Code Quality Analysis of StackOverflow Code Examples for Reusability: An Exploratory Study

Mohammad Masudur Rahman   Chanchal K. Roy
Computer Science
University of Saskatchewan, Canada
{mor543, ckr353}@mail.usask.ca

Iman Keivanloo
Computer Science
Concordia University, Canada
i_keiv@encs.concordia.ca

ABSTRACT

StackOverflow, a popular programming Q & A site, often contains working code examples that solve particular programming problems, as a part of the answers against the posted questions. Study shows that the community is highly interested of those code examples, and they also contribute greatly to the promotion and demotion of the container answers. Given the developing interests, we are motivated to study whether the metrics-based code quality evaluation of those examples actually agrees with their subjective evaluation by StackOverflow, and this is an important piece of information for the developers willing to reuse those examples. In this paper, we conduct an exploratory study with 110 code examples against 55 programming questions, and develop a metrics-based quality model to determine that agreement. Our model agrees with StackOverflow for 74.54% code examples in relative quality prediction, which is interesting and revealing about the effectiveness of the evaluation of StackOverflow. While the preliminary findings are promising, they must be validated with larger dataset and more extensive experiments.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*maintainability measures, performance measures*

Keywords

Code quality, Readability, Security, Maintainability

1. INTRODUCTION

During development and maintenance of a software product, software developers deal with different programming problems or challenges, and they frequently look into different programming Q & A sites, forums, discussion boards and other sources for helpful information. A programming Q & A site discusses about different programming issues and their possible fixations in the form of questions and answers respectively posted by the community. The answers posted in the site often contain working code examples that solve particular programming problems or accomplish certain programming tasks. The developers find such examples reusable and also frequently apply them in their every day problem solving and learning activities. The posted code examples are generally viewed and evaluated by a large crowd of technical users from their subjective viewpoints, but their actual code level quality remains unknown to the developers during reuse. Therefore, the practice of reusing untested code examples from the publicly available sites helps them to reduce the workload and make the development or maintenance process faster in one hand, but exposes their software projects to the risk of having low quality code, and consequently leads to more maintenance overhead in the long run on the other hand.

StackOverflow[1], a popular social programming Q & A site, has a large community of 1.9 million technical users, and it discusses about 5.5 million programming questions from different programming domains and languages. In this site, users promote a question post or answer post through up-voting when they find it useful and informative, and down-vote a post if they find its content inappropriate, irrelevant or erroneous, and the difference between up votes and down votes is considered as the *score* for the post. Nasehi et al. [9] study the characteristics of the accepted answers to 163 different programming questions in StackOverflow, and argue that the accepted answers are very likely to contain efficient and concise code examples accompanied by comprehensive textual description. Their study also reveals that the discouraged (i.e., extensively down-voted) answers from StackOverflow either do not contain code examples or contain low quality code snippets. Treude et al. [15] study which type of questions are answered correctly for most of the time, and suggest that the answers of the code-review questions have the maximum acceptance rate of 92%.

Given that the code examples either in question posts or answer posts are of great interest to the community [9, 15], it is reasonable to conjecture that StackOverflow users often consult with them or more precisely reuse them in their everyday programming activities. Basically, this developing interest of the community on code examples provides us the motivation for their classification or reevaluation. As the study suggests, code examples arguably play a major role behind the *acceptance* or *rejection* of a programming answer by the community [9], they also can be categorized as *promoted*

ICSE '2013, Hyderabad India

[1] http://en.wikipedia.org/wiki/Stack_Overflow, Visited on Nov, 2013

(i.e., extracted from promoted answer) or *discouraged* (i.e., extracted from discouraged answer) ones analogously based on the *scores* of their corresponding posts. However, the vote score-based classification of the code examples suffers from several limitations. First, the votes can be considered as the quantification of the evaluation by the community users, which is based on their subjective viewpoints. The viewpoints may vary based on the expertise level, interests or demographics of the users, and therefore, the *vote score* may not reflect a representative measure for the quality of the code example. Second, evaluations (i.e., votes) both from expert and novice users are considered with equal importance which makes the vote score unreliable. Third, vote score of a code example (i.e., answer post) can be influenced by its age and exposure to the community. That means, a recently introduced good code example may have the vote score equal to that of a moderate quality example introduced long ago; however, the classification does not consider those factors and treats both code examples as of equal merit. Thus, it is pretty evident that score-based annotation of the code examples is affected by certain unattended factors, and it is of great importance to check the actual code level quality of those code examples given that millions of StackOverflow users are using them for their problem solving.

A number of existing studies focus on software code level metrics for checking readability [2, 10], reusability [14], effectiveness of refactoring [13] and comprehensive quality [3, 6, 8, 11, 12] of the software code. Taibi [14] studies the reusability of 33 open source projects based on code level metrics such as understandability, low complexity and modularity, and propose different weights (i.e., importance) to different metrics. Mäntylä and Lassenius [7] conduct an empirical study to determine correlation between subjective evaluation and metrics based evaluation on software quality (i.e., based on code smells), and suggest that no evaluation alone is completely reliable. However, as per our knowledge, no study focuses on the metrics-based code quality of the StackOverflow code examples. In our research, we are interested to check whether the perceived quality of the code examples based on code level metrics complies with the vote score-based classification. More specifically, we attempt to find out whether the promoted code example (Fig 1-(a)) is actually preferable to the discouraged one (Fig 1-(b)) for a programming problem in terms code quality metrics. In order to do that, we formulate the following research questions.

- RQ1: Is the code level quality of a discouraged code example worse than that of a promoted code example?
- RQ2: Does the metrics-based classification of the code examples totally agree with vote score-based classification by StackOverflow ? If not, why?

In our research, we conduct a limited exploratory study with 110 promoted and discouraged code examples against 55 programming problems from StackOverflow. We apply three code level metrics– readability, code soundness and coding rule violation, and one associated metric (e.g., author's expertise), and develop a code quality model to perceive the quality of those code examples. Our model agrees with StackOverflow for 74.54% code examples in relative quality prediction, which is interesting and revealing about the effectiveness of the evaluation of StackOverflow. We also investigate into the examples for which the model does not agree with StackOverflow on quality analysis, and attempt to answer the research questions. While our preliminary findings are promising, they must be validated with larger dataset and more extensive experiments to reach a reliable conclusion.

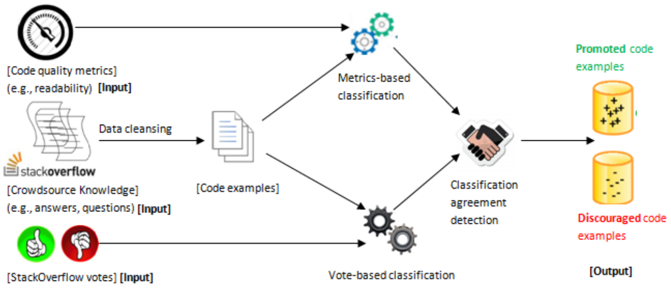


Figure 2: Schematic diagram of the proposed study

The rest of the paper is organized as follows– Section 2 focuses on our adopted methodology and code quality metrics, Section 3 discusses about the conducted experiments and findings, Section 4 describes the related works, and finally Section 5 concludes the paper with future works.

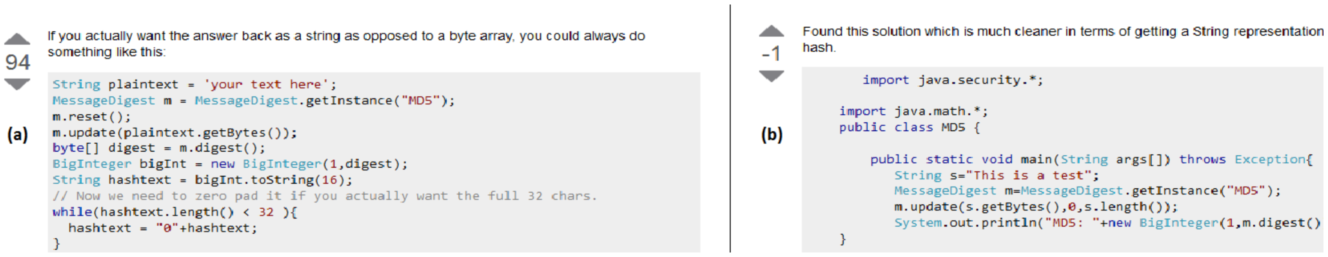
2. PROPOSED METHODOLOGY

Fig. 2 shows the schematic diagram of our exploratory study on code quality analysis of StackOverflow code examples. In this section, we discuss about the detailed technique of the study and the metrics we use to perceive the code quality of the examples.

2.1 Data Collection

Given the millions of questions in StackOverflow Q & A site from multiple programming languages and domains, we select a subset of 55 questions related to Java programming for the study. We impose certain constraints on the selection of a question–(1) The question has to be strictly related to a *particular programming problem*, and the ideal answer should contain a *precise and efficient code example* with comprehensive discussion, (2) Each question should have more than ten answers containing code examples to ensure that it is a *commonly discussed programming issue.* We collect the latest data dump[2] provided under creative commons, and extract the questions and answers. Out of 82,882 questions tagged as *Java*, we find 26,869 questions are strictly related to programming, and their answers contain code examples. We select a list of 75 programming questions from them, where each of the questions has more than ten answers that contain code examples. We extract the code examples from the raw HTML of StackOverflow answers against those questions, and manually analyze a random subset of the answers. We find that most of them are not directly compilable, and we perform simple tweaking (e.g., adding import statements or semicolons, declaring undeclared variables and so on) on the examples to make them compilable. However, we find some code examples are too trivial or they cannot be compiled at all with simple tweaking, which we discard from the list. Finally, we select 55 programming questions and the representative code examples posted in their answers for the study.

[2] http://blog.stackoverflow.com/category/cc-wiki-dump/



If you actually want the answer back as a string as opposed to a byte array, you could always do something like this:

```
String plaintext = 'your text here';
MessageDigest m = MessageDigest.getInstance("MD5");
m.reset();
m.update(plaintext.getBytes());
byte[] digest = m.digest();
BigInteger bigInt = new BigInteger(1,digest);
String hashtext = bigInt.toString(16);
// Now we need to zero pad it if you actually want the full 32 chars.
while(hashtext.length() < 32 ){
  hashtext = "0"+hashtext;
}
```

Found this solution which is much cleaner in terms of getting a String representation hash.

```
      import java.security.*;

    import java.math.*;
    public class MD5 {

        public static void main(String args[]) throws Exception{
           String s="This is a test";
           MessageDigest m=MessageDigest.getInstance("MD5");
           m.update(s.getBytes(),0,s.length());
           System.out.println("MD5: "+new BigInteger(1,m.digest()
    }
```

Figure 1: (a) Promoted code example, (b) Discouraged code example

2.2 Vote Score Based Classification

StackOverflow recognizes the technical merit and quality of a programming answer in terms of votes [9], and code examples are posted as a part of the answers. While the existing studies [9, 15] explain their undeniable role behind the promotion and demotion of the answers, the votes cast by the large crowd of technical users for those answers can also be considered to approximate the subjective quality of the contained code examples. Given that the subjective perception of quality may vary, we choose the code examples of two extreme perceptions– highly promoted and highly discouraged. We consider total score (i.e., difference between up votes and down votes) and age (i.e., difference between dumping date and post creation date) of the code examples, and calculate *vote score per day* for each of them. Then, based on the scores gained per day, we choose one highly promoted and one highly discouraged code examples for each of 55 questions. The idea is to determine whether the subjective evaluation of the code example agrees with the metrics-based evaluation. We also manually analyze each code example for possible false positives, and to perceive their relative quality difference.

2.3 Code Quality Metrics

StackOverflow code examples generally are of smaller sizes; in our case, we find them less than 15 lines in average. The code examples are posted either as complete methods or method body segments containing a few lines, and the complete class-structure are often unlikely. Therefore, most of the available code level quality metrics such as, object-oriented complexity metrics and so on, are not applicable for the code examples. Moreover, the motivation behind the quality evaluation is to determine how reusable and maintainable the code examples are for the developers. In this section, we discuss about four code related metrics used for the study.

2.3.1 Readability (R)

Readability of software code refers to a human judgement of how easy the code is to understand [2]. Alternatively, it also can be defined as the perceived barrier to understanding, which is essential to overcome before working with the code [10]. Reading (i.e., understanding) software code is the most time-consuming components of all maintenance activities [2], and therefore, readability is directly related to software maintainability. Knight and Myers [4] suggest that readability also greatly contributes to reusability and portability of the code. Thus, readability is an established code quality metric, and the baseline idea is– the more readable and understandable the code is, the easier it is to reuse and maintain in the long run. Buse and Weimer [2] propose a code readability model trained on human perception of readability or understandability. The model uses different textual source features (e.g., length of identifiers, number of comments, line length and so on) that are likely to affect the humans' perception of readability, and then the model predicts a readability score on the scale from zero to one, inclusive, with one describing that the code is highly readable. We use the readily available library[3] by Buse and Weimer [2] to calculate the readability metric of the code examples.

2.3.2 Author Rank (AR)

We think that the expertise of the author of a code example is likely to influence its quality [5]. StackOverflow provides incentives to the community users, who are actively contributing to the body of knowledge by asking important questions, posting helpful answers, adding informative comments and so on. One of those incentives is *Reputation*, which is an estimated quantification of the overall contributions of a user to StackOverflow, and it can be considered as the proxy of her expertise. The idea is that a code example posted by an experienced user is likely to contain less security and performance concerns compared to that posted by an inexperienced user. Le Goues and Weimer [5] use *author expertise* as an indicator of code quality for specification mining, and in our study, we also use it with a focus on reusability of the code examples. To determine the rough estimate of the author expertise, we consider the reputations of all users of StackOverflow, and provide a normalized estimate on the scale from zero to one for each user, where zero describes the least experience.

2.3.3 Code Soundness (CS)

Mäntylä and Lassenius [7] conduct a study on the analysis of agreement between subjective evaluation and metrics-based evaluation of software evolvability using code smells, and argue that neither technique alone is enough to detect all smells. Similarly, we can conjecture that code level metrics are not sufficient to discover all types of defects, inefficiencies or possible scopes for improvement in the code examples. StackOverflow facilitates to include the subjective evaluations for each code example in the form of comments which often contain invaluable and insightful analysis about its code quality. While one can argue that the comments are merely based on subjective viewpoints, we note that they also contain objective observations which can be considered to derive metrics describing the soundness of the code example. We leverage the objective observations to identify the strengths and weaknesses of the code example. Basically,

[3] http://www.arrestedcomputing.com/readability

we analyze all the comments about a code example against the code and count their numbers discussing about positive aspects (i.e., strength) and negative aspects (i.e., weakness) of the code. Then we normalize the *strength* and *weakness* measures using *maximum comment count* among the code examples of the *same question* as follows.

$$S_i = \frac{S_{i,count}}{max(TC_i)}, \quad W_i = \frac{W_{i,count}}{max(TC_i)} \tag{1}$$

Here, $S_{i,count}, W_{i,count}$ and TC_i denote the positive comments count, negative comments count and total comments count of a code example respectively. Both of *strength* and *weakness* provide a normalized score on the scale from zero to one, where zero represents the least measure of each metric. While we are not aware of if such metrics are used by any existing study, we hypothesize that they are important measures to determine the relative code quality of two code examples against the same question.

2.3.4 Rule Violation (RV)

Deviation from standard coding practices can be considered as an important measure to estimate the code quality of the code examples. Traditional metrics-based quality evaluation is dominated by code analysis tools, and they try to analyze the code against a certain set of recommended rules. Most of the tools concentrate on particular aspects of code. For example, *CheckStyle* focuses on conventions, *PMD* on bad practices and *FindBugs* focuses on potential bugs or threats. Thus, rules and standards of one tool may vary from another, and the tools are no way competitive rather than complementary. Given the facts about the tools, using any single one may not serve our purpose of detecting rule violations, and we use *sonarQube*[4] which combines the rules and standards of *PMD, FindBugs, CheckStyles* and so on. We collect three types of violations– critical, major and minor, in the code examples and determine *violations per source line* for each of them. Lochmann and Heinemann [6] propose a comprehensive quality model for a software project based on a set of rules associated with the attributes of software components and artifacts. However, given the coding structure and size of StackOverflow code examples, we hypothesize that *violation per source line* is an important and credible metric to estimate the relative quality of the code examples.

2.4 Metrics Based Evaluation

In our study, we focus on the quality analysis of StackOverflow code examples with the notion of their reusability. Given the developing interests of the community on the posted code examples [9, 15], the idea of investigating into their code quality using available metrics, is quite intriguing to us. We consider the readability, adherence to the best practices and identified issues or threats in the code examples to estimate their quality for reusability. One can argue that complexity can be a useful metric for this purpose. However, given the size and structure of the code examples, object-oriented complexity metrics (e.g., CK metrics) or cyclomatic complexity are not usefully applicable, and we believe that the discussed metrics (in Section 2.3) are the most contributing factors towards reusability of the StackOverflow examples. While other metrics contribute to the code quality positively, *weakness* and *rule violation* metrics perform negatively. We manually analyze 50 code examples and label them either as *promoted* or *discouraged.* Then we use those labeled code examples along with their computed metrics (i.e., Section 2.3), and use logistic regression to determine their relative predictive power (i.e., Odds ratios). Finally, we get the following quality model to perform the relative quality analysis among the code examples, where the coefficients are the predictive power (i.e., importance) of corresponding metrics.

$$\begin{aligned} Q_i = 3.0043 \times R_i + 0.0454 \times AR_i + 8.04884 \times S_i \\ +0.5078 \times W_i + 0.1547 \times RV_i \end{aligned} \tag{2}$$

In the model, we find *readability* and *strength* as the most dominating features, whereas *author rank* has the least influence in predicting code quality, which refutes our initial assumption about it.

3. RESULTS AND DISCUSSIONS

In our experiment, we use the quality model to estimate the quality measures of 110 code examples against 55 programming questions [1]. It should be noted that we use the estimates to perform the comparative quality analysis among the code examples of the same question. The idea is to determine whether a code example promoted by StackOverflow is actually of better code quality than the one which is discouraged by StackOverflow. Table 3 shows the results of our preliminary experiments., where the metric-based relative quality prediction agrees with that of StackOverflow at best for 41(74.54%) out of 55 questions. It also shows the results for different combination of metrics, and the empirical findings show that *readability* and *strength* of code examples are the most effective metrics for relative quality analysis. In the next section, we attempt to answer our research questions.

RQ1: Is the code level quality of a discouraged code example worse than that of a promoted code example? The preliminary results show that it is true for 74.54% code examples of the dataset. *Readability*, *strength* and *weakness* metrics are greatly related to easier comprehension, efficiency, security, maintainability and other attributes (i.e., quality) of the software code that stimulate its reuse [2], and we also found those metrics promising in our experiment for quality analysis. Basically, our model performs better when those metrics can be calculated properly from the available information, especially *strength* and *weakness* metrics which are derived from the conversations (i.e., comments) among the community users followed by the code example. Given the challenges of extracting metrics from text, they really help in code quality analysis if they can be extracted properly. So, to answer the first question, the quality of the discouraged code examples is inferior to that of promoted code examples for most of the time; however, this can be verified with the availability of essential and reliable information.

RQ2: Does the metrics-based classification of the code examples totally agree with vote score-based classification by StackOverflow ? If not, why? According to our experiment, the two classifications agree mostly, about 75%; however, the complete agreement may not be possible. We investigate into the 14 cases (28 code examples and their metrics) for which our quality model does not match with StackOverflow, and find a few issues or scenarios. First, most of them do not contain comments given that metrics from the com-

[4] http://www.sonarqube.org/

Table 1: Experimental Results

Metrics	APC[1]	A[2]	D[3]
{R}	25(55)	45.45%	54.55%
{R, S}	38(55)	69.09%	30.90%
{R, S, W}	41(55)	74.54%	25.45%
{R, S, W, RV}	40(55)	72.72%	27.27%
{R, S, W, AR}	41(55)	74.54%	25.45%
{R, S, W, RV, AR}	41(55)	**74.54%**	**25.45%**

[1] No. of example pairs for which quality evaluation matches with that of StackOverflow

[2] % of agreement

[3] % of disagreement

ments play major parts in our model, and the model does not perform well for those cases. Second, our model does not use any threshold to describe a code example either as promoted or discouraged, rather it uses relative quality analysis which may not effective all the time. For example, if there is little difference in the quality estimate of two code examples, the model still identifies the promoted and discouraged one; however, actually both of them should be considered either as promoted or discouraged. Third, StackOverflow contains some code examples, which are highly simplified with little technical merit, and are often intended for student homeworks and preliminary learning, and they are also highly voted. Our model does not perform well in that case. Fourth, we also find some cases where *vote score* does not necessarily represent the evaluation of the contained code examples or they do not play major part in answering the question.

4. RELATED WORKS

Existing studies focus on software code metrics for checking readability [2, 10], reusability [14], effectiveness of refactoring [13] and comprehensive quality [3, 6, 8, 11, 12] of the software code. Our work is closely related to the comparative analysis between subjective and metrics based evaluation of software evolvability by Mäntylä and Lassenius [7] as we have a similar goal with different methodology and subject system. Two existing studies [9, 15] explain the role of code examples in StackOverflow question or answer post; however, to our knowledge, this is the first attempt to analyze the subjective and metrics based code quality of the StackOverflow code examples, and to determine their agreement.

5. CONCLUSION & FUTURE WORKS

To summarize, given the developing interest of StackOverflow community to the code examples, we attempt to determine whether their subjective evaluation by the community agrees with the metrics-based evaluation. We conduct an exploratory study with 110 representative code examples against 55 programming questions, and develop a code quality analysis model with the available and applicable metrics. The model agrees with StackOverflow in relative quality prediction for 74.54% code examples, which is quite promising and revealing about the effectiveness of StackOverflow votes. However, the finding must be extensively validated with more code examples from Java programming and other languages or domains to the answer the research questions properly.

References

[1] Stacksuggest experimental data. URL `homepage.usask.ca/~masud.rahman/ss/expdata`.

[2] R.P.L. Buse and W.R. Weimer. Learning a metric for code readability. *Softw. Eng., IEEE Trans.*, 36(4):546–558, 2010.

[3] Mandeep K. Chawla and Indu Chhabra. Implementing source code metrics for software quality analysis. *IJERT*, 1(5), July 2012.

[4] John C. Knight and E. Ann Myers. Phased inspections and their implementation. *SIGSOFT Softw. Eng. Notes*, 16(3):29–35, July 1991.

[5] C. Le Goues and W. Weimer. Measuring code quality to improve specification mining. *Softw. Eng., IEEE Trans.*, 38(1):175–190, 2012.

[6] Klaus Lochmann and Lars Heinemann. Integrating quality models and static analysis for comprehensive quality assessment. In *Proc. of WETSoM*, pages 5–11, 2011.

[7] Mika V. Mäntylä and Casper Lassenius. Subjective evaluation of software evolvability using code smells: An empirical study. *Empirical Softw. Engg.*, 11(3):395–431, September 2006.

[8] P. Meirelles, C. Santos, J. Miranda, F. Kon, A. Terceiro, and C. Chavez. A study of the relationships between source code metrics and attractiveness in free software projects. In *Proc. SBES*, pages 11–20, 2010.

[9] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example -a study of programming q and a in stackoverflow. In *Proc. ICSM*, pages 25 –35, 2012.

[10] Daryl Posnett, Abram Hindle, and Premkumar Devanbu. A simpler model of software readability. In *Proc. MSR*, pages 73–82, 2011. ISBN 978-1-4503-0574-7.

[11] Mrinal Singh. Rawat, Arpita Mittal, and Sanjay Kumar Dubey. Survey on impact of software metrics on software quality. *IJACSA*, 3(1), 2012.

[12] Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou, and Georgios L. Bleris. Code quality analysis in open source software development. *Info Systems J*, 12:43–60, 2002.

[13] K. Stroggylos and D. Spinellis. Refactoring–does it improve software quality? In *In Proc. WoSQ*, pages 10–, 2007.

[14] Fathi Taibi. Reusability of open-source program code: a conceptual model and empirical investigation. *SIGSOFT Softw. Eng. Notes*, 38(4), July 2013.

[15] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *ICSE*, pages 804–807, 2011.