# Reusability Analysis of Crowdsource Code Examples

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

Iman Keivanloo
Queen's University, Canada
iman.keivanloo@queensu.ca

## ABSTRACT

In StackOverflow, code examples are generally analyzed and subjectively evaluated by a large crowd of technical users. Given the growing interest of the examples to the community and their undeniable role in answers, we are motivated to study whether the metric-based code quality evaluation of those code examples actually agrees with their subjective evaluation by StackOverflow. 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 from 55 programming questions, and develop a metric-based quality model to examine the agreement level. Our model agrees with StackOverflow for 78.18% code examples in relative quality prediction, which is interesting, and it reveals the effectiveness of the subjective evaluation by StackOverflow. While the preliminary findings are promising, they must be validated with larger dataset.

## Categories and Subject Descriptors

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

## Keywords

Code quality, readability, reusability, maintainability

## 1. INTRODUCTION

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 upvoting when they find it useful and informative, and downvote a post if they find its content ~~irrelevant~~ or erroneous, and the difference between up votes and down votes is considered as the *score* for the post.

During development and maintenance of a software product, software developers deal with different programming problems or challenges, and they frequently look into different programming issues posted on StackOverflow. The answers posted on the site often contain working code examples (i.e., code fragments) 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; however, the objective quality of the examples is not taken into consideration by the developers during reuse. Reusing such code examples can be a threat if the encouraged (i.e., promoted) examples by the crowd maintain low quality from code metrics.

Nasehi et al. [5] study the characteristics of the accepted answers of 163 different programming questions from 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 downvoted) answers from StackOverflow either do not contain code examples or miss the explanation about the code. Treude et al. [8] study which type of questions are answered correctly for most of the time, and suggest that the answers of the *code-review questions* (i.e., containing code examples) have the maximum acceptance rate of 92%. However, the quality of the code examples is not studied from metric-based point of view.

A number of existing studies focus on software code level metrics for checking readability [1], reusability [7], and comprehensive quality [3, 6] of the software code. Taibi [7] studies the reusability of 33 open source projects based on code level metrics such as understandability, low complexity and modularity, and proposes different weights (i.e., importance) to different metrics. Mäntylä and Lassenius [4] conduct an empirical study to determine correlation between subjective evaluation and metric-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 metric-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

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


*WeTSOM* '2014, Hyderabad India

94

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

(a)

```
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;
}
```

-1

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

(b)

```
    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**

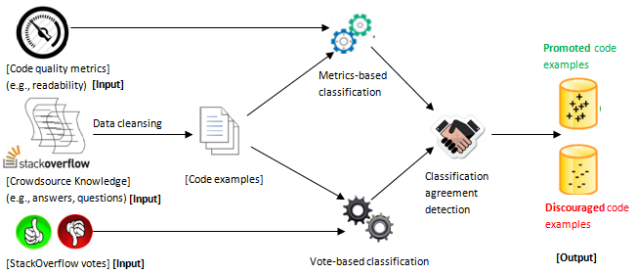


**Figure 2: Schematic diagram of the proposed study**

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 of 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: Why doesn't the metric-based classification of the code examples completely agree with vote score-based classification by StackOverflow?

In our research, we conduct an exploratory study with 110 highly promoted and highly discouraged code examples (e.g., programming solutions) against 55 programming problems from StackOverflow. We apply four code level metrics – *readability*, *strength*, *weakness* 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 78.18% code examples in *relative quality prediction*, which is interesting, and it reveals the effectiveness of the subjective evaluation by 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.

## 2. PROPOSED METHODOLOGY

Fig. 2 shows our approach for code quality analysis of StackOverflow code examples. In this section, we discuss about the detailed technique of the study, ~~and~~ the metrics and the model we use to perceive the code quality of the examples.

### 2.1 Data Collection

We collect the latest data dump[2] provided under creative commons, and extract 75 programming questions and their corresponding answers from it. It should be noted that each of those questions has *more than ten answers that contain code examples*. We extract the code examples from the raw HTML of StackOverflow answers, and manually analyze those examples. 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 also 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 110 promoted and discouraged code examples posted in their answers for the study. It should be noted that we select two code examples for each question, and one of them is highly promoted (e.g., up-voted) and the other is highly discouraged (e.g., down-voted).

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

### 2.2 Vote Score Based Classification

StackOverflow recognizes the technical merit and quality of a programming answer in terms of votes [5], and code examples are posted as a part of the answers. While the existing studies [5, 8] explain the undeniable role of the code examples 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 *a highly promoted* (e.g., highest scores gained per day) and *a highly discouraged* (e.g., lowest scores gained per day) code examples for each of 55 questions.

### 2.3 Code Quality Metrics

StackOverflow code examples are generally posted either as complete methods or method body segments containing a few lines, and the complete class-structure is often unlikely. Therefore, most of the available code level quality metrics such as object-oriented complexity metrics are not applicable for the code examples. In this section, we discuss five code related metrics used for the study.

**Readability (R)**: Readability of software code refers to a human judgement of how easy the code is to understand [1]. Reading (i.e., understanding) software code is the most time-consuming components of all maintenance activities, and

therefore, readability is directly related to software maintainability. Study suggests that readability also greatly contributes to reusability and portability of the code [1]. 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 [1] 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 [1] to calculate the readability metric of the code examples.

**Author Rank (AR)**: We believe that the expertise of the author of a code example is likely to influence its quality [2]. 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. Le Goues and Weimer [2] 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 a rough estimate for the *author expertise* metric of a code example, we consider the *Reputation* of the corresponding author and the *Maximum Reputation* among all authors of StackOverflow. Then we provide a normalized estimate on the scale from zero to one, where zero describes the least experience.

**Code Soundness (S, W)**: Mäntylä and Lassenius [4] conduct a study on the analysis of agreement between subjective evaluation and metric-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 level 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, 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* metrics provide a normalized score on the scale from zero to one, where zero represents the least measure of each metric.

**Rule Violation (RV)**: Traditional metric-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 thus 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 [3] propose a rule-based quality model for the comprehensive quality assessment of a complete software project using the rules extracted from static code analysis tools. 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. In order to preserve uniformity with other metrics, we normalize *violation per source line* for each code example.

## 2.4 Metric-Based Quality Model

We consider readability, author's expertise, adherence to the best coding practices, identified issues and threats in the code examples to estimate the quality of the examples with the focus on their reusability. We randomly select 50 code examples from 25 programming questions in the dataset, analyze their quality, and manually label them either as *promoted* or *discouraged*. Then we use those labeled examples along with their computed metrics (i.e., Section 2.3), and use logistic regression-based classifier from *Weka* ~~with ten-fold cross-validation~~ to determine the relative predictive power of the proposed metrics. It should be noted that we use Odd Ratio[2] of each metric, which is a logarithmic transformation of the metric coefficient in the regression equation of the classifier, and tune them under controlled iterations to determine the predictive power (i.e., importance) of the metric. Since, we are interested in determining the relative quality (i.e., without a threshold) of two code examples of the same question, we ignore the intercept of the equation, and develop the following quality model.

$$\begin{aligned} Q_i = 3.0043 \times R_i + 4.3067 \times AR_i + 8.04884 \times S_i \\ +0.5078 \times W_i + 0.5812 \times RV_i \end{aligned} \tag{2}$$

In the model, we find *readability*, *author rank*, and *strength* as the most dominating features (i.e., metrics), whereas *weakness* and *rule violation* have the least influence in predicting code quality.

# 3. RESULTS AND DISCUSSIONS

In our experiment, we use the proposed quality model to estimate the quality of 110 code examples against 55

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

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

**Table 1: Experimental Results**

| Metrics | APC[1] | A[2] | D[3] |
|---|---|---|---|
| {R} | 25(55) | 45.45% | 54.55% |
| {AR} | 32(55) | 58.18% | 41.82% |
| {S} | 30(55) | 54.55% | 45.45% |
| {W} | 22(55) | 40.00% | 60.00% |
| {RV} | 24(55) | 43.64% | 56.36% |
| {R, AR} | 30(55) | 54.55% | 45.45% |
| {R, S} | 39(55) | 70.91% | 29.09% |
| {R, W} | 29(55) | 52.72% | 47.28% |
| {R, AR, S} | **42(55)** | **76.36%** | **23.64%** |
| {R, S, W} | 41(55) | 74.54% | 25.45% |
| {R, AR, S, W} | **43(55)** | **78.18%** | **21.82%** |
| {R, S, W, RV} | 41(55) | 74.54% | 25.45% |
| {R, AR, S, W, RV} | **43(55)** | **78.18%** | **21.82%** |

[1] No. of example pairs for which relative quality evaluation matches with that of StackOverflow
[2] % of agreement
[3] % of disagreement

programming questions (dataset can be found online[5]). It should be noted that we use the quality estimates to perform the comparative analysis among the two 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, from metric-based point of view. Table 1 shows the results of our preliminary experiments using Equation (2), where the metric-based relative quality of the code examples agrees with that of StackOverflow at best for 43 (78.18%) out of 55 questions. It also shows how different component quality metrics can influence the estimated overall quality of the code examples, and the empirical findings show that *readability*(R), *author rank*(AR) and *strength*(S) are the most effective metrics for relative quality analysis when they are considered in combination. We note that *weakness* and *rule violation* metrics have a little or no influence to the proposed model, which refutes our initial assumption.

*RQ1: Is the code level quality of a discouraged code example worse than that of a promoted code example?* Our preliminary results (Table 1) indicate that the code quality of discouraged code examples is worse than that of promoted code examples in 78% of the examples. Based on the subjective evaluation (e.g., score per day) by StackOverflow, we estimated the relative quality of the two code examples against each of our selected questions, and determined the *promoted* and the *discouraged* ones. In order to determine their code level quality, we collected the target quality metrics (Section 2.3) of each code example, and applied them to the proposed quality model. The model provides an estimate about the quality of each example, and we used those estimates to determine the relative code quality of the two code examples against a question. Then we compared the metric-based relative quality against the corresponding relative quality obtained from subjective evaluation. The ~~experimental~~ result shows that the discouraged code examples are of inferior quality in terms of code metrics to the promoted ones for 43(78.18%) test cases out of 55 cases.

*RQ2: Why doesn't the metric-based classification of the code examples completely agree with vote score-based classification by StackOverflow?* According to our experiment, the two classifications agree mostly, about 78%; however, the complete agreement may not be possible. We investigated ~~into~~ the 12 cases (24 code examples and their metrics) for which our quality model does not match with StackOverflow, and found a few issues or scenarios. First, most of them do not contain comments given that metrics (e.g., strength and weakness) from the comments play major parts in our model, and the model does not perform well for those cases (e.g., 9 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 be effective all the time. For example, if there is a little difference in the quality estimate of two code examples, the model still determines the promoted and discouraged ones; however, both of them should be considered either as promoted or discouraged in practical. We found one such case in the experiment. Third, StackOverflow contains some code examples, which are highly simplified with little technical merit, and are often intended for preliminary learning, and they are also highly voted. Our model does not perform well in that case (e.g., 3 cases).

Given that *code quality* of the software code is a multi-faceted term [6], we focus on the quality analysis to determine the *reusability* of a code example. *Readability*, *strength*, *weakness* and *rule violation* metrics are greatly related to comprehensibility, efficiency, security, maintainability and other attributes (i.e., quality) of the software code that stimulate its reuse [1]. In our experiment, we found *strength*(e.g., weight 8.05) and *readability* (e.g., weight 3.00) are the most predictive metrics while combined for quality analysis. On the other hand, *weakness* and *rule violation* metrics are found not predictive. We can speculate that *rule violation* metric may not be properly applicable for StackOverflow codes due to their fragmented nature, and *weakness* metric may need to be refined for effective use; however, we need to experiment with more data to reach a conclusion.

[5] www.usask.ca/∼masud.rahman/ss/expdata

## 4. CONCLUSION & FUTURE WORKS

To summarize, given the growing interest of StackOverflow community to the code examples, we attempt to determine whether their subjective evaluation by the community agrees with the metric-based evaluation. We conduct an exploratory study with 110 representative code examples against 55 programming questions, and found that the subjective evaluation agrees with the metric-based evaluation for 78.18% code examples. The finding is quite promising, and it reveals the effectiveness of StackOverflow votes. It also has the potential to encourage more research in the quality analysis of the code examples often found in the programming Q & A sites, and our developed quality model can assist the developers in reusing code examples with informed knowledge of metric-based quality. However, the finding and the model must be validated with larger dataset to reach a reliable level.

## References


[1] R.P.L. Buse and W.R. Weimer. Learning a Metric for Code Readability. *TSE*, 36(4):546–558, 2010.

[2] C. Le Goues and W. Weimer. Measuring Code Quality to Improve Specification Mining. *TSE*, 38(1):175–190, 2012.

[3] K. Lochmann and L. Heinemann. Integrating Quality Models and Static Analysis for Comprehensive Quality Assessment. In *Proc. of WETSoM*, pages 5–11, 2011.

[4] M.V. Mäntylä and C. Lassenius. Subjective Evaluation of Software Evolvability using Code Smells: An Empirical Study. *ESE*, 11(3):395–431, 2006.

[5] 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.
[6] M.S. Rawat, A. Mittal, and S.K. Dubey. Survey on Impact of Software Metrics on Software Quality. *IJACSA*, 3(1), 2012.
[7] F. Taibi. Reusability of Open-Source Program Code: A Conceptual Model and Empirical Investigation. *SE Notes*, 38(4), 2013.
[8] C. Treude, O. Barzilay, and M.A. Storey. How Do Programmers Ask and Answer Questions on the Web? (NIER Track). In *ICSE*, pages 804–807, 2011.