What Do Developers Use the Crowd For?

A Study Using Stack Overflow

Rabe Abdalkareem, Emad Shihab, and Juergen Rilling,
Concordia University, Montreal

// Developers use crowdsourced knowledge to support development tasks and collect user feedback. The crowd is the most helpful on topics such as development tools and programming languages. The questions that take the longest to resolve are related to web frameworks. //

QUESTION-AND-ANSWER (Q&A) sites such as Stack Overflow are extremely popular with software developers. Typically, developers post questions on these sites, and one or more participants provide answers. In essence, the answering is outsourced to the crowd.[1]

Over the years, Q&A sites have evolved to do more than just answer questions. To determine their role in today's development lifecycle, we aimed to answer these questions:

- What do developers resort to the crowd on Stack Overflow for?
- For what topics is the crowd most helpful, and what topics take longer to obtain answers for?"

Other studies have had a similar goal (see the sidebar). But unlike them, we specifically examined cases in which a developer making a code commit on GitHub explicitly referenced a Stack Overflow post as a knowledge resource. The explicit mentioning of the posts

- gave us confidence that Stack Overflow was indeed related to a particular code commit;
- let us build a richer dataset (because we had a link to the commit and the associated Stack Overflow post); and
- indicated cases in which developers saw the need for traceability, justification, or documentation of their committed changes.

Moreover, our study didn't focus on a specific type of task (for example, bug triage[2]), domain (for example, mobile apps[3]), or programming language. Rather, we examined all types of commits that explicitly mentioned Stack Overflow. Doing so helped provide a more holistic view of the reasons developers use Stack Overflow.

Study Design and Approach

Here, we describe how we collected the Stack Overflow–related commits, classified them, and measured their helpfulness and delay (the time it took to receive an accepted answer).

Selecting the Projects

First, we needed to identify the software projects that contained Stack Overflow–related commits. Studying a large sample of software projects is important to improve confidence in

0740-7459/17/$33.00 © 2017 IEEE

RELATED WORK IN HOW DEVELOPERS USE STACK OVERFLOW

Christoph Treude and his colleagues qualitatively analyzed a sample of Stack Overflow questions.[1] They found that the developers used Stack Overflow to share knowledge, support development, learn new technologies, and search for solutions to both common and specific programming problems.

Anton Barua and his colleagues proposed a semiautomatic approach to study general topics discussed on Stack Overflow and their trends.[2] The most popular topics were web and mobile development. Kartik Bajaj and his colleagues used Stack Overflow data to analyze common challenges and misconceptions among web developers.[3] Christoffer Rosen and Emad Shihab investigated what mobile-app developers on Stack Overflow ask about.[4]

Bogdan Vasilescu and his colleagues established associations between GitHub and Stack Overflow users and found a correlation between participants' activities on the two platforms.[5]

Alexey Zagalsky and his colleagues investigated the R project's use of Stack Overflow and mailing lists as communication channels.[6] Both resources provided active communication channels in which participants were willing to help others. Stack Overflow employed crowd-based knowledge construction in which participants contributed knowledge independently, whereas mailing lists focused on improving specific answers.

References

1. C. Treude, O. Barzilay, and M.-A. Storey, "How Do Programmers Ask and Answer Questions on the Web?, *Proc. 33rd Int'l Conf. Software Eng.* (ICSE 11), 2011, pp. 804–807.
2. A. Barua, S.W. Thomas, and A.E. Hassan, "What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow," *Empirical Software Eng.*, vol. 19, no. 3, 2014, pp. 619–654.
3. K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining Questions Asked by Web Developers," *Proc. 11th Working Conf. Mining Software Repositories* (MSR 14), 2014, pp. 112–121.
4. C. Rosen and E. Shihab, "What Are Mobile Developers Asking About? A Large Scale Study Using Stack Overflow," *Empirical Software Eng.*, vol. 21, no. 3, 2016, pp. 1192–1223.
5. B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," *Proc. Int'l Conf. Social Computing* (SocialCom 13), 2013, pp. 188–195.
6. A. Zagalsky et al., "How the R Community Creates and Curates Knowledge: A Comparative Study of Stack Overflow and Mailing Lists," *Proc. 13th Int'l Workshop Mining Software Repositories* (MSR 16), 2016, pp. 441–451.

analysis results. We used the GHTorrent dataset[4] to obtain a list of nonforked projects (mainline) written in the most popular programming languages: Ruby, Python, JavaScript, PHP, Java, Scala, C, and C++.[5] On the basis of our selection criteria, we identified 4,163,814 projects.

However, GitHub contains many inactive or immature projects. So, to ensure that we considered only active, mature projects, we considered only projects that had

- at least 100 pull requests,
- at least three developers, and
- more than 100 commits in the past year.

Eirini Kalliamvakou and her colleagues recommended similar constraints.[6] This filtering further reduced the number of projects to 4,026, which we tried to clone for analysis. Because some projects were no longer available (for example, they were deleted or made private), we could clone and study only 3,974 projects.

Extracting the Commits

To detect the Stack Overflow–related commits, we relied on string pattern-matching techniques. We searched each project's commit logs for "stackoverflow" and its variants (a capitalized first letter, all capitalized, or with spaces). This produced 1,780 commits from 929 projects. Finally, one of us manually inspected the commits and their associated projects to filter out duplicates and irrelevant commits (false positives).

We were left with 1,414 commits from 808 projects (for a list of the projects, see das.encs.concordia.ca/wp-content/uploads/2016/12/list_of_studied_projects.csv). Approximately 97 percent of these commits contained a link to the Stack Overflow–related post. Table 1 shows statistics describing our dataset.

Classifying the Commits

Next, we performed iterative coding to identify and categorize the reasons developers used Stack Overflow.[7] We first inspected every

commit message, the source code associated with the commit, and the Stack Overflow post the commit referenced. We read the main issues discussed in the commit message and the post, which helped us determine how to classify the commit. Every time we added a new category, we reexamined the previously classified commits to determine whether the categorization changed. We ended up with 14 topics for which developers reused Stack Overflow posts, which we discuss later.

TABLE 1

Statistics related to our study's dataset.

Language	No. of projects	No. of Stack Overflow–related commits
JavaScript	189	307
Python	179	348
Ruby	132	227
Java	123	193
PHP	99	154
C/C++	65	131
Scala	21	54
Total	808	1,414

Measuring Helpfulness and Delay

As a proxy for helpfulness, we used the number of votes (the sum of the upvotes minus the downvotes) that a Stack Overflow question received. Our intuition was that if a question helped a developer, he or she would give it an upvote. Once we measured the votes for the individual posts, we grouped them according to the 14 topics. To measure the helpfulness for a topic, we determined the median number of votes for its posts.

Also, we believe that if it takes a long time for a post to obtain an accepted answer, the developer will be delayed more, which he or she will perceive negatively. So, we used the time to obtain an accepted answer for posts as a proxy for delay. For each topic, we measured the time from the initial post to its accepted answer. Then, we aggregated the values and determined the median time per topic.

The Results

We broke down our study into two main parts—one for each of the questions we discussed in the introduction.

Why Developers Resorted to Stack Overflow

Table 2 shows the topics for which developers reused Stack Overflow posts, grouped into five high-level categories: using knowledge, documenting bugs, promoting Stack Overflow, feature or system improvements, and code reuse. The category "other" includes commits that rarely appeared or didn't fit in any of the major categories.

Developers used Stack Overflow mainly to gain knowledge. The most frequent knowledge was related to programming languages (22.07 percent of the commits), API use (21.00 percent), configuration management (7.21 percent), web frameworks (6.51 percent), and web browsers (4.31 percent). The crowd's key role was to support and complement traditional documentation.

Developers also used Stack Overflow to document bugs (13.08 percent) and even feature or system improvements they implemented (1.77 percent). These findings show that the crowd on Stack Overflow did more than just provide knowledge or find relevant code. The crowd also provided insight on known issues and features users wanted to see. These results suggest that developers pay attention to such issues raised by the crowd and that Stack Overflow serves as medium for identifying and tracking feature requests and issues.

Another interesting topic involved the promotion of Stack Overflow in a project (3.18 percent), where developers would introduce a tag to facilitate documentation. Moreover, the direct reuse of code from Stack Overflow was minimal in terms of explicitly mentioning the Stack Overflow posts in commit messages (1.70 percent). However, we believe that developers reused more code than they admitted, for various reasons such as potential copyright violations or plagiarism.

Helpfulness and Delay

Table 3 shows the helpfulness and delay results for each of the 14 topics. The crowd was the most helpful for development tools, programming languages, implementation issues, and configuration management. The posts that took the longest to answer were related to web frameworks, documenting bugs, and development tools.

However, the most interesting analysis comes from plotting the

TABLE 2

The topics for which developers sought Stack Overflow help.

Topic	Subtopic	Description	Example	Percentage of the commits
Using knowledge	Programming languages	Knowledge of programming languages and their features—for instance, using `format` in Python, using a sequential `for` loop in JavaScript, how to do casting, a regular expression, or a programming-language limitation.	"Changed all Boolean casts that were using Boolean() function to use double negation(!!), which is faster: [Stack Overflow link]."	22.07
	API use	Knowledge of how to use an API, including arguments, deprecation, and specifying a method to perform a task in an API.	"mailutils: send_email() with attachments* Extends mailutils. send_email() API to support attachments, following the recipe: [Stack Overflow link]."	21.00
	Configuration management	Knowledge of configuration management—for example, how to configure the Maven tool in the development environment.	"Fix maven assembly warning about using root dir. It's a bad practice in Maven to define '/' as the output dir. It's better to leave it empty. See also [Stack Overflow link]."	7.21
	Web frameworks	Knowledge on using web frameworks and their configurations.	"Fix client names with dot do not work this is Spring Framework MVC behavior as described in [Stack Overflow link]."	6.51
	Web browsers	Knowledge regarding web browsers—for example, the presence or absence of features in specific browsers.	"Fix grid context menu position for Firefox. Firefox does not have offsetX. pageX is absolute and lets the menu jump all over the place. Solution based on [Stack Overflow link]."	4.31
	Development tools	Knowledge on configuring development tool (for example, IDEs, Git, and SVN) versions, settings, and so on.	"Update Git to delete a remote branch with '-delete' more memorable syntax. Use Git1.7 syntax based on this answer: [Stack Overflow link]."	4.17
	Implementation issues	Using suggestions or tutorials from Stack Overflow posts to implement an algorithm or a feature in projects without copying and pasting source code.	"Introduce the non-daemon process pool as an alternative to the original multiprocessing pool. This adds support for hierarchical multiprocessing (child classes can use multiprocessing again). The code is based on the following StackOverflow answer: [Stack Overflow link]."	3.89
	Database technologies	Knowledge of databases and their supporting technologies (for example, database configurations, maintenance, modeling, and queries).	"Postgres column renaming. Switched 'name' column name to 'shoreline_name' so we don't collide with possible Postgreskeywords/types [Stack Overflow link]."	2.83
	Operating systems	Knowledge of OS features or issues.	"Explicitly set empty extension name for backup files on Mac, this parameter is needed, otherwise an error is shown. See this SO post for more information: [Stack Overflow link]."	2.40

helpfulness rankings versus the delay rankings for each topic. In Figure 1, a bubble's size represents the number of commits for a particular topic. Questions related to topics such as implementation issues, programming languages, database technologies, and web browsers provided the highest utility; that is, they both were helpful and received accepted answers quickly. On the other hand, questions related to

TABLE 2 (CONT.)

The topics for which developers sought Stack Overflow help.

Topic	Description	Example	Percentage of the commits
Documenting bugs	Fixing a bug in the project and providing the link to the Stack Overflow post describing the bug.	"Fix AttributeError when IssueEvent has assignee. This was discovered by a user on StackOverflow [Stack Overflow link] and fixed as soon as I realized it was a bug."	13.08
Promoting Stack Overflow	Introducing a tag related to a project to facilitate its documentation or promote the tag's use.	"Promote stackoverflow for questions." "Drop google groups in favor of stackoverflow tag." "Link to [Stack Overflow link] for Q and A. Thanks to Vincent Scheib for arranging and Paul Kinlan for donating his Stack Overflow karma to create the tag."	3.18
Feature or system improvements	Implementing a new feature or improving a project on the basis of a Stack Overflow user's request.	"Extend key bindings for prompt commands to support predefined searches. This adds support for binding keys to ':/' and ':?', for example: bind stage 2: Based on this request by Joelpet on stackoverflow: [Stack Overflow link]"	1.77
Code reuse	Copying and pasting a source code snippet from a Stack Overflow post.	"Close tip popup on click outside the tip box. Credit: [Stack Overflow link]."	1.70
Other	Unidentifiable knowledge or (rarely) a commit that wasn't worth having a separate category for.	—	5.87

topics such as API use and operating systems tended to be answered quickly, but the developers didn't perceive the questions as helpful. Similarly, questions related to topics such as web frameworks and documenting bugs were perceived as less helpful and took even longer to obtain accepted answers. In such cases, the Stack Overflow crowd might not be the right resource for developers looking for answers.

Looking Ahead

On the basis of our study results, we offer the following three recommendations. The first two will help Stack Overflow designers enhance the site's features to meet the increasing demand from developers. The third will help developers improve traceability and documentation of their changes.

Provide Direct Feedback

Future versions of Stack Overflow must incorporate a mechanism with which developers can obtain direct feedback from the crowd. Currently, the role of the crowd is mainly to report bugs and request requirements. However, we envision that the crowd on Stack Overflow will increasingly be a source for refining requirements, providing testing, and even helping refine software design.

Assess Code Snippet Quality

We found that developers reused code snippets from Stack Overflow. Ensuring such snippets' quality or integrity is an area in which crowd-based platforms can do better (now all someone can do is give an upvote). Also, providing some sort of score that indicates a snippet's adaptability or ease of integration would be beneficial.

So, we recommend that Stack Overflow designers provide techniques to assess snippets' quality. For example, they should find ways to automatically generate test cases for snippets.

Link Changes to Discussions

Developers should link their commits to the discussions they used to help reach their final coded solutions. These discussions can help document and provide rationale for certain decisions. Such discussions could continue to evolve, so even if a bug was found in posted code, other developers could help provide an update or a fix.

Study Limitations

Our study had two main limitations. First, manual classification of commits is susceptible to error.

TABLE 3

Helpfulness and delay, classified by the topics in Table 2.

Topic	Helpfulness			Delay		
	No. of questions	Median no. of upvotes for a question	Ascending ranking	No. of accepted answers	Median time to receive an accepted answer (h)	Descending ranking
Development tools	61	39	1	52	6.7	9
Programming languages	310	29	2	279	0.3	1
Implementation issues	55	26	3	48	2.6	4
Configuration management	102	19	4	87	5.6	7
Database technologies	41	17	5	37	1.1	3
Web browsers	61	16	6	50	0.5	2
Web frameworks	93	15	7	77	13.0	11
Operating systems	37	13	8	33	0.5	2
API use	301	10	9	254	3.1	5
Code reuse	25	7	10	19	6.1	8
Feature or system improvements	27	3	11	18	3.6	6
Documenting bugs	178	2	12	139	9.4	10
Promoting Stack Overflow	0	0	N/A	0	0	N/A
Other	76	15	—	59	2.6	—

However, to determine our classification's validity, we got a PhD student to independently classify a statistically significant sample (302 commits) to reach a 95 percent confidence level using a 5 percent confidence interval. We used Cohen's kappa coefficient to evaluate the level of agreement between the two annotators. Cohen's kappa coefficient is a well-known statistical method for evaluating the inter-rater agreement level for categorical scales. The resulting coefficient is scaled to range between –1.0 and +1.0, where a negative value means poorer than chance agreement, zero indicates exactly chance agreement, and a positive value indicates better than chance agreement. The level of agreement between the annotators was +0.78, which is considered excellent.[8]

Second, as we mentioned before, our findings are based on commits in which developers explicitly mentioned Stack Overflow. Other cases might have existed in which developers used Stack Overflow but didn't mention it in the commit message.

Finally, because our study was based on open source projects hosted on GitHub, it might not

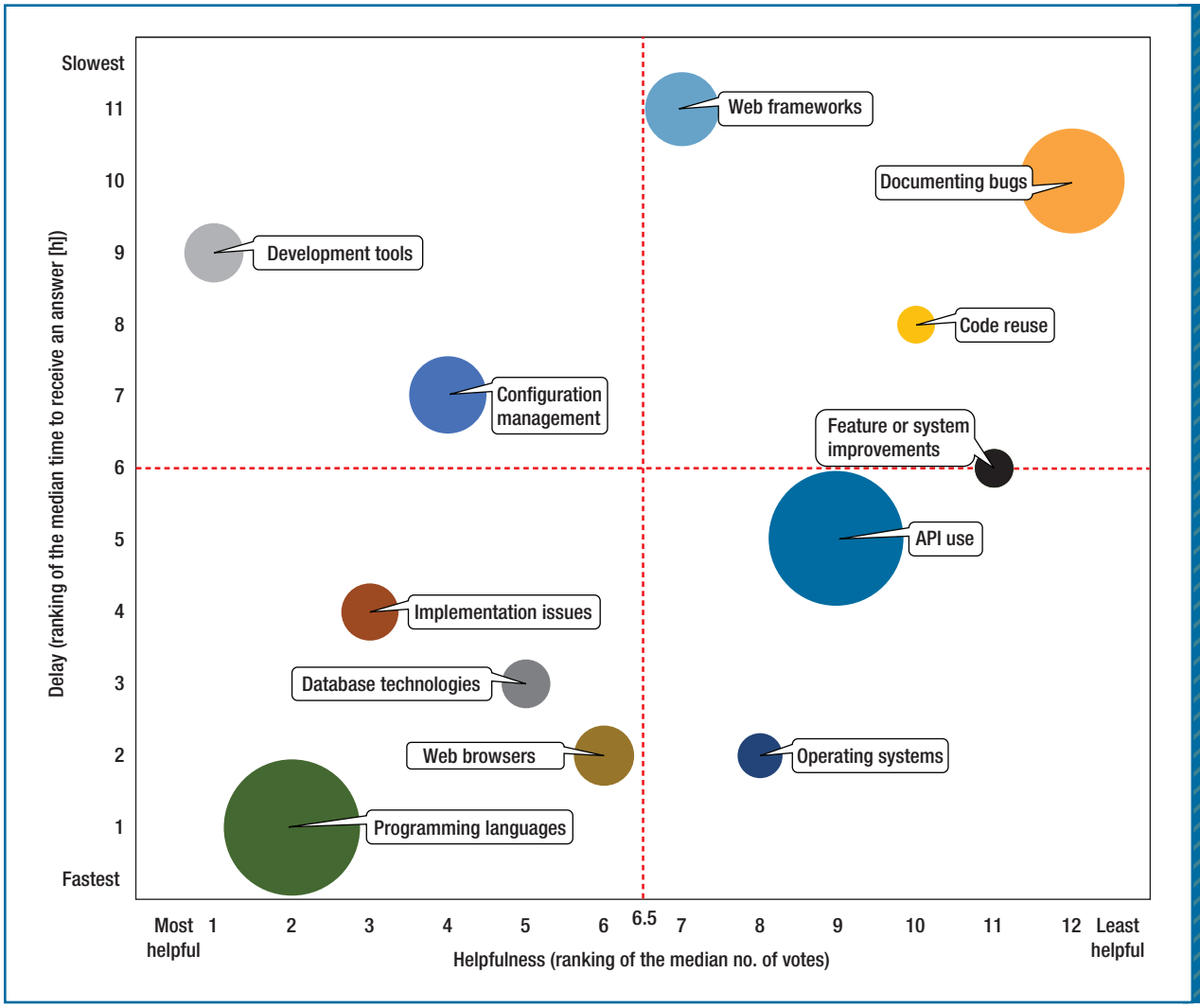


FIGURE 1. Helpfulness versus delay for the topics in Table 2. A bubble's size represents the number of commits.

generalize to other open source or commercial projects.

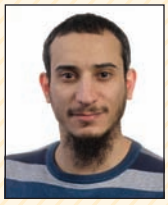
We plan to examine why developers perceive the crowd as more helpful for certain topics and why delays occur for answers in other topics. In particular, we plan to study how technological and sociotechnical factors affect the delivery of helpful and timely answers for different topics.

References

1. T.D. LaToza and A. van der Hoek, "Crowdsourcing in Software Engineering: Models, Motivations, and Challenges," *IEEE Software*, vol. 33, no. 1, 2016, pp. 74–80.
2. A.S. Badashian, A. Hindle, and E. Stroulia, "Crowdsourced Bug Triaging," *Proc. IEEE Int'l Conf. Software Maintenance and Evolution* (ICSME 15), 2015, pp. 506–510.
3. C. Rosen and E. Shihab, "What Are Mobile Developers Asking About? A Large Scale Study Using Stack Overflow," *Empirical Software Eng.*, vol. 21, no. 3, 2016, pp. 1192–1223.
4. G. Gousios, "The GHTorent Dataset and Tool Suite," *Proc. 10th Working Conf. Mining Software Repositories* (MSR 13), 2013, pp. 233–236.
5. B. Vasilescu et al., "Quality and Productivity Outcomes Relating to Continuous Integration in GitHub," *Proc. 10th Joint Meeting Foundations of Software Eng.* (ESEC/FSE 15), 2015, pp. 805–816.

6. E. Kalliamvakou et al., "The Promises and Perils of Mining GitHub," *Proc. 11th Working Conf. Mining Software Repositories* (MSR 14), 2014, pp. 92–101.
7. C.B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Trans. Software Eng.*, vol. 25, no. 4, 1999, pp. 557–572.
8. J. Fleiss, "The Measurement of Interrater Agreement," *Statistical Methods for Rates and Proportions*, John Wiley & Sons, 1981, pp. 212–236.

ABOUT THE AUTHORS



RABE ABDALKAREEM is a PhD student in the Department of Computer Science and Software Engineering at Concordia University, Montreal. His research investigates how the adoption of crowdsourced knowledge affects software development and maintenance. Abdalkareem received an MSc in computer science from Concordia University. Contact him at rab_abdu@encs.concordia.ca.

EMAD SHIHAB is an assistant professor in the Department of Computer Science and Software Engineering at Concordia University, Montreal. His research interests are software quality assurance, mining software repositories, technical debt, mobile applications, and software architecture. Shihab received a PhD in computer science from Queen's University in Kingston. He's a member of IEEE and ACM. Contact him at emad.shihab@encs.concordia.ca.

JUERGEN RILLING is a professor and the director of the Software Engineering Research Center in the Department of Computer Science and Software Engineering at Concordia University, Montreal. His research focuses on supporting the evolution, modeling, and analysis of global software ecosystems. Rilling received his PhD in computer science from the Illinois Institute of Technology. He's a member of the IEEE Computer Society. Contact him at juergen.rilling@concordia.ca.