TYPE Perspective
PUBLISHED 17 August 2022
DOI 10.3389/feduc.2022.908712

OPEN ACCESS

EDITED BY
Xiaofeng Wang,
Free University of Bozen-Bolzano, Italy

REVIEWED BY
Erika Osztián,
Sapientia Hungarian University
of Transylvania, Romania

*CORRESPONDENCE
B. Katalin Szabó
szabo.katalin@ek-cer.hu

SPECIALTY SECTION
This article was submitted to
Digital Learning Innovations,
a section of the journal
Frontiers in Education

RECEIVED 30 March 2022
ACCEPTED 11 July 2022
PUBLISHED 17 August 2022

CITATION
Szabó BK (2022) Web search
of software developers—
Characteristics and tips.
Front. Educ. 7:908712.
doi: 10.3389/feduc.2022.908712

Web search of software developers—Characteristics and tips

B. Katalin Szabó[1,2]*

[1]Reactor Monitoring and Simulator Laboratory, Centre for Energy Research, Budapest, Hungary,
[2]Doctoral School of Informatics, University of Debrecen, Debrecen, Hungary

There is more and more software in the world and this software has to be developed. All the people who develop software can be regarded as software developers, not just the professionals. Naturally, they often perform web searches to support their development activity. The article, based on the pertinent literature and also on the author's own experiences as a longtime software developer, discusses characteristics of software developers' web searches and gives some recommendations and tips to increase the efficiency of their searches, especially complex, exploratory searches. To the author's knowledge, no such summary combined with tips, aimed at software developers, has been published before. It has been written in the hope that software developers, such students and their teachers would find it useful.

KEYWORDS

web search, online search, exploratory search, software developers, software development, tips, recommendations

Introduction

Software developers (people who develop software) often perform web searches in their development activity. The efficiency and the outcome of these searches—such as finding a suitable software library or the right documentation or good example code—may fundamentally affect the success of their development work.

There exist several useful resources on how to search on the web in general: books (Dornfest et al., 2006; Russell-Rose and Tate, 2013; Russell, 2019), articles (Blakeman, 2013), and online resources.[1]

However, as the literature reveals, software developers' web searches have some specificities. Being aware of how their peers search, and knowing some tips can make developers perform their own web searches more consciously and efficiently.

(By searches of software developers I mean their searches performed in the capacity of software developers.)

1 http://www.rba.co.uk/search,
http://searchresearch1.blogspot.com/,
https://docs.google.com/document/d/1ydVaJJeL1EYbWtlfj9TPfBTE5IBADkQfZrQaBZxqXGs,
https://sites.google.com/site/resourcesandsearchstrategies/,
https://www.google.com/insidesearch/searcheducation/index.html

Characteristics of software developers' web searches

This section summarizes the most important findings in the literature.

Xia et al. (2017) studied professional developers' web searches, with various methods (analyzing search logs, interviewing developers, performing surveys). They identified the most common search tasks as searching for explanations for unknown terminologies, explanations for exceptions/error messages, reusable code snippets, solutions to common programming bugs, and suitable third-party libraries/services. According to the developers, the most difficult search tasks were 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.

Bansal et al. (2019) analyzed the search log of a major general purpose search engine and identified software engineering related queries with the aid of a machine learning based classifier. 2.61% of all the web search sessions were software engineering related. (While the term "software engineering related queries" perhaps does not absolutely correspond to "developer queries," we can disregard the difference for our purposes.) Software engineering search sessions were shorter than other sessions, had a higher word count, and had a higher rate of term additions and removals in query reformulations. Software engineering search queries had a lower click rate (i.e., the rate of clicking to a web page after the query) than other search queries. (The authors opine that this can be attributed to software engineering related search tasks being more difficult than other search tasks. I must remark, however, that we do not know the proportion of cases when the result preview already gave the answer and there was no need for clicking.) Dwell time, the amount of time spent by users on the clicked documents (calculated as the time between the click and the next seen click or query on the search engine) was lower than with other queries. [The authors, on the basis of search literature, state that longer dwell time correlates with success in finding the required information and they come to the conclusion that therefore software engineering queries are less effective than other queries. I think that this may be a hasty conclusion, as people issuing software engineering queries may be more effective and thus faster in evaluating search results than the average population, and software engineering related web pages may be more to the point, more easily digestible etc. than other web pages. Also, the methodological issue arises that the calculation of dwell time may be questionable if software engineering people tend more to click multiple candidate results in browser tabs in rapid succession and evaluate them one-by-one, such behavior has been described (e.g., in Brandt et al., 2009, p. 3).]

The most researched search type among developers' web searches is search for code, i.e., software. They look for ready-to-use software, linkable/includable code libraries, code examples, code samples, coding tips, bug fixing tips, also algorithms from which code can be written.

The most relevant findings on code search:

Developers prefer general-purpose search engines over any other information gathering means, specialized code search engines are not used too often (Kakarontzas et al., 2010; Sim et al., 2011; Hucka and Graham, 2018). Sim et al. (2013) found that subjects using Google (vs. other search engines) in a code search experiment employed a higher average number of terms per query, a higher average clickthrough rate, and more time overall, and received more hits which were perceived relevant. The researchers opined that the success of Google was probably due to the high number and variety of pages that it indexed, and also because participants in the experiment were more familiar with Google than with the other engines.

A study analyzing the log of a code search engine has come to the conclusion that "users who find code search engines usable are those who already know to a high level of specificity what to look for" (Bajracharya and Lopes, 2009, p. 1).

Sometimes code searches are also performed in code repositories (Kakarontzas et al., 2010; Sim et al., 2011, 2013).

As far as source code search is concerned, Sim et al. (2011) state that, by motivation of such searches, there are two major search archetypes. At one end of the spectrum is the search for source code for reuse as-is, when the developer does not want to modify the code at all. The other archetype is search for code as a reference example, when the developer only wants to use the knowledge behind the code. Between these archetypes are the searches which are some mixes of the two. A search process may even start at one end of the spectrum and may move toward the other end. A minor archetype has also been identified: search for information about bugs or defects, which is characterized by long queries (Rahman et al., 2018, p. 8), as full error messages are often submitted as queries.

(Bai et al., 2020) is about a lab study where graduate students searched for code to resolve programming tasks in an unfamiliar programming language. Emphasis was put on investigating search success. Taking only clicking to a search result into account as proof of search success is inadequate, as inspecting a result is not a guarantee for success yet, and also the correct answer to a search query may appear in the preview in the hit list, not necessitating clicking. Therefore, participants were surveyed during their search activity (when a search tab was closed and when there were signs that a query was reformulated) about their success. A few findings:

Participants most frequently searched for example code and ways to resolve bugs or errors. Learners formulated much more verbose queries than professionals did. Most participants borrowed terms from programming languages which were familiar to them, these queries were more successful than the average query. Successful searches used natural language phrases. [As to the possible reason: Gallardo-Valencia and Sim (2009) opined earlier that web pages with code snippets, examples, usually contain a lot of explanations in natural language, and the general purpose search engines pick up these.] The majority of "How to" queries were successful, and "(. . .) the most successful searches are those that consult resources with examples, such as documentation and official/third-party tutorials." (Bai et al., 2020, p. 2,5,7,10).

Li et al. (2022) conducted an experiment with subjects having to correct buggy Python codes with the aid of searching Stack Overflow through Google. Subjects were categorized as Novices, Non-Python-familiar-experts and Python-familiar-experts. Experts tended to click more search results than novices and had higher success rate. Python experts (the most successful of the three categories) clicked on more links that were further down the result list.

Rahman et al. (2018) compared code search and other search activities of the same developers. In comparison to other searches, code searches contained more words, the number of queries per search session was much higher, query modifications occurred more often, more websites were visited in a session, more time was spent on the search. This implies that searching for code is harder, and also that developers do not give it up easily.

Exploratory searches

Martie (2017), writing about code search, emphasizes the iterative nature of search, especially when the search is not for one specific piece of code, the developer is not entirely sure at the beginning what he/she is searching for.

"In such cases, search is more of an exploratory process where multiple queries are issued by the programmer so that they can discover results to learn and gain ideas from or to learn more about what code they might want in the search engine." (Martie, 2017, p. 61).

Exploratory searches may take a long time, even months (White and Roth, 2009, p. 6) and they are hard to study. In my own practice as a developer, the longest and toughest search for code took months indeed. I wanted a detailed, realistic virtual hand model in a particular game engine, driven with input data from a hand movement detector, for a virtual reality project. After a while it became clear that no ready-made solution existed, I had to create the hand model myself. Luckily, I found a hand mesh (geometric model of a hand) which looked suitable, so I "only" had to integrate it with an armature (a skeleton representation of the bones of the hand in the game engine) and drive the armature's bones with bone position and orientation data from the detector. Apparently, no one had done this before for that particular game engine, so I had to learn a lot about the extremely complicated armature mechanism of the game engine. Certain features worked correctly only under special and undocumented circumstances. My search process was heavily intertwined with development efforts. After each failure I searched further and tried out the new findings. I maintained a search and development diary. The subsequent mining of this diary has revealed that I recorded 853 distinct URLs worthy of noting and these belonged to 174 different websites. 26% of the recorded hits contained code (mostly code snippets) which I thought worthy of copying or downloading. At the time of writing the diary I had no intention to make any kind of statistics from it, so the data presented here contain no such bias. **Table 1** shows how the top 4 websites were represented among the URLs. *blenderartists.org* is an independent user site (with forums) dedicated to the *Blender* (RRID:SCR_008606) suite which includes the game engine, *blender.org* is the official site of *Blender*, *stackexchange.com* is a network of Q&A (question-and-answer) sites on diverse topics, *leapmotion.com* is the official site of the hand movement detector. The two official sites both contain forums as well. 8 further websites held 1–2% of the URLs each. All other sites were below 1%. 119 sites were represented with only one URL. These results show how important the forums have become, and how diverse the URLs can be.

One day, after I applied a trick found in the blog of a small company, the hand model started working correctly. It was published in Szabó (2019).[2]

I mostly used Google for search (Bing and Yandex did not yield additional results), set the number of hits to be displayed

2 See demo at https://youtu.be/uEhEZ1MIul4

TABLE 1 How the top 4 websites were represented among the recorded URLs.

Website	Among all URLs (847 total)	Among important URLs (134 total)	Among code-containing URLs (218 total)
Blenderartists.org	30%	33%	33%
Blender.org	16%	12%	5%
Stackexchange.com	9%	15%	14%
Leapmotion.com	4%	6%	6%
All 4 sites together	59%	66%	58%

on one page to the maximum (to speed up evaluation). When I thought necessary, I scanned several hundred hits in the hit list. When there were several promising hits, I opened them in new tabs one-by-one, then evaluated them one-by-one, relatively quickly.

I often found useful tips in forums and blogs, and it sometimes paid off searching for other posts of authors whose posts I found informative.

My search diary was in Word format, I copied into it relevant URLs, explanatory texts, interesting pieces of code which I sometimes annotated, figures, and even screenshots of my program runs as well. I put exclamation marks before the recorded address of web pages I thought important, I also used font properties (size, color, style) to mark different types of information. When a part of the text (and the idea behind it) proved to be a dead-end street, I changed its font to a smaller one and crossed it out, thus invalidated it but still left it in the diary.

I believe that documenting the searches in the way described above highly contributed to the successful outcome of the search. Also, that I was persistent, did not necessarily stop at 10, 20 or even 100 hits, and assessed the promising hits fast.

Why could the examination of many hits prove useful, especially with exploratory searches? According to White and Roth (2009, p. 15):

"Exploratory searches may be more concerned with recall (maximizing the number of possibly relevant objects that are retrieved) than precision (minimizing the number of possibly irrelevant objects that are retrieved). Thus, they are not well supported by today's Web search engines that are highly tuned toward precision in the first page of results."

Pariser (2011, p. 59) has put this in a simpler way:

"Google is great at helping us find what we know we want, but not at finding what we don't know we want."

However, it can happen that the lack of relevant results among the first hits simply means that the query is misguided, it is better to abandon or at least modify the query. This is always up to the searcher's judgment (which usually develops with practice).

Also, we should not forget that Google does not display more than 1,000 results for any query (Hummel, 2008, p. 75; Russell-Rose and Tate, 2013, p. 151).

To help users in their searches (and to profit from offering products to users), search engines often create profiles of users. For this, they use their search history, geographical location and other data such as what commercial products they bought in the past, what information they have given about themselves. Thus search engines are able to provide search results more specific and relevant to the user, but, with this, they put the user into a sort of "bubble." This feature (and also that the hits frequently clicked by other users will probably be ranked higher, with which they will get even more clicks) can be counterproductive in exploratory search, when the solution often does not lie along the well-trodden paths.

One, time-consuming, strategy to counteract this effect is the above mentioned examination of a high number of search hits. We can also try countermeasures to depersonalize, to "debubble" the search.[3]

A few more web search tips for developers

If you want to use search engines efficiently, you should know their advanced search options and characteristics. Not only Boolean operators can be put to use. For software component search, the *filetype* operator (Google, Bing, Yandex) can be of help (it recognizes the file types of several programming languages) and also Google's and Bing's wildcard (*) (e.g., when looking for functions/methods with known parameter types, wildcards can serve as substitutes for the unknown parameter names) (Hummel, 2008, p. 76). With the *verbatim* option Google will not look for synonyms and variations of the search terms, this can be useful e.g., when looking for specific function/method names in source code search.

You can learn a lot about the capabilities and characteristics of search engines from search experts.[4]

It can pay off to use search engines other than Google, you may get different results.[5]

A potentially useful type of information source is scholarly literature, both in the application domain for which you develop your software, and in the software engineering field. Scholarly articles, books, studies, research reports, dissertations may contain code snippets, algorithms, coding tips, references to ready-to-use programs and to components. Sometimes authors make their own software available to the public. Many scholarly sources can be found through general purpose search engines, but search engines specializing in scholarly literature can also be of great help. Of these scholarly search engines, Google Scholar can be recommended in the first place, because it has the widest coverage of sources. There are also topical search sites for various domains, and the sites of publishers of scholarly journals and books are also valuable resources. Publications usually contain references, some of these may be worth to inspect, also, a publication of interest may be cited in other sources which could in turn contain further useful information for us. Such citation searching is supported e.g., by Google Scholar.

3 Depersonalization tips for Google: https://nostop.net/depersonalize-google-search.html

4 See the Introduction for such resources.

5 For alternative search engines, see e.g., https://www.searchenginejournal.com/alternative-search-engines/271409/

Social media, such as Q&A sites (notably, Stack Overflow), forums, blogs, microblogs (Twitter) collaboratory code repositories (SourceForge, GitHub etc.) can also be great sources of information. In code search, these can also help with the evaluation of code candidates, as they often show the community's opinion of the code.

Discussion

On the basis of the available literature, I have listed some features of developers' web searches, and a few tips for making searches more efficient, derived from the literature and my own experiences.

Where are we headed? This greatly depends on the evolution of search engines. The current practice is to use mostly general-purpose search engines even for code search. The formerly popular Google Code Search was shut down years ago[6] and several other code search engines have become obsolete, too.[7] Perhaps the ever-increasing quantity of available code has become too much for them. It is in principle possible that a search tool specializing in code search emerges and becomes dominant, but it has not happened so far.

Possibly, general-purpose search engines might evolve into more efficient tools through addition of software-developer-friendly features. However, as mentioned above, only less than 3% of web search sessions are software engineering related, so this is probably not high on the priority list of search engine companies. Until then (and even after, in the case of exploratory searches), we must resort to search tricks to make our search efficient.

[6] http://googleblog.blogspot.com/2011/10/fall-sweep.html

[7] In 2018, Code Sample Answer (https://blogs.bing.com/search-quality-insights/2018-07/Intelligent-search-Coding-answers-at-your-fingertips/) was introduced in Bing, but now (2022) the code samples are no longer available in the search results.

Data availability statement

The data analyzed in this study is subject to the following licenses/restrictions: Some slight statistical data (number and proportion of recorded URLs) were derived from the search and development diary of the author. (If such a diary can be regarded as a dataset.) The diary is of several hundred pages and contains know-how which is irrelevant from the point of view of this article and is the property of the author's employer. Requests to access these datasets should be directed to corresponding author.

Author contributions

The author confirms being the sole contributor of this work and has approved it for publication.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Bai, G. R., Kayani, J., and Stolee, K. T. (2020). "How graduate computing students search when using an unfamiliar programming language," in *Proceedings of the IEEE/ACM International Conference on Program Comprehension (ICPC'20)*, (New York, NY: ACM), 160–171. doi: 10.1145/3387904.3389274

Bajracharya, S., and Lopes, C. (2009). "Mining search topics from a code search engine usage log," in *Working Conference on Mining Software Repositories*, (Vancouver, BC: IEEE), 111–120. doi: 10.1109/MSR.2009.5069489

Bansal, C., Zimmermann, T., Awadallah, A. H., and Nagappan, N. (2019). The Usage of Web Search for Software Engineering. *arXiv 1912.09519* [Preprint].

Blakeman, K. (2013). Finding research information on the web: how to make the most of Google and other free search tools. *Sci. Prog.* 96, 61–84. doi: 10.3184/003685013X13617253047438

Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., and Klemmer, S. R. (2009). "Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, (New York, NY: ACM), 1589–1598. doi: 10.1145/1518701.1518944

Dornfest, R., Bausch, P., and Calishain, T. (2006). *Google Hacks*, 3rd Edn. Sebastopol, CA: O'Reilly Media, Inc.

Gallardo-Valencia, R. E., and Sim, S. E. (2009). "Internet-scale code search," in *Proceedings of the 2009 ICSE Workshop on Search-Driven Development – Users, Infrastructure, Tools and Evaluation*, (Washington, DC: IEEE Computer Society), 49–52. doi: 10.1109/SUITE.2009.5070022

Hucka, M., and Graham, M. (2018). Software search is not a science, even among scientists: a survey of how scientists and engineers find software. *J. Syst. Softw.* 141, 171–191. doi: 10.1016/j.jss.2018.03.047

Hummel, O. (2008). *Semantic Component Retrieval in Software Engineering.* Ph.D. thesis. Mannheim: Universität Mannheim.

Kakarontzas, G., Katsaros, P., and Stamelos, I. (2010). Component Certification as a Prerequisite for Widespread OSS Reuse. *Electron. Commun. EASST* 33, 1–20. doi: 10.14279/tuj.eceasst.33.449

Li, A., Endres, M., and Weimer, W. (2022). "Debugging with stack overflow: web search behavior in novice and expert programmers," in *Proceedings of the International Conference on Software Engineering – Software Engineering Education and Training (ICSE-SEET)*, 69–81. doi: 10.1109/ICSE-SEET55299.2022.9794240

Martie, L. (2017). *Understanding the Impact of Support for Iteration on Code Search*. Ph.D. thesis. Irvine: University of California.

Pariser, E. (2011). *The Filter Bubble: What the Internet is Hiding from You*. London, UK: Penguin.

Rahman, M. M., Barson, J., Paul, S., Kayani, J., Lois, F. A., Quezada, S. F., et al. (2018). "Evaluating how developers use general-purpose web-search for code retrieval," in *Proceedings of the 15th International Conference on Mining Software Repositories (MSR)*, (New York, NY: ACM), 465–475. doi: 10.1145/3196398.3196425

Russell, D. M. (2019). *The Joy of Search: A Google Insider's Guide to Going Beyond the Basics*. Cambridge, MA: MIT Press.doi: 10.7551/mitpress/11920.001.0001

Russell-Rose, T., and Tate, T. (2013). *Designing the Search Experience: The Information Architecture of Discovery*. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Sim, S. E., Agarwala, M., and Umarji, M. (2013). "A Controlled Experiment on the Process Used by Developers During Internet-Scale Code Search," in *Finding Source Code on the Web for Remix and Reuse*, eds S. E. Sim and R. E. Gallardo-Valencia (New York, NY: Springer), 53–77. doi: 10.1007/978-1-4614-6596-6_4

Sim, S. E., Umarji, M., Ratanotayanon, S., and Lopes, C. V. (2011). How Well Do Search Engines Support Code Retrieval on the Web? *ACM Trans. Softw. Eng. Methodol.* 21:1–25. doi: 10.1145/2063239.2063243

Szabó, B. K. (2019). Rigged hand model for the Blender Game Engine. *Recent Innov. Mechatron.* 6, 1–7. doi: 10.17667/riim.2019.1/5

White, R. W., and Roth, R. A. (2009). *Exploratory Search: Beyond the Query-Response Paradigm. Synthesis Lectures on Information Concepts, Retrieval, and Services*. San Rafael, CA: Morgan & Claypool, 1–98. doi: 10.2200/S00174ED1V01Y200901ICR003

Xia, X., Bao, L., Lo, D., Kochhar, P. S., Hassan, A. E., and Xing, Z. (2017). What do developers search for on the web? *Empir. Softw. Eng.* 22, 3149–3185. doi: 10.1007/s10664-017-9514-4