Impact of stack overflow code snippets on software cohesion: a preliminary study

Preprint · March 2019

DOI: 10.13140/RG.2.2.14791.75688

2 authors:

Mashal Ahmad
University College Dublin
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Mel Ó Cinnéide
University College Dublin
69 PUBLICATIONS **1,627** CITATIONS
SEE PROFILE

Impact of stack overflow code snippets on software cohesion: a preliminary study

Mashal Ahmad
Lero, School of Computer Science
University College Dublin
Dublin, Ireland.
mashal.ahmad@ucdconnect.ie

Mel Ó Cinnéide
Lero, School of Computer Science
University College Dublin
Dublin, Ireland
mel.ocinneide@ucd.ie

***Abstract*—Developers frequently copy code snippets from publicly-available resources such as Stack Overflow (SO). While this may lead to a 'quick fix' for a development problem, little is known about how these copied code snippets affect the code quality of the recipient application, or how the quality of the recipient classes subsequently evolves over the time of the project. This has an impact on whether such code copying should be encouraged, and how classes that receive such code snippets should be monitored during evolution. To investigate this issue, we used instances from the SOTorrent database where Java snippets had been copied from Stack Overflow into GitHub projects. In each case, we measured the quality of the recipient class just prior to the addition of the snippet, immediately after the addition of the snippet, and at a later stage in the project. Our goal was to determine if the addition of the snippet caused quality to improve or deteriorate, and what the long-term implications were for the quality of the recipient class. Code quality was measured using the cohesion metrics Low-level Similarity-based Class Cohesion (LSCC) and Class Cohesion (CC). Over a random sample of 378 classes that received code snippets copied from Stack Overflow to GitHub, we found that in almost 70% of the cases where the copied snippet affected cohesion, the effect was to reduce the cohesion of the recipient class. Furthermore, this deterioration in cohesion tends to persist in the subsequent evolution of recipient class. In over 70% of cases the recipient class never fully regained the cohesion it lost in receiving the snippet. These results suggest that when copying code snippets from external repositories, more attention should be paid to integrating the code with the recipient class.**

***Index Terms*—Quality, Cohesion, Metrics, Evolution**

I. Introduction

Copying code snippets from publicly-available resources such as Stack Overflow is commonplace in software development. This practice can have adverse effects such as license violation, introduction of vulnerabilities and bug propagation, and can also degrade the quality of project code [1]. Developers also modify code snippets copied from Stack Overflow (SO) so these snippets evolve over time. Our research focusses on the impact that copied code can have on the code quality of the recipient class. This research also aims to identify the impact on quality evolution of recipient classes. Monitoring quality evolution can help developers to prioritise their efforts for long term maintenance activities.

In this paper, our goal is to investigate whether the copied snippet contributes to code quality, or disimproves it. In order to measure the code quality, we used the class cohesion metrics LSCCC and CC. Our study involves a sample of 378 GitHub (GH) classes with code snippets copied from Stack Overflow.

The rest of the paper is organised as follows. Section II comprises a literature review followed by a presentation of our research methodology in Section III. In Section IV we present our results while Section V we discuss these results and consider some threats to validity. Finally, our conclusions are summarised Section VI.

II. Literature Review

Various research studies have been carried out on different aspects of Stack Overflow, examining the quality of code available on SO and GitHub repositories from different perspectives. An insight evolution analysis of SO posts was carried out by Baltes et al. [2]. They found that posts start evolving soon after their creation as authors of posts start editing the posts. Posts with more edits attract more comments and hence get more attention from developers. This follows on from their earlier study [3] where they found that two-thirds of the clones from the ten most commonly copied Java code snippets on Stack Overflow do not contain any reference the original SO source. In related work, Kavaler et al. [4] used view counts and user scores to assess the quality of SO answers related to new Android APIs, finding that questions posted with incentives and APIs with proper documentation receive higher quality answers.

Ragkhitwetsagul et al. [1] performed a study of online code clones on Stack Overflow and discussed their toxicity through two developer surveys and a large-scale detection of code clones. They found that almost 66% of code snippets of Stack overflow are *toxic*, i.e. involve deprecated code or contain bugs. Our work is different in that we do not analyse the quality of snippet itself, but the impact of the copied snippet on the quality of recipient class.

Abdalkareem et al. [5] analysed if copied code from SO posts has adverse effects on the quality of applications. Using F-Droid repositories, they identified clones between Stack Overflow posts and Android apps. The behaviour of copying the code was detected by the use of timestamps. Every one of the 22 apps studied was found to contain cloned code, and this code cloning is practiced generally for enhancing the existing

code. For assessing the impact of quality on mobile apps they classify the commits before and after the code reuse. If the bug ratio of bug fixing commits (based on commit messages) increases after the addition of a snippet from Stack Overflow, then the quality of that mobile app is considered low. Our approach does not rely on commit messages for assessing the quality, rather we employed cohesion metrics that do not require human interpretation.

Zhang et al. [6] investigated the quality of SO code snippets by examining the misuse of API calls. They found that approximately 31% of the analysed SO posts possibly incorporate API usage contraventions that could result in program failure and resource leaks.

In summary, while the phenomenon of copying code from Stack Overflow has received much attention from researchers, little work has been done on the impact such copying has on code quality, and this is what our research focusses on.

III. RESEARCH METHODOLOGY

In this section we briefly discuss the data set, the data preparation process, the cohesion metrics employed and the research questions that we investigate in this study.

A. SOTorrent Database

For this experiment we used the SOTorrent dataset [7], which provides the version history of individual posts and links from SO posts to other datasets like GitHub. The table PostReferenceGH of the SOTorrent database contains a total 6,111,157 such records where SO posts are directly linked to GitHub files. The records include various programming languages such as Python, R and Java; in our work we elected to use Java due to its popularity as a general-purpose programming language.

B. Data Preparation

To study the effect of a code snippet that has been copied to a GitHub, two versions are of particular interest: the version just prior to the addition of the snippet, which we term the *pre-SO snapshot* and the one just after the addition of the snippet, which we term the *post-SO snapshot*. The history of a class on GitHub starts from the commit where the class is created and ends with the latest commit in which the class still exists. This history of commits is used to study how the cohesion of the recipient class evolves. We chose 378 samples from the PostReferenceGH table. The choice of sample data was random, except that we ignored examples where the SO snippet was added on the first commit because in this case we have no pre-SO snapshot, and also ignore classes with fewer than three commits because such classes do not have the rich history required to study their evolution.

C. Cohesion Metrics

An important quality attribute in the object-oriented paradigm is class cohesion. Many metrics have been developed for assessing the cohesion of a class. We chose two of the most popular cohesion metrics, namely Class Cohesion (CC) [8] and Low-level Similarity-based Class Cohesion (LSCC) [9].
The formula for LSCC is as follows:

$$LSCC_{\mathrm{c}} = \begin{cases} 0 & if l = 0 \ and \ k > l, \\ 1 & if (l > 0 \ and \ k = 0) \ or \ k = 1, \\ \sum_{i=1}^{l} x^i(x^i - 1)/lk(k-1) & otherwise. \end{cases} \tag{1}$$

In the above formula *k* and *l* indicate the number of methods and attributes of a class respectively. The LSCC value is minimum for a class with many methods but no attributes, as methods are unrelated in this case. LSCC is highly cohesive for a class with many attributes but just a single method, or no methods at all. The LSCC value between two methods is assessed with a so-called *Method Attribute Reference* (MAR) which is the ratio of summation of common attributes, assessed directly or indirectly by a method, to the total number of methods and attributes.

The formula for CC is as follows:

$$CC_{\mathrm{c}} = 2 \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{|I_i \cap I_j|}{|I_i \cup I_j|} / k(k-1) \tag{2}$$

In Equation 2, I_i represents the set of attributes referenced by method *i*, while *k* represents the total number of methods in a class. The cohesion between two methods is measured as the similarity ratio between common attributes over the total referenced attributes. This ratio is computed for all possible combinations of methods in a class.
Both LSCC and CC were employed to measure cohesion for both the pre- and post- SO snapshots of a class.

D. Research Questions

We performed an experiment to answer the following research questions.
RQ1: Quality difference between snapshots of SO recipient classes: Is there any difference in cohesion between the pre-SO and post-SO snapshots of the recipient classes?
RQ2: Quality evolution of SO recipient classes: Do recipient classes remain of lower cohesion, or do they evolve to regain the cohesion of their pre-SO snapshot?

IV. RESULTS

A. RQ1: Quality difference between snapshots of SO recipient classes

The range of LSCC and CC is from 0 to 1, where 1 represents perfect cohesion. We observe the change in values of LSCC and CC for the pre- and post-SO snapshots of recipient classes. We consider two broad cases for the change in the metric values:

- Varying
- Constant

The first case, *varying*, is one which causes a change in value of the metric under consideration for the post-SO snapshot of the recipient class. This variation can be positive or negative, indicating an improvement or deterioration in cohesion respectively. In the case of positive variation, the value of the metric

is greater in the post-SO snapshot than in the pre-SO snapshot, while in negative case it is less in the post-SO snapshot than in the pre-SO snapshot.

In the *constant* case the value of the metric does not change between the pre-SO snapshot and the post-SO snapshots. The *varying* case is further divided into the following categories a) *deteriorating* and b) *improving*. When the metric value changes and the value decreases, it falls into the *deteriorating* category. On the other hand, if the changing value is increased, it is categorised as *improving*.

Figure 1 depicts the results of this investigation. From the sample set of 378 cases, the Low-level Similarity-based Class Cohesion metric (LSCC) remains constant in 39% of cases and varies in the remaining 61% of cases. Out of the 229 cases that vary, 69% show deterioration in cohesion, as against the 31% that show improvement in LSCC.

As can been seen in Figure 1, the results for the Class Cohesion metric (CC) are similar. It should be noted though that in spite of this similarity, there is a significant degree of disagreement between the metrics on the impact of the added snippet. For example, where LSCC disimproves CC also disimproves in 70% of cases; in the other 30% of cases it improves or remains constant.

Our first experimental result shows that, in cases where the copied code snippet does affect the metrics, the effect is negative in almost 70% of cases. This result suggests that developers should pay attention to quality issues when using copied snippets from external sources.

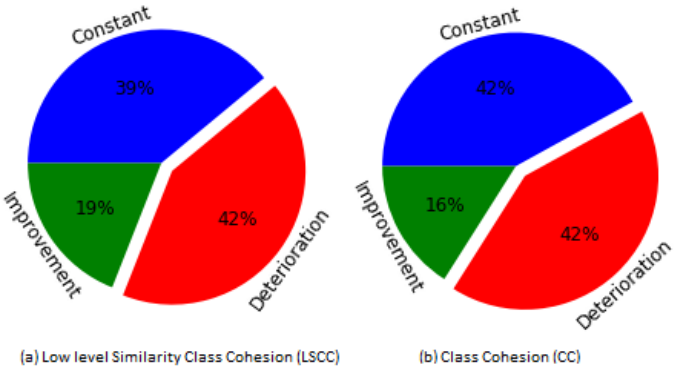

Fig. 1. Impact of Stack Overflow snippet on cohesion metrics

B. RQ2: Quality evolution of SO recipient classes

Following on from RQ1, it is natural to ask if the dip in cohesion caused by the copied code snippet is temporary or of a more enduring nature. To investigate this further, we examine the 42% of cases where the copied code snippet caused a deterioration in cohesion. In each case, we examine the sequence of commits that occur *after* the post-SO snapshot to determine if the metric value for the recipient class ever regains the value it had in the pre-SO snapshot.

We categorised the results into three groups as follows: i) Fully recovered ii) Not recovered and iii) Partially recovered. In the first category, the value of metric reaches or exceeds its value in the pre-SO snapshot. In second category, the metric never recovers to its pre-SO snapshot. In the third category, the metric reaches half of value it had in the pre-SO snapshot.

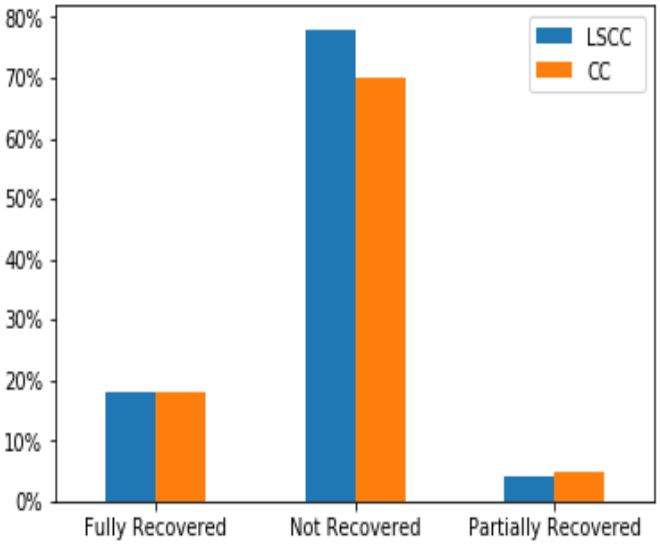

Fig. 2. Trends of Cohesion Metrics vs. percentage

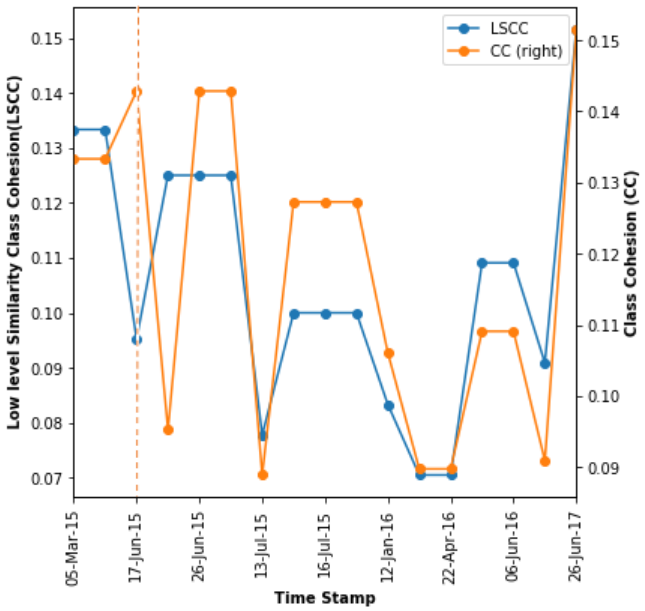

Fig. 3. Route.java cohesion evolution trend.

The results for the LSCC metric showed that 78% of the recipient classes failed to recover their pre-SO snapshot cohesion, 18% fully recovered this cohesion and 4% only partially recovered their pre-SO snapshot cohesion. The results

for the CC metric were similar with 70% not recovered, 25% fully recovered and 5% partially recovered. The percentage of classes failing to recover their cohesion is striking: 70% or over for both metrics.

To provide some further insight into what these evolutions look like, Figures 3, 4, and 5 are randomly chosen examples that represent the three categories described above namely partially recovered, not recovered and fully recovered respectively. We computed the LSCC and CC values for the history of commits and plotted it against the date the commit was performed.

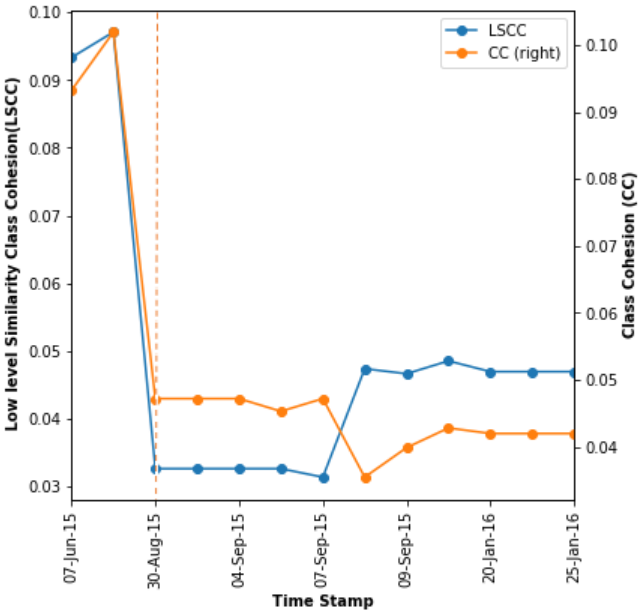

Fig. 4. Ble.java cohesion evolution trend.

V. DISCUSSION AND THREATS TO VALIDITY

Our results show that when a code snippet is copied from Stack Overflow, the cohesion of the recipient class deteriorates in a significant number of cases (42%). Furthermore, in the majority of cases (78%) the recipient class never recovers the cohesion it had prior to the addition of the snippet. This suggests that developers do not try to integrate the copied snippet with the recipient class in any meaningful way. We speculate that this may be due to time pressure, a fear of inducing bugs in a copied snippet that the developer does not fully understand, or a sense that the copied snippet is "correct" and should not be changed further. These reasons could also explain why the snippet is often not integrated any better in subsequent commits. While it is clear that copied snippets are causing quality issues in the classes that receive them, further work is needed to determine the causes and what steps should be taken to ameliorate this problem.

Now we consider briefly some threats to the validity of this study. Regarding *internal validity*, we have assumed that the change in metric values is related to the copied snippet, but the evolution graphs of LSCC and CC (Figures 3, 4, and 5)

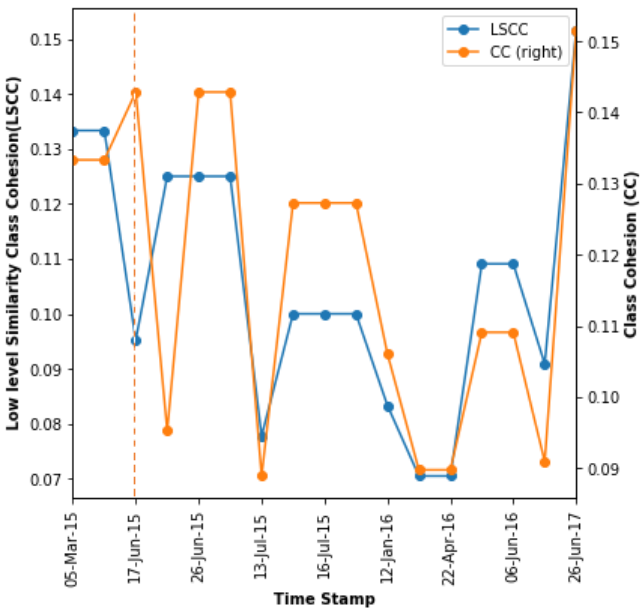

Fig. 5. SkeletonFragment.java quality evolution trend

show that the metric values fluctuate considerably and clearly the copied code is not the only factor contributing to the deterioration in the cohesion metrics. We reduced this effect by using a large number of randomly chosen examples. Regarding *construct validity*, we have implicitly assumed that the chosen cohesion metrics somehow measure the enigmatic notion of software quality, though is open to question [10]. Regarding *external validity*, we used open source Java projects on GitHub for our study and cannot claim that our results generalise outside this realm.

VI. CONCLUSIONS

The paper presented a study of the effect code snippets copied from Stack Overflow have on the cohesion of GitHub Java classes. Over a random sample of 378 classes that received code snippets copied from Stack Overflow to GitHub, we found that in almost 70% of cases where the copied snippet affected code quality, the effect was to reduce the quality of the recipient class. Furthermore, this deterioration in quality tends to persist in the subsequent evolution of recipient class. In over 70% of cases the recipient class never fully regains the quality it lost in receiving the snippet. We suggested some reasons why this drop in quality occurs, and why it tends to persist. These results indicate that a developer should take more care to integrate a copied code snippet with its recipient class. Future work involves extending this study to consider other cohesion and coupling metrics, applying it to a larger body of examples, and studying in more depth how the evolution of a class that receives a code snippet differs from the evolution of a non-recipient class.

Acknowledgement: This work was supported, in part, by Lero through Science Foundation Ireland grant 13/RC/2094.

REFERENCES

[1] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on stack overflow," *arXiv preprint arXiv:1806.07659*, 2018.

[2] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "SOTorrent: Reconstructing and analyzing the evolution of stack overflow posts," *arXiv preprint arXiv:1803.07311*, 2018.

[3] S. Baltes, R. Kiefer, and S. Diehl, "Attribution required: stack overflow code snippets in github projects," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 161–163.

[4] D. Kavaler and V. Filkov, "Determinants of quality, latency, and amount of stack overflow answers about recent android apis," *PloS one*, vol. 13, no. 3, p. e0194139, 2018.

[5] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from stackoverflow: An exploratory study on android apps," *Information and Software Technology*, vol. 88, pp. 148–158, 2017.

[6] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online Q&A forum reliable?: A study of API misuse on stack overflow," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 886–896.

[7] S. Baltes, C. Treude, and S. Diehl, "SOTorrent: Studying the origin, evolution, and usage of stack overflow code snippets," in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)*, 2019.

[8] C. Bonja and E. Kidanmariam, "Metrics for class cohesion and similarity between methods," in *Proceedings of the 44th annual Southeast regional conference*. ACM, 2006, pp. 91–95.

[9] J. Al Dallal and L. C. Briand, "A precise method-method interaction-based cohesion metric for object-oriented classes," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 21, no. 2, p. 8, 2012.

[10] M. Ó Cinnéide, I. Hemati Moghadam, M. Harman, S. Counsell, and L. Tratt, "An experimental search-based approach to cohesion metric evaluation," *Empirical Software Engineering*, vol. 22, no. 1, pp. 292–329, 2017.