Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

9-2017

Understanding stack overflow code fragments

Christoph TREUDE
Singapore Management University, ctreude@smu.edu.sg

Martin P. ROBILLARD

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Databases and Information Systems Commons, and the Software Engineering Commons

Citation

TREUDE, Christoph and ROBILLARD, Martin P.. Understanding stack overflow code fragments. (2017). *Proceedings of the 33rd International Conference on Software Maintenance and Evolution, Shanghai, China, 2017 September 17-22*. 509-513.
Available at: https://ink.library.smu.edu.sg/sis_research/8876

This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Understanding Stack Overflow Code Fragments

Christoph Treude
School of Computer Science
University of Adelaide
Adelaide, SA, Australia
christoph.treude@adelaide.edu.au

Martin P. Robillard
School of Computer Science
McGill University
Montréal, QC, Canada
martin@cs.mcgill.ca

***Abstract*—Code fragments posted in answers on Q&A forums can form an important source of developer knowledge. However, effective reuse of code fragments found online often requires information other than the code fragment alone. We report on the results of a survey-based study to investigate to what extent developers perceive Stack Overflow code fragments to be self-explanatory. As part of the study, we also investigated the types of information missing from fragments that were not self-explanatory. We find that less than half of the Stack Overflow code fragments in our sample are considered to be self-explanatory by the 321 participants who answered our survey, and that the main issues that negatively affect code fragment understandability include incomplete fragments, code quality, missing rationale, code organization, clutter, naming issues, and missing domain information. This study is a step towards understanding developers' information needs as they relate to code fragments, and how these needs can be addressed.**

I. Introduction

The Internet has provided software developers with an effective infrastructure for sharing and accessing programming knowledge. There currently exist dozens of websites hosting curated collections of code examples, including high-profile hubs such as Gist [1] and MSDN [2]. Although large catalogs of curated code samples provide undeniable value to developers, their reliance on *explicitly contributed* samples inevitably limits the supply. As software development problems and technology quickly evolve, it is likely that new programming situations will arise that do not have any supporting samples.

The limitation of curated collections of code samples motivates the exploration of more dynamic developer social media as a potential source of code examples. The prime example of a dynamic forum is the Question and Answer website Stack Overflow, with more than 14 million questions and more than 22 million answers as of July 2017. Many of the questions and answers contain code fragments. Can we opportunistically use code *fragments* in Stack Overflow answers as code *samples* that can support development tasks other than their related question?

The vast majority of code fragments on Stack Overflow are accompanied by natural language text that complements the code in the answer. However, it can be difficult for readers to estimate, at first glance, the proportion of this text that is dedicated to general explanation of the code, as opposed to answering the specific needs to the original poster. For this reason, the text may often be skipped wholesale by readers looking for a quick answer to a *how-to* programming question.

We currently have only a limited understanding of the information developers need to understand code fragments posted in Stack Overflow answers, if any. Current related work provides some insights, but only indirectly (see Section V).

The ultimate goal of our work is the investigation of the information developers need to understand code fragments. As a first step towards this goal, we conducted a survey-based study of the extent to which code fragments on Stack Overflow are self-explanatory based on the question's title alone, and what information is missing if only the title is given as context.

We sampled code fragments from Stack Overflow, removed all explanatory text, and asked developers, through a web-based survey, to indicate the extent to which the code fragments were self-explanatory and what information was missing. We received responses from 321 participants, indicating that less than half of the code fragments that we had sampled from Stack Overflow were completely self-explanatory. We also found that whether a code fragment was considered to be self-explanatory depends on factors such as incompleteness, code quality, missing rationale, code organization, clutter, naming issues, and missing domain information.

Our findings shed light on what authors can do to enable effective reuse of code fragments on Stack Overflow and elsewhere. They also lay the groundwork for more structured approaches to code fragment documentation that could ensure that each potential issue exhibited by a code fragment is properly addressed.

II. Methodology

We present our research questions followed by a description of our data collection and analysis methods.

A. Research Questions

The work presented in this paper was guided by the following two research questions:

RQ1. To what extent are Stack Overflow code fragments self-explanatory when accompanied with thread titles?

RQ2. What information is missing if only thread titles are given as context?

Additional research questions for extending this work in the future will include studying whether the missing information is found elsewhere in the Stack Overflow thread and determining what information is superfluous for code fragment reuse.

B. Data Collection

To answer our research questions, we sampled code fragments from Stack Overflow, stripped away all explanatory text accompanying these code fragments (including code comments), and asked software developers to rate the code fragments in terms of how self-explanatory they were and what information was missing.

Specifically, we sampled 120 code fragments from Stack Overflow using a stratified sampling strategy. For twelve topics that we chose to represent a wide variety of aspects of the Java API, we sampled the ten most recent Stack Overflow threads that had the words "how to" in the title, had an accepted answer, and were not closed. The focus on "how to" questions was motivated by previous work by Nasehi et al. [3], who identified that questions on Stack Overflow can be classified in a "dimension that is about the main concern of the questioners and what they wanted to solve" and the subsequent work of de Souza et al. [4] who developed classifiers to separate "how to" questions from other question types, such as conceptual questions and debugging questions. We assume that the most useful code fragments are likely to be found in responses to "how to" questions.

In this exploratory study, we focus on code fragments in the Java programming language. We filtered out threads for which the accepted answer did not contain Java code, threads that contained two or more almost identical alternative code fragments, threads where the code elements mentioned in the accepted answer explicitly referenced code elements from the question, and threads where code was of excessive length. This filtering was done manually by the first author. Table I (following page) shows the details of the sampling: *Topic* indicates the particular aspect of the Java API, *Tags* shows the Stack Overflow tags we used for the search, *Results* shows how many results Stack Overflow returned in total for "how to" questions with these tags and an accepted answer, *Analyzed* shows how many threads we needed to investigate until we found ten threads that fulfilled our criteria, and the remaining columns show how many threads we discarded for a specific reason. On average, we analyzed 19 threads per topic to find 10 suitable threads for our study, with the majority of threads being filtered out due to the accepted answer not containing code. The last column shows the date on which the query was conducted.

To enable a comparison between our findings for Stack Overflow and a curated source of code fragments, we also sampled 40 code fragments from Kode Java [5], a popular website that provides Java programming examples, articles, and tutorials. We employed a similar stratified sampling strategy, focusing on four of the topic areas that we had considered for Stack Overflow: Regex, Util, Spring, and JDBC. We downloaded all 1,039 posts on Kode Java as of 27-Oct-15. For each of the selected topics, we determined its corresponding category on Kode Java (e.g., Util is referred to as "Java Util Package"), determined the median number of lines of code for all posts related to this topic, and randomly sampled five posts with less code than the median and five posts with more code than the median, resulting in ten code fragments for each of the four topics. We followed this procedure to increase the representativeness of our sample. Table II shows the corresponding descriptive statistics.

We then randomly assigned each of the 160 code fragments (120 from Stack Overflow plus 40 from Kode Java) to three different survey instruments for a total of 160 surveys with three randomly selected code fragments each. We handled surveys for Stack Overflow and Kode Java separately. In addition to demographic questions about participants' main occupation, their development experience in Java, and their use of English, we asked two categorical questions about each code fragment: "How familiar are you with the task <thread title> in Java?" and "How self-explanatory do you find the code example?". The thread title was either the title of the question on Stack Overflow or the title of the tutorial on Kode Java. To increase the reliability of the answers to the question about task familiarity, the code fragment was not visible until participants had answered this question. Table III shows the answer options that the survey provided for these two categorical questions.[1] In addition, for each code fragment, the survey contained the following open-ended question: "If you didn't answer 'Everything is completely self-explanatory', name one thing that's missing."

We sent each of these 160 surveys to 10 randomly sampled GitHub users that had at least one public activity in the last 12 months, and additional surveys were sent out until each code fragment had received at least three ratings. For the 40 Kode Java surveys, we did not need to send out any additional surveys, whereas for Stack Overflow, we sent out an additional 26 batches of 10 emails until we had received at least three responses for each code fragment. In total, we received 321 survey responses (response rate 17.3%) distributed over 160 code fragments, with each response containing answers for three code fragments. At the end of the data collection process, our data set contained 963 code fragment–response mappings.

C. Data Analysis

In addition to the use of descriptive statistics to answer our first research question about the extent to which code fragments are self-explanatory, we qualitatively coded responses to the open-ended survey questions to answer our second research question about what prevents code fragments from being self-explanatory. The two authors of this paper coded all 248 answers to "If you didn't answer 'Everything is completely self-explanatory', name one thing that's missing" (207 out of 723 possible answers for code fragments from Stack Overflow and 41 out of 240 possible answers for code fragments from Kode Java). This coding was done iteratively by coding responses for one topic area at a time and refining

[1] An example survey is available at http://tinyurl.com/code-fragment-survey. We designed the answer options to be more meaningful than a Likert scale.

TABLE I
CODE FRAGMENT SAMPLING FROM STACK OVERFLOW

Topic	Tags	Results	Analyzed	No Code	Alternatives	Reference	Length	Date
JDBC	java, jdbc	757	18	4	1	3		30-Dec-15
Regex	java, regex	1094	13	3				30-Dec-15
Spring	java, spring	2832	25	13	2			30-Dec-15
Util	java, java-util-scanner OR java-util-logging	194	12			2		30-Dec-15
Hibernate	java, hibernate	1763	14	4				25-Apr-16
IO	java, java-io	69	27	14	3			25-Apr-16
Java2D	java, java-2d	58	14	2	1		1	25-Apr-16
JavaMail	java, javamail	125	37	25	2			25-Apr-16
JAXB	java, jaxb	409	22	9	3			25-Apr-16
Jodatime	java, jodatime	154	15	3	2			25-Apr-16
Logging	java, java.util.logging	28	17	3	4			25-Apr-16
Reflection	java, reflection	475	14	2	2			25-Apr-16

TABLE II
CODE FRAGMENT SAMPLING FROM KODE JAVA

Topic	Category	Posts	median LOC
JDBC	JDBC API	62	34.5
Regex	Regular Expressions	25	21
Spring	Spring Core	32	48
Util	Java Util Package	104	19

TABLE III
SURVEY ANSWER OPTIONS

How familiar are you with the task <thread title> in Java?
I have no idea how to go about accomplishing this
I don't really know how to accomplish this but would know exactly what information to search for
I know how to accomplish this but would need to look up some details
I can accomplish this without looking anything up
How self-explanatory do you find the code example above?
Cryptic
Some parts are self-explanatory
Most of it is self-explanatory
Everything is completely self-explanatory

the coding guide after each step. The following is the final list of issues that emerged from the analysis:

- **Organization**: The organization of the code (structure, indentation, modules, etc.) is inadequate.
- **Naming**: The choice of identifier naming is not helpful.
- **Rationale**: The reason for doing something is not self-evident, including the meaning of literal values ("magic numbers").
- **Incomplete**: The code fragment is clearly missing other code (such as calling or integration code) that is necessary for understanding, including unbound variables.
- **Quality**: There is some obvious problem with the actual code or the thread title.
- **Domain**: Specialized or context-specific domain concepts and/or terminology are not self-evident.
- **Clutter**: The code fragment is too verbose given the task it accomplishes.

After the initial coding, we asked two annotators who are not authors of this paper to go through all 160 code fragments and annotate each fragment in terms of whether it exhibited any of these 7 issues. This annotation was initially done separately, resulting in 68 cases where both annotators agreed that a code fragment exhibited a particular issue, 834 cases where both annotators agreed that a code fragment did not exhibit a particular issue, and 218 cases where the annotators disagreed, for a total of 1,120 cases (160 code fragments multiplied by 7 issues). We asked both annotators to comment on the cases where they disagreed, and the first author resolved the disagreements based on these comments, for a total of 207 cases where a code fragment exhibited an issue and 913 cases where it did not.

III. FINDINGS

We present our findings organized by research questions.

A. *Self-Explanatory Code Fragments*

Table IV presents the results for our first research question about the extent to which code fragments are self-explanatory based on thread titles. For Stack Overflow, 49% of the answers indicated that a code fragment was completely self-explanatory, while another 28% of the answers indicated that most of the code was self-explanatory. The highest ratios of "Cryptic" ratings were assigned to code fragments from Java2D and Hibernate, while code fragments from Jodatime were rated as most self-explanatory.

Code fragments were more likely to be rated as self-explanatory if the responder was familiar with the corresponding task (Chi-Squared test on a 4×4 matrix, $p < 0.00001$, Cramer's V $= 0.24$, moderate association). We also found that the source of a code fragment (Stack Overflow vs. Kode Java) was a significant variable. Specifically, responders were more likely to be familiar with tasks described on the curated tutorial website Kode Java than on Stack Overflow (Chi-Squared test on a 4×2 matrix, $p < 0.00001$, Cramer's V $= 0.25$, moderate association). Similarly, participants were more likely to consider a code fragment from Kode Java to be self-explanatory (Chi-Squared test on a 4×2 matrix, $p < 0.002$, Cramer's V $= 0.34$, moderate association). We speculate that most of this difference results from the amount of effort that has been spent on curating code fragments in each source.

TABLE IV
SURVEY RESULTS

Source	Topic	I have no idea...	I don't really know...	I know how to accomplish this but...	I can accomplish this...	Answers	Cryptic	Some parts are self-explanatory	Most of it is self-explanatory	Everything is completely self-explanatory	Answers	Open-ended answers
Stack Overflow	Hibernate	30%	26%	35%	9%	57	17%	20%	39%	24%	54	19
Stack Overflow	IO	4%	22%	49%	25%	55	9%	15%	30%	46%	54	17
Stack Overflow	Java2D	31%	39%	31%	0%	59	18%	4%	30%	49%	57	13
Stack Overflow	JavaMail	23%	34%	36%	7%	44	7%	12%	28%	53%	43	13
Stack Overflow	JAXB	35%	37%	25%	4%	52	13%	21%	33%	33%	52	24
Stack Overflow	JDBC	15%	35%	40%	10%	78	5%	13%	27%	55%	77	22
Stack Overflow	Jodatime	16%	25%	41%	18%	61	7%	11%	20%	62%	61	15
Stack Overflow	Logging	8%	38%	36%	18%	50	6%	8%	28%	58%	50	15
Stack Overflow	Reflection	13%	47%	32%	9%	47	11%	13%	21%	55%	47	13
Stack Overflow	Regex	1%	8%	58%	32%	71	4%	11%	25%	59%	71	14
Stack Overflow	Spring	24%	32%	28%	16%	79	14%	18%	30%	38%	79	23
Stack Overflow	Util	4%	16%	50%	29%	68	6%	13%	28%	53%	68	19
Stack Overflow	**Total**	17%	29%	39%	16%	721	10%	13%	28%	49%	713	207
Kode Java	JDBC	2%	43%	34%	21%	56	4%	9%	25%	62%	55	10
Kode Java	Regex	5%	23%	42%	30%	60	3%	7%	24%	66%	58	13
Kode Java	Spring	8%	32%	42%	17%	59	5%	12%	22%	60%	58	10
Kode Java	Util	2%	15%	35%	48%	65	2%	9%	14%	75%	65	8
Kode Java	**Total**	4%	28%	38%	30%	240	3%	9%	21%	66%	236	41

We compared the results for each topic to the results for all other topics (e.g., Hibernate vs. non-Hibernate), and while task familiarity was not independent of the topic for all of these tests (Chi-Squared tests, $p < 10^{-8}$), the extent to which code fragments were rated as self-explanatory was always independent of the topic.[2] In the absence of strong evidence that whether a code fragment is self-explanatory depends on its topic, we pursued our inquiry with a qualitative analysis of potential factors. The next section presents our qualitative findings on what affects the extent to which a code fragment is self-explanatory.

B. Missing Information

Table V shows the results of the annotation of the 160 code fragments in terms of the issues that emerged from our qualitative coding (see Section II-C). For code fragments from Stack Overflow, incompleteness was the most common issue. In contrast, this issue never occurred in code fragments from the curated website Kode Java. On Stack Overflow, code fragments from all topic areas suffered from incompleteness as well as code quality issues, while missing rationale was relatively common in code fragments from both sources. With the exception of Jodatime code fragments having relatively fewer issues, there is no clear trend indicating that the issues exhibited by code fragments depend on the particular topic area.

[2]This comparison involved 32 Chi-Squared tests on 4×2 matrices (12 Stack Overflow topics plus 4 Kode Java topics, comparison of task familiarity and self-explanatory ratings). We applied a Bonferroni correction for multiple comparison to the all p-values.

IV. DISCUSSION

Our results provide evidence that some code fragments from non-curated sources such as Stack Overflow can indeed stand on their own as self-explanatory fragments, even without code comments or explanatory text. These code fragments are ready to be reused by developers in other contexts. However, this only applies to just under half of the Stack Overflow code fragments in our study. The remaining code fragments are not completely self-explanatory and require more context to understand.

Our survey participants mentioned several issues that prevent code fragments from being self-explanatory. Of those, we found that incompleteness is the most common for code fragments from the non-curated source Stack Overflow while incompleteness was not an issue at all for code fragments from the curated source Kode Java. Interestingly, missing rationale and explanations of domain concepts and/or terminology are the only issues that emerged from our study that could not be addressed by source code alone. Very few code fragments exhibited problems related to domain knowledge, suggesting that the one major piece of information that cannot be captured in source code is rationale. This observation opens up avenues for future work around providing developers with a structured way of expressing rationale.

V. RELATED WORK

In their study on what makes a good code example, Nasehi et al. found that the explanations accompanying examples are as important as the examples themselves [3]. In contrast to our methodology, their data collection was solely based on an analysis of code fragments, and they did not ask developers

TABLE V
ANNOTATION RESULTS

Source	Topic	Organization	Naming	Rationale	Incomplete	Quality	Domain	Clutter	Total
Stack Overflow	Hibernate	5		4	5	3	1	1	19
Stack Overflow	IO	2	1	4	3	2			12
Stack Overflow	Java2D			5	7	5		3	20
Stack Overflow	JavaMail	3	3		3	3	2	3	17
Stack Overflow	JAXB	1		1	7	3			12
Stack Overflow	JDBC	3		2	7	3		2	17
Stack Overflow	Jodatime		1	1	3	3		1	9
Stack Overflow	Logging	1	1	4	4	4			14
Stack Overflow	Reflection		1	5	4	3		2	15
Stack Overflow	Regex			7	4	3	1		15
Stack Overflow	Spring	3		3	5	5		2	18
Stack Overflow	Util	2	2	4	5	5		3	21
Stack Overflow	**Total**	20	9	40	57	42	4	17	189
Kode Java	JDBC			2				6	8
Kode Java	Regex			4			1		5
Kode Java	Spring							2	2
Kode Java	Util	1		1		1			3
Kode Java	**Total**	1	0	7	0	1	1	8	18

about their information needs. As a result, our work provides more specific information about the issues that negatively affect code fragment understandability. Barzilay et al. [6] made use of textual and social information available on Stack Overflow in a code search and recommendation tool called Example Overflow.

Thread titles on Stack Overflow have attracted the attention of the research community. For example, in their work on analyzing the topics that mobile developers are asking about, Rosen and Shihab [7] argue that "titles summarize and identify the main concepts being asked in the post" while "the body of the question adds non-relevant information". Tools such as Seahawk [8] or NLP2Code [9] determine the relevance of a Stack Overflow thread for a code fragment search at least in part based on the thread's title.

Although there has not been much work on the information needs of software developers around code fragments, there has been work on information needs of software developers in general. To name a few, Fritz and Murphy [10] provided a list of questions that focus on issues that occur within a project and Sillito et al. [11] provided a similar list focusing on questions during evolution tasks. LaToza and Myers [12] 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. [13] found that the most frequently sought information included awareness about artifacts and coworkers.

VI. Future Work

As next steps in this research project, we plan to investigate which of the identified code fragment issues have the most impact on how self-explanatory a code fragment is, to what extent the identified issues are resolved by the original text that accompanies each code fragment, and how to identify the most useful explanation in the text. Another promising avenue for future work is the automated classification of code fragments based on how self-explanatory they are. The ultimate goal of this work is understanding developers' information needs related to code fragments and how these needs can be addressed by documentation guidelines or on-demand developer documentation [14].

References

[1] "GitHub Gist Home Page," https://gist.github.com/, verified 20 June 2017.

[2] "MSDN Developer Code Samples," https://code.msdn.microsoft.com/, verified 20 June 2017.

[3] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example? A study of programming Q&A in StackOverflow," in *Proc. Int'l. Conf. on Software Maintenance*, 2012, pp. 25–34.

[4] L. B. L. de Souza, E. C. Campos, and M. d. A. Maia, "Ranking crowd knowledge to assist software development," in *Proc. Int'l. Conf. on Program Comprehension*, 2014, pp. 72–82.

[5] "Kode Java," http://kodejava.org/, verified 20 June 2017.

[6] O. Barzilay, C. Treude, and A. Zagalsky, *Facilitating Crowd Sourced Software Engineering via Stack Overflow*. Springer New York, 2013, pp. 289–308.

[7] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using Stack Overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[8] L. Ponzanelli, A. Bacchelli, and M. Lanza, "Seahawk: Stack Overflow in the IDE," in *Proc. Int'l. Conf. on Software Engineering*, 2013, pp. 1295–1298.

[9] B. A. Campbell and C. Treude, "NLP2Code: Code snippet content assist via natural language tasks," in *Proc. Int'l. Conf. on Software Maintenance and Evolution*, 2017, to appear.

[10] T. Fritz and G. C. Murphy, "Using information fragments to answer the questions developers ask," in *Proc. Int'l. Conf. on Software Engineering - Vol. 1*, 2010, pp. 175–184.

[11] J. Sillito, G. C. Murphy, and K. De Volder, "Questions programmers ask during software evolution tasks," in *Proc. Int'l. Symp. on Foundations of Software Engineering*, 2006, pp. 23–34.

[12] T. D. LaToza and B. A. Myers, "Hard-to-answer questions about code," in *Evaluation and Usability of Programming Languages and Tools*, 2010, pp. 8:1–8:6.

[13] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proc. Int'l. Conf. on Software Engineering*, 2007, pp. 344–353.

[14] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vásquez, G. Murphy, L. Moreno, D. Shepherd, and E. Wong, "On-demand developer documentation," in *Proc. Int'l. Conf. on Software Maintenance and Evolution*, 2017, to appear.