How Do Programmers Ask and Answer Questions on the Web? (NIER Track)

Christoph Treude
Dept. of Computer Science
University of Victoria
Victoria, Canada
ctreude@uvic.ca

Ohad Barzilay
Blavatnik School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel
ohadbr@tau.ac.il

Margaret-Anne Storey
Dept. of Computer Science
University of Victoria
Victoria, Canada
mstorey@uvic.ca

ABSTRACT

Question and Answer (Q&A) websites, such as Stack Overflow, use social media to facilitate knowledge exchange between programmers and fill archives with millions of entries that contribute to the body of knowledge in software development. Understanding the role of Q&A websites in the documentation landscape will enable us to make recommendations on how individuals and companies can leverage this knowledge effectively. In this paper, we analyze data from Stack Overflow to categorize the kinds of questions that are asked, and to explore which questions are answered well and which ones remain unanswered. Our preliminary findings indicate that Q&A websites are particularly effective at code reviews and conceptual questions. We pose research questions and suggest future work to explore the motivations of programmers that contribute to Q&A websites, and to understand the implications of turning Q&A exchanges into technical mini-blogs through the editing of questions and answers.

Categories and Subject Descriptors

H.m [**Information Systems**]: Miscellaneous

General Terms

Documentation, Human Factors

Keywords

Q&A, Social Media, Questions, Stack Overflow

1. INTRODUCTION AND MOTIVATION

The design and implementation of software systems requires knowledge that is often distributed among many individuals with different areas of expertise and capabilities. Software development has been described as knowledge-intensive [10] and knowledge management plays a central role in many software organizations.

The success of social media has introduced new ways of exchanging knowledge via the Internet. Q&A websites such as Yahoo! Answers[1], Answers.com[2] or the recently created Facebook Questions[3] are founded on the success of social media and built around an "architecture of participation" [9] where user data is aggregated as a side-effect of using Web 2.0 applications. Questions and answers on Q&A websites represent archives with millions of entries that are of value to the community [3]. For the domain of software development, the website Stack Overflow[4] facilitates the exchange of knowledge between programmers connected via the Internet. In the 2 years since its foundation in 2008, more than 1 million questions have been asked on Stack Overflow, and more than 2.5 million answers have been provided.

Answers on Stack Overflow often become a substitute for official product documentation when the official documentation is sparse or not yet existent[5]. To facilitate the crowd-sourcing of documentation, the Stack Overflow community explicitly encourages contributions where the person asking the question also provides an answer. In these cases, the Q&A portal becomes a platform for technical mini-blogs[6].

Despite such widespread use of Q&A websites, the role of Q&A websites in the software documentation landscape is not well understood. It is unclear what kinds of questions get answered well and which ones remain unanswered, as well as how to phrase questions effectively. We do not know how helpful the questions and answers are to a broader community nor what the implications are from the duality of a Q&A website as a question and documentation portal.

In this paper, we pose research questions and report preliminary results to identify the role of Q&A websites in software development using qualitative and quantitative research methods. Our preliminary findings, obtained through the analysis of archival data from Stack Overflow and qualitative coding, indicate that Q&A websites are particularly effective at code reviews, explaining conceptual issues and answering newcomer questions. The most common use of Stack Overflow is for how-to questions, and its dominant programming languages are C#, Java, PHP and JavaScript. Future work lies in enhancing these results and understanding the motivations of programmers to contribute content

[1]http://answers.yahoo.com/
[2]http://www.answers.com/
[3]http://www.facebook.com/questions/
[4]http://stackoverflow.com/
[5]https://stackoverflow.fogbugz.com/default.asp?W25450
[6]http://meta.stackoverflow.com/questions/2706

to a Q&A website. Ultimately, understanding the processes that lead to the creation of knowledge on Q&A websites will enable us to make recommendations on how individuals and companies, as well as tools for programmers, can leverage the knowledge and use Q&A websites effectively. We will also shed light on the credibility of documentation on these websites.

The remainder of this paper is structured as follows. We review related work in Section 2, and we introduce the key concepts of Stack Overflow in Section 3. Section 4 identifies our research questions and our methodology. We report and discuss preliminary findings in Sections 5 and 6, and we highlight the impact of our work in Section 7.

2. BACKGROUND AND RELATED WORK

Work related to our research can be divided into work on Q&A websites in general, and work on questions that software developers ask.

In particular the use of Yahoo! Answers has been studied by several researchers. Gyöngyi *et al.* [5] identified three fundamental ways in which Yahoo! Answers is used: for focused questions, to trigger discussions, and for random thoughts and questions. Adamic *et al.* [1] found that users who focused on certain areas of expertise often got the best ratings. In order to find high quality content, Agichtein *et al.* [2] introduce a framework that is able to separate high quality items from the rest. In a related project, Shah and Pomerantz [11] found that contextual information such as a user's profile can be used to predict content quality.

While researchers have not yet studied what kind of questions software developers ask using Q&A websites, there has been work on questions asked by software developers in general. Letovsky [8] identified five main question types: why, how, what, whether and discrepancy. Fritz and Murphy [4] provide a list of questions that focus on issues that occur within a project. Sillito *et al.* [12] provide a similar list focusing on questions during evolution tasks. LaToza and Myers [7] found that the most difficult questions from a developer's perspective dealt with intent and rationale. In their study on information needs in software development, Ko *et al.* [6] found that the most frequently sought information included awareness about artifacts and coworkers.

In contrast to the settings of these studies, Q&A websites provide a platform for questions aimed at a general audience that is not part of the same project. Q&A websites contain questions, but can also contain answers to anticipated questions as well as opinions through comments and ratings.

3. STACKOVERFLOW.COM

Stack Overflow is centered around nine design decisions[7]: **Voting** is used as a mechanism to distinguish good answers from bad ones. Users can up-vote answers they like, and down-vote answers they dislike. In addition, the user asking a question can accept one answer as the official answer. **Tags** are used to organize questions. Users have to attach at least one tag and can attach up to five tags when asking a question. **Editing** of both questions and answers allows users to improve their quality and to turn Q&A exchanges into wiki-like structures. **Badges** are given to users to reward them for their contributions once they reach certain thresholds. This form of **karma** is used to encourage contribution. **Pre-Search** helps avoid duplicate questions by showing similar entries as soon as a user has finished typing the title of a question. Stack Overflow was designed to be used such that **Google is UI**. Web pages on stackoverflow.com are optimized towards search engines and **performance**. To ensure **critical mass**, several programmers were explicitly asked to contribute in the early stages of Stack Overflow.

[7]http://www.youtube.com/watch?v=NWHfY_lvKIQ

4. RESEARCH METHODOLOGY

This section describes our methodology by outlining our research questions as well as our data collection and analysis methods. We pose five overarching research questions about the role of Q&A websites in software development. In this short paper, we focus on the first two questions.

1. What kinds of questions are asked on Q&A websites for programmers?
2. Which questions are answered and which ones remain unanswered?
3. Who answers questions and why?
4. How are the best answers selected?
5. How does a Q&A website contribute to the body of software development knowledge?

Our data collection follows a mixed-methods approach, collecting both quantitative and qualitative data. We created a script to extract questions along with all answers, tags and owners using the Stack Overflow API. The data reported in this paper was extracted on November 23, 2010 and contains all questions that were asked between November 1, 2010 and November 15, 2010. The amount of data that we extracted is provided in Table 1.

Table 1: Extracted Data

data item	amount
questions	38,419
owners	31,729
answers	68,467
tag instances	111,408

To answer our research questions, we analyzed quantitative properties of questions, answers and tags, and we applied qualitative codes to a sample of tags and questions. Qualitative coding was done individually and then codes were confirmed in collaborative coding sessions.

5. PRELIMINARY FINDINGS

In this section, we report our preliminary findings to the first two research questions.

Different Kinds of Questions.

To analyze the different kinds of questions asked on Stack Overflow, we did qualitative coding of questions and tags. The tags were mainly used to learn about the topics covered by Stack Overflow, while the question coding gave insight into the nature of the questions.

Each question has between one and five tags that are set by the person asking a question. Most questions (72.30%) have between 2 and 4 tags. 10,272 different keywords were used to tag questions, and there were 111,408 instances of a tag being applied to a question. Table 2 shows the most used tags.

Table 2: Most Used Tag Keywords

tag keyword	instances
c#	3,765
java	2,909
php	2,599
javascript	2,310
jquery	2,084

We applied qualitative coding to the 200 most frequently used tag keywords in our data. These keywords covered 60,193 of the tag instances (54.03%). We identified five categories of tags that are shown in Table 3. We show the number of instances in each category, the number of different keywords per category, and highlight the most used tags per category as examples. For the keyword "homework", we did not find any related tags and thus left it uncategorized.

Table 3: Tag Coding

code	keyw.	inst.	examples
prog. lang.	63	28,218	c#, java
framework	48	11,532	jquery, ruby-on-rails
environm.	45	14,127	android, iphone
domain	29	4,125	regex, database
non-funct.	14	2,071	multithreading
homework	1	120	homework

Users self-code their questions through tags to index their questions and to allow for navigation to their questions. Tags reveal the topics covered on Stack Overflow, but only allow limited insights into the nature of the questions asked. To further understand the characteristics of questions on Stack Overflow, we coded a random sample of 385 questions from our data set (1%). We analyzed the titles and body texts of these questions and found the following categories, ordered by their frequency:

how-to. Questions that ask for instructions, e.g. *"How to crop image by 160 degrees from center in asp.net"*.

discrepancy. Some unexpected behavior that the person asking the question wants explained, e.g. *"iphone - Coremotion acceleration always zero"*.

environment. Questions about the environment either during development or after deployment, e.g. *"How to use windows emacs as a svn client?"*.

error. Questions that include a specific error message, e.g. *"C# Obscure error: file " could not be refactored"*.

decision help. Asking for an opinion, e.g. *"Should a business object know about its corresponding contract object"*.

conceptual. Questions that are abstract and do not have a concrete use case, e.g. *"Concept of xml sitemaps"*.

review. Questions that are either implicitly or explicitly asking for a code review, e.g. *"Simple file download via HTTP - is this sufficient?"*.

non-functional. Questions about non-functional requirements such as performance or memory usage, e.g. *"Mac - Max Texture Size for compatibility?"*.

novice. Often explicitly states that the person asking the question is a novice, e.g. *"Oracle PL/SQL performance tuning crash course"*.

noise. Questions not related to programming, e.g. *"Apple Developer Program"*.

Most questions in our random sample fit into one of these categories, but for some of the questions (9.61%), we assigned two categories. The most frequent type of question (39.22%) was **how-to**, followed by questions about **discrepancies** and **environment**. The first two columns of Table 4 show the detailed results of our coding.

Table 4: Question Coding

		answered		no
code	**sum**	**accepted**	**not acc.**	**answer**
how-to	151	67 (44%)	63 (42%)	21 (14%)
discrepancy	50	27 (54%)	11 (22%)	12 (24%)
environment	40	13 (33%)	17 (43%)	10 (25%)
error	36	19 (53%)	14 (39%)	3 (8%)
decision help	22	9 (41%)	10 (45%)	3 (14%)
conceptual	18	10 (56%)	7 (39%)	1 (6%)
how-to/nov.	16	10 (63%)	3 (19%)	3 (19%)
review	13	12 (92%)	1 (8%)	0 (0%)
non-funct.	10	6 (60%)	1 (10%)	3 (30%)
novice	5	2 (40%)	3 (60%)	0 (0%)
other	24	10 (42%)	11 (46%)	3 (13%)
sum	385	185 (48%)	141 (37%)	59 (15%)

Which Questions Are Answered and Which Are Not.

Figure 1 shows the distribution of answers per question. The number of answers per question is shown on the x-axis, and the number of questions with that number of answers is shown on the y-axis using a log scale. 5,450 (14.19%) questions were not answered. The remaining questions had at least one and up to 23 answers. Only 3,243 out of 68,467 answers (4.74%) were provided by the same person that had asked the question.

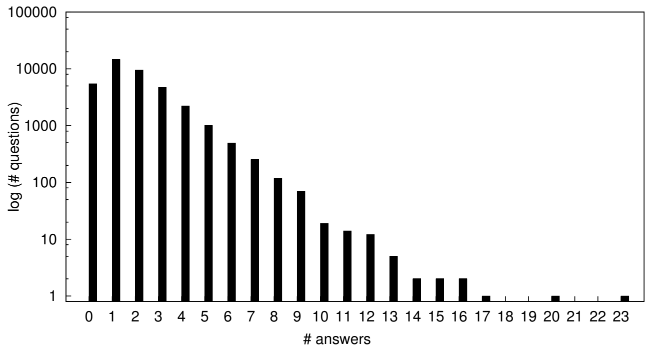

Figure 1: Answers Per Question (Log Scale)

On Stack Overflow, the user who is asking a question can mark at most one answer per question as *accepted*. We use this feature to examine the implications of different question characteristics on the success of a question. We define successful and unsuccessful questions as follows: A **successful question** has an accepted answer, and an **unsuccessful question** has no answer. Following these definitions, we further analyzed the 185 successful questions and 59 unsuccessful questions from our random sample of 385 questions. Table 4 shows the number of questions per category for all questions in our random sample, for all successful questions, for all questions with answers but no accepted answer, and for all questions without an answer.

It is interesting to note that the community answered **review**, **conceptual** and **how-to / novice** questions more frequently than other kinds of questions.

6. DISCUSSION

A possible reason for the high answer ratio of review questions is the fact that review questions are usually very concrete. They contain code snippets, and often no external sources are necessary to understand the code and make a recommendation about its quality. Also, code review questions can have more than one "correct" answer, and often any input is better than no input. The knowledge required to answer conceptual questions is usually broad. It is available in documentation or books and only needs to be presented effectively. Novices are easy to sympathize with and their questions are usually easy to answer.

The type of question is not the only factor for getting good answers. Other factors seem to include: the technology in question, the identity of the user, the time and day in which the question was asked, whether the question included a code snippet, or the length of the question. In future research we will further investigate these factors.

As with any research methodology, there are limitations with our choice of methods. The first limitation lies in the small amount of data we analyzed in our random sample. However, by triangulating our findings through qualitative coding of tags and questions, we were able to mitigate some of these concerns. Our definitions of successful and unsuccessful questions are limited. They offer a first approximation, but in future research we will have to analyze the answers in more detail to understand the differences between successful and unsuccessful uses of Q&A websites. Our conclusions are limited to Stack Overflow and its current use. However, Stack Overflow has managed to attract a large group of users and is one of the first Q&A websites specifically for software developers. Further studies will have to be conducted to analyze the use of other Q&A websites.

7. IMPACT AND FUTURE WORK

Understanding the interactions on Q&A websites, such as Stack Overflow, will shed light on the information needs of programmers outside closed project contexts and will enable recommendations on how individuals, companies and tools can leverage knowledge on Q&A websites.

Our preliminary findings indicate that Stack Overflow is particularly effective at code reviews, for conceptual questions and for novices. The most common questions include how-to questions and questions about unexpected behaviors. Next, we plan to investigate who the participants on Q&A websites are and what motivates them to contribute. In addition, understanding the role and effectiveness of ratings to identify the best answers and the role of comments to facilitate discussion are important venues for future research. We also plan to look at different data sets to understand trends over time such as emerging topics and correlate that data with Web activity around these topics in general.

Q&A websites have an impact on the software documentation landscape. With the edit functionality for questions and answers, Q&A exchanges can turn into wiki-like structures. Our preliminary results show that 15,230 out of 38,419 questions (39.64%) and 13,120 out of 68,467 answers (19.16%) have been edited afterwards. Future work will have to investigate the implications of this blurred line between Q&A websites and technical mini-blogs.

8. REFERENCES

[1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *WWW '08: Proc. of the 17th Intl. Conf. on World Wide Web*, pages 665–674, New York, NY, USA, 2008. ACM.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proc. of the Intl. Conf. on Web search and web data mining*, WSDM '08, pages 183–194, New York, NY, USA, 2008. ACM.

[3] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *WWW '08: Proc. of the 17th Intl. Conf. on World Wide Web*, pages 467–476, New York, NY, USA, 2008. ACM.

[4] T. Fritz and G. C. Murphy. Using information fragments to answer the questions developers ask. In *Proc. of the 32nd Intl. Conf. on Software Engineering - Volume 1*, ICSE '10, pages 175–184, New York, NY, USA, 2010. ACM.

[5] Z. Gyöngyi, G. Koutrika, J. Pedersen, and H. Garcia-Molina. Questioning yahoo! answers. Technical Report 2007-35, Stanford InfoLab, 2007.

[6] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *Proc. of the 29th Intl. Conf. on Software Engineering*, ICSE '07, pages 344–353, Washington, DC, USA, 2007. IEEE.

[7] T. D. LaToza and B. A. Myers. Hard-to-answer questions about code. In *Proc. of the 2nd Workshop on the Evaluation and Usability of Programming Languages and Tools*, 2010.

[8] S. Letovsky. Cognitive processes in program comprehension. In *Proc. of the 1st Workshop on empirical studies of programmers*, pages 58–79, Norwood, NJ, USA, 1986. Ablex Publishing Corp.

[9] T. O'Reilly. What is Web 2.0: Design patterns and business models for the next generation of software, 2005, accessed in December 2010. http://oreilly.com/web2/archive/what-is-web-20.html.

[10] P. N. Robillard. The role of knowledge in software development. *Commun. ACM*, 42(1):87–92, 1999.

[11] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community QA. In *Proc. of the 33rd Intl. Conf. on Research and development in information retrieval*, SIGIR '10, pages 411–418, New York, NY, USA, 2010. ACM.

[12] J. Sillito, G. C. Murphy, and K. De Volder. Questions programmers ask during software evolution tasks. In *Proc. of the 14th Intl. Symp. on Foundations of software engineering*, SIGSOFT '06/FSE-14, pages 23–34, New York, NY, USA, 2006. ACM.