----------------------- REVIEW 1 ---------------------

----------- Questions to the authors -----------

The authors should clarify the motivation behind their study, focusing on how serious the problem is, how extensive it is, and how it affects software development. The assumption that code is being copied from stack overflow (where it is not maintained) to active projects, and thus introducing flaws into active code, is not really discussed and motivated convincingly.

The authors started from 315 million pairs and analyzed 32 thousand in detail. What is the implication of this drop in numbers on the motivation of this study?

*The motivation still holds. The number of 36,000 analysed clone pairs are the ones that are not trivial clones according to the clone agreement and our clone filtering process. It does not mean that these are the 'total' number of clones between Stack Overflow and Qualitas.*

*It is known that clone detection tools report a lot of false/trivial clones. We focus on the important ones.*

Can we conclude that this is not as serious a problem? The authors should discuss why this is a problem. Perhaps examples of flaws introduced in stack overflow being transferred on active projects would help in this matter.

*See citation [1] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. 2016. You Get Where You’re Looking for: The Impact of Information Sources on Code Security. In SP ’16. 289–305 for examples of flaws.*

*See citation [2] Le An, Ons Mlouki, Foutse Khomh, and Giuliano Antoniol. 2017. Stack Over ow: A Code Laundering Platform?. In SANER ’17 for impacts of licensing conflicts on real-world software projects. [Include SANER’17 results.]*

*A good example from our results:*

*Clone pair:*

*stackoverflow\_formatted/801987\_0.java (1,17)*

*hadoop-1/hadoop-1.0.0/src/core/org/apache/hadoop/util/StringUtils.java (40, 56)*

*The StringUtils.humanReadableInt(..) has a race condition (that’s why it’s fixed). However, the example on Stack Overflow never get changed.*

[*https://issues.apache.org/jira/browse/HADOOP-9252*](https://issues.apache.org/jira/browse/HADOOP-9252)

[*https://fisheye.apache.org/changelog/hadoop?cs=ef2ff99d36752d2e95236c4ab8e5290a2e41e5bf*](https://fisheye.apache.org/changelog/hadoop?cs=ef2ff99d36752d2e95236c4ab8e5290a2e41e5bf)

Some of the assessment in the manual analysis part seems subjective. The authors report that 61 out of 364 pairs that were manually analyzed by the two evaluators were subject to discussion. This suggests a high level of subjectivity in the analysis. This aspect, and its impact on the conclusions of the paper, should be discussed in more detail.

Would a different pair of evaluators following the same protocol expect the same result?

Yes. We have only 12 cases of conflicts between TP and FP. Most of the conflicts are regarding the classification into QS, SQ, EX, UD, BP and IN (TP pairs).

No. Classifications of clones by human are known to be very subjective.

Would a higher number of evaluators expect similar results?

Yes.

Code on stack overflow is usually provided as a minimum working example, or just as illustration, without any context.

Is it reasonable to expect this type of code to be maintained actively by the posters?

Is it reasonable to compare the quality of code and the maintenance aspect in this context with the quality and maintenance of code in active projects?

*That would be an ideal case. However, Stack Overflow should at least implement some checks to at least warn the users that this code has not been checked and probably be outdated or license-violating.*

*Although the code is provided as minimum working examples, it is known that the code is used as-is and causing problems in active projects [1,2]. Active maintenance and support provided by Stack Overflow can mitigate such problems.*

----------------------- REVIEW 2 ---------------------

----------- Questions to the authors -----------

Why non-open source projects were not considered in the case study?

*We are not aware of any curated set of software that includes non-open source projects. We focused on open source projects to allow replication of our study.*

How many license violations are indeed violations without attributing or acknowledging the sources of clone?

*See Table 11: 79 (QS) and 77 (EX) pairs (snippets) violate the license but acknowledge the source, 202 (UD) pairs violate the license and have no acknowledgement.*

----------------------- REVIEW 3 ---------------------

1) All the criteria and configurations of tools made sense. But, why is disagreed clone pairs inspected and not just disregarded?

*Disagreed clone pairs are also clones, they just have lower confidence to be true clones. By only inspecting agreed clones, one would miss several true clones due to the tools’ differences [61].*

3) How is a clone of code in a project across multiple Stack Overflow answers treated? Is that still considered a single pair, or is that two pair instances -- one for each accepted answer.

*We consider them as two clone pairs.*

4) Why is classification of boiler plate code not performed first? That would seem to make the search space much smaller, especially if the eliminate 90% of clone pairs.

*The automatic classification of boiler plate code is done first (Figure 2). The manual investigation (Figure 4) only classifies boiler plate clones that have been missed by the automatic classification. During the manual investigation, the classification of accidental clones is easier than the boiler plate classification, thus is comes second.*

5) Page 7, 25-44: What percentage of code does the found clone pairs represent? This can give a better sense of impact.

*See Table 7, last column: for the manual confirmed pairs, 516/144,064 of snippets with on average 26% of the snippets’ code. And for the reported pairs, 14,856/144,064 with an average of 38%. Note that it is restricted to only 111 Qualitas projects and one can expect a large number of clones to other projects.*

----------- Questions to the authors -----------

Please answer 1, 3, 4, 5.

------------------------------------------------------