV (2012) Analyzing and mining a code search engine usage log. *Empir Softw Eng* 17(4-5):424–466
- Bao L, Xing Z, Wang X, Zhou B (2015a) Tracking and analyzing cross-cutting activities in developers' daily work. In: Proceedings of the 30th IEEE/ACM international conference on automated software engineering (ASE), pp 277–282
- Bao L, Ye D, Xing Z, Xia X, Wang X (2015b) Activityspace: a remembrance framework to support interapplication information needs. In: Proceedings of the 30th IEEE/ACM international conference on automated software engineering (ASE), IEEE, pp 864–869
- Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. *J Mach Learn Res* 3:993–1022
- Brandt J, Guo PJ, Lewenstein J, Dontcheva M, Klemmer SR (2009) Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 1589–1598
- Broder A (2002) A taxonomy of web search. In: ACM SIGIR Forum, ACM, vol 36, pp 3–10
- Cutrell E, Guan Z (2007) What are you looking for?: an eye-tracking study of information usage in web search. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 407–416
- Haiduc S, Bavota G, Marcus A, Oliveto R, Lucia AD, Menzies T (2013) Automatic query reformulations for text retrieval in software engineering. In: Proceedings of the 35th international conference on software engineering (ICSE), pp 842–851
- Jansen BJ, Spink A, Saracevic T (2000) Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf Process Manag* 36(2):207–227
- Ko AJ, Myers BA, Coblenz MJ, Aung HH (2006) An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Trans Softw Eng (TSE)* 32(12):971–987
- Lee U, Liu Z, Cho J (2005) Automatic identification of user goals in web search. In: Proceedings of the 14th international conference on world wide web (WWW), ACM, pp 391–400
- Lemos OAL, Bajracharya SK, Ossher J, Morla RS, Masiero PC, Baldi P, Lopes CV (2007) Codegenie: using test-cases to search and reuse source code. In: Proceedings of the 22nd IEEE/ACM international conference on automated software engineering (ASE), ACM, pp 525–526
- Li H, Xing Z, Peng X, Zhao W (2013) What help do developers seek, when and how? In: Proceedings of the 20th working conference on reverse engineering (WCRE), IEEE, pp 142–151
- Linstead E, Bajracharya S, Ngo T, Rigor P, Lopes C, Baldi P (2009) Sourcerer: mining and searching internet-scale software repositories. *Data Min Knowl Disc* 18(2):300–336
- Ponzanelli L, Bacchelli A, Lanza M (2013) Seahawk: Stack overflow in the ide. In: Proceedings of the 2013 international conference on software engineering, IEEE Press, pp 1295–1298
- Rahman MM, Yeasmin S, Roy CK (2014) Towards a context-aware ide-based meta search engine for recommendation about programming errors and exceptions. In: Software evolution week-IEEE conference on software maintenance, reengineering and reverse engineering (CSMR-WCRE), 2014, IEEE, pp 194–203
- Rose DE, Levinson D (2004) Understanding user goals in web search. In: Proceedings of the 13th international conference on world wide web (WWW), ACM, pp 13–19
- Sadowski C, Stolee KT, Elbaum S (2015) How developers search for code: a case study. In: Proceedings of the 10th joint meeting on foundations of software engineering (FSE), ACM, pp 191–201
- Scott AJ, Knott M (1974) A cluster analysis method for grouping means in the analysis of variance. *Biometrics* 30(3):507–512
- Sillito J, Murphy GC, De Volder K (2006) Questions programmers ask during software evolution tasks. In: Proceedings of the 14th ACM SIGSOFT international symposium on foundations of software engineering, ACM, pp 23–34
- Silverstein C, Marais H, Henzinger M, Moricz M (1999) Analysis of a very large web search engine query log. In: ACM SIGIR Forum, ACM, vol 33, pp 6–12
- Sim SE, Clarke CL, Holt RC (1998) Archetypal source code searches: a survey of software developers and maintainers. In: Proceedings of the 6th international workshop on program comprehension (IWPC), IEEE, pp 180–187
- Sim SE, Umarji M, Ratanotayanon S, Lopes CV (2011) How well do search engines support code retrieval on the web? *ACM Trans Softw Eng Methodol (TOSEM)* 21(1):4
- Sim SE, Philip K, Umarji M, Agarwala M, Gallardo-Valencia R, Lopes CV, Ratanotayanon S (2012) Software reuse through methodical component reuse and amethodical snippet remixing. In: Proceedings of the ACM 2012 conference on computer supported cooperative work, ACM, pp 1361–1370

- Spink A, Jansen BJ, Wolfram D, Saracevic T (2002) From e-sex to e-commerce: Web search changes. *Computer* 35(3):107–109
- Stolee KT, Elbaum S, Dobos D (2014) Solving the search for source code. *ACM Trans Softw Eng Methodol (TOSEM)* 23(3):26
- Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2017) An empirical comparison of model validation techniques for defect prediction model. *IEEE Trans Softw Eng (TSE)* 43(1):1–18
- Treude C, Barzilay O, Storey MA (2011) How do programmers ask and answer questions on the web?: Nier track. In: Proceedings of the 33rd international conference on software engineering (ICSE), IEEE, pp 804–807
- Wuensch KL (2005) What is a likert scale? and how do you pronounce 'likert?'. East Carolina University



**Xin Xia** received his PhD degree in computer science from the College of Computer Science and Technology, Zhejiang University, China in 2014. He is currently a post-doc research fellow in the software practices lab at the University of British Columbia, Canada. His research interests include software analytic, empirical study, and mining software repository.



**Lingfeng Bao** is currently a Postdoc in the College of Computer Science and Technology, Zhejiang University. He received his B.E. and PhD degrees from the College of Software Engineering, Zhejiang University in 2010 and 2016. His research interests include software analytics, behavioral research methods, data mining techniques, and human computer interaction.



**David Lo** received his PhD degree from the School of Computing, National University of Singapore in 2008. He is currently an Associate Professor in the School of Information Systems, Singapore Management University. He has close to 10 years of experience in software engineering and data mining research and has more than 200 publications in these areas. He received the Lee Foundation Fellow for Research Excellence from the Singapore Management University in 2009, and a number of international research awards including several ACM distinguished paper awards for his work on software analytics. He has served as general and program co-chair of several prestigious international conferences (e.g., IEEE/ACM International Conference on Automated Software Engineering), and editorial board member of a number of high-quality journals (e.g., Empirical Software Engineering).



**Pavneet Singh Kochhar** is a PhD candidate in School of Information Systems at Singapore Management University. He has previously done summer internship at Microsoft Research and an exchange programme at Carnegie Mellon University. His research interests involve empirical software engineering, mining software repositories, software testing and bug localization. His work has been published in many international conferences and journals such as ASE, ISSTA, EMSE, SANER, ICST, MSR and QSIC. He has also served as an external reviewer for many conferences and journals.



**Ahmed E. Hassan** is the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. He received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. He also serves on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, and PeerJ Computer Science. Contact [ahmed@cs.queensu.ca](mailto:ahmed@cs.queensu.ca). More information at: <http://sail.cs.queensu.ca/>



**Zhenchang Xing** is the senior lecturer at the research school of computer science, Australian National University, Australia. Dr. Xing's research interests include software engineering and human-computer interaction. His work combines software analytics, behavioral research methods, data mining techniques, and interaction design to understand how developers work, and then build recommendation or exploratory search systems for the timely or serendipitous discovery of the needed information.