Toxic Code Snippets on Stack Overflow

Anonymous Author(s)

ABSTRACT

FIXME: Outdated. Revise this last Online code clones are code fragments that are copied from software projects or online sources to Stack Overflow as examples in questions and answers. Due to an absence of a checking mechanism after the code has been copied to Stack Overflow, online code clones may suffer from becoming outdated or violating the original software license. We present a study incorporating several state-of-the-art tools and techniques to automatically extract, prioritise, and filter detected online clone pairs between 144,064 Java code snippets on Stack Overflow and 111 Java open source projects in the curated Qualitas corpus. We discovered and analysed 32,533 non-trivial online clone candidates. Our investigation of 3,636 manually confirmed clone pairs revealed strong evidence that 112 clones have been copied from a Qualitas project to Stack Overflow. We found 64 of them (57%) to be outdated and potentially harmful for reuse. Furthermore, we found 357 code snippets on Stack Overflow that violate the license of their original software.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories**; *Software evolution*;

KEYWORDS

Clone Detection, Stack Overflow, Outdated Code, Software Licensing

ACM Reference format:
Anonymous Author(s). 2017. Toxic Code Snippets on Stack Overflow. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 11 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Stack Overflow is a popular online programming community with 6.8 million users[1]. It allows programmers to ask questions and give answers to programming problems. The website has found to be useful for software development [14, 28, 38–40, 52, 53, 57] and also valuable for educational purposes [36]. On Stack Overflow, each conversation contains a question and a list of answers. The answers frequently contain at least one code snippet as a solution to the question asked. We found that the code snippets are usually not authored directly on the Stack Overflow website but copied from another location. A snippet in an answer could be copied and modified from a code snippet in the question, copied from the answerer's own code or from other locations including open source software (OSS) systems. The process of posting and answering questions on Stack Overflow that involves reusing of source code, can be considered as code cloning.

Code cloning is an activity of reusing source code by copying and pasting. It normally occurs in software development and account from 7% to 23% of source code in typical software systems [5]. The benefits and drawbacks of clones are still controversial. Several authors state that clones lead to bug propagations and software maintenance issues [24], while some others suggest that clones are not harmful and can even be beneficial [25, 46].

Code cloning can also have side effects such as violating software licenses or introducing software vulnerabilities. Carelessly cloning code from one project to another project with a different license may cause a software licensing violation [17]. This also happens within the context of online Q&A websites such as Stack Overflow. An et al. [2] showed that 1,279 cloned snippets between Android apps and Stack Overflow potentially violate software licenses. Security is also among the main concerns when code is copied from an online source. For example, Stack Overflow helps developers to solve Android programming problems more quickly than other resources while, at the same time, offers less secure code than books or the official Android documentation [1].

We call code snippets that are copied from software systems to online Q&A websites (such as Stack Overflow) and vice versa as "online code clones". There are two directions in creating online code clones: (1) code is cloned from a software project to a Q&A website as an example; or (2) code is cloned from a Q&A website to a software project to obtain a functionality, perform a particular task, or fixing a bug. Similar to classic clones, online code clones can lead to license violations, bug propagation, introduction of vulnerabilities, and re-use of outdated code. Unfortunately, online clones are difficult to locate and fix since the search space in online code corpora is larger and no longer confined to a local repository.

In this study, we are interested in locating online code clones on Stack Overflow that are cloned from open source projects and study them on two aspects: **outdated code** and **license conflicts.**

A motivating example of outdated online code clones can be found in an answer to a Stack Overflow question regarding how to implement `RawComparator` in hadoop[2]. Figure 1 shows—on the left—a code snippet embedded as a part of the accepted answer. The snippet shows how hadoop implements the `compare` method in its `WritableComparator` class. The code snippet on the right shows another version of the same method, but at this time extracted from the latest version (as of September 26, 2016) of hadoop. We can see that they both are highly similar except a line containing `buffer.reset(null,0,0);` which was added in the latest version. The added line is intended for cleaning up the reference in the `buffer` variable. While this change has already been introduced into the `compare` method in the latest version of hadoop, the code example in Stack Overflow post is still unchanged. This example shows that inconsistencies between online code clones and their originals can lead users of Stack Overflow to reuse outdated code.

[1]Statistics as of February 25, 2017 from http://stackexchange.com/sites

[2]http://stackoverflow.com/questions/22315734

Conference'17, July 2017, Washington, DC, USA
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

```
/* Code in Stack Overflow post ID 22315734 */
public int compare(byte[] b1,int s1,int l1, ...) {
  try {
    buffer.reset(b1,s1,l1); /* parse key1 */
    key1.readFields(buffer);
    buffer.reset(b2,s2,l2); /* parse key2 */
    key2.readFields(buffer);
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
  return compare(key1,key2); /* compare them */
}
```

```
/* WritableComparator.java (2016-09-26) */
public int compare(byte[] b1,int s1,int l1, ...) {
  try {
    buffer.reset(b1,s1,l1); /* parse key1 */
    key1.readFields(buffer);
    buffer.reset(b2,s2,l2); /* parse key2 */
    key2.readFields(buffer);
    buffer.reset(null,0,0); /* clean up reference */
  } catch (IOException e) {
    throw new RuntimeException(e);
  }
  return compare(key1, key2); /* compare them */
}
```

Figure 1: A motivating example of the two code fragments of `WritableComparator.java`. The one from the Stack Overflow post 22315734 (left) is outdated when compared to its latest version in the hadoop code base (right).

While research has mostly focused on reusing code snippets *from* Stack Overflow (e.g. [2, 28, 63]), fewer studies have been conducted on finding the origins of code examples copied *to* Stack Overflow and problems of reusing them such as outdated code and software licensing violations. This paper fills this gap and makes the following primary contributions:

(1) **A manual study of online code clones:** We used two clone detection tools to discover 315,786,118 similar code snippet pairs, followed by a manual classification of 3,636 candidate clone pairs between 144,064 Java code snippets obtained from Stack Overflow's accepted answers and 111 Java open source projects from the curated Qualitas corpus [56].

(2) **An investigation of outdated and license-violating online clones on Stack Overflow:** Our study shows that from the 3,636 online clones, at least 671 likely have been copied from open source projects or external online sources to Stack Overflow, potentially violating software licenses. For 112 of them, we found evidence that they have been copied from a specific Qualitas project and that 64 of them are meanwhile outdated.

(3) **An online code clone oracle:** The 3,636 manually investigated and validated online clone pairs are available for download[3] and can be used as a clone oracle.

2 EMPIRICAL STUDY

We performed an empirical study of online code clones between Stack Overflow and 111 Java open source projects to answer the following research questions:

RQ1 (Online code clones): *To what extent is source code cloned between Stack Overflow and open source projects?* We quantitatively measured the number of online code clones between Stack Overflow and open source projects to understand the scale of the problem.

RQ2 (Patterns of online code clones): *Why do online code clones occur?* We categorised online clones into seven categories allowing insights to why online code clones are created.

RQ3 (Outdated online code clones): *Are online code clones up-to-date compared to their counterparts in the original projects?* We were interested in the outdated Stack Overflow code examples since users are potentially reusing them.

RQ4 (Software licensing violation): *Do licensing conflicts occur between Stack Overflow clones and their originals?* We investigated whether the reuse of online code clones can cause software developers to violate licenses.

[3] https://to.be.disclosed.after.review

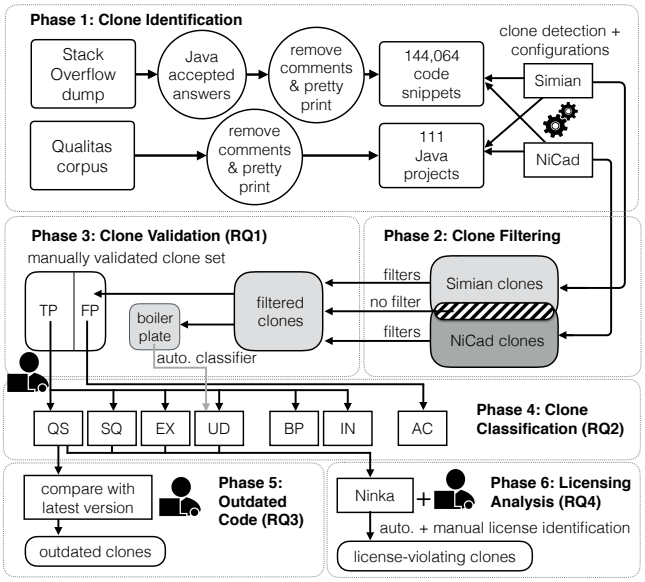

Figure 2: Experimental framework

To answer these four research questions, we perform an empirical study to locate and study the online code clones between Stack Overflow and open source projects. We designed our study in 6 phases as depicted in Figure 2 where we build different data sets to answer each of our four research questions.

2.1 Phase 1: Clone Identification

We rely on two source data sets in this study: Java code snippets in answers on Stack Overflow and open source projects from the Qualitas corpus [56].

Table 1: Stack Overflow and Qualitas datasets

Dataset	No. of files	SLOC	Median
Stack Overflow	144,064	2,332,940	10
Qualitas	166,709	19,614,083	60,667

Stack Overflow: We extracted Java code snippets from a snapshot of a Stack Overflow dump[4] in January 2016. The data dump is in XML format containing information of posts (questions and answers) and we were interested in code snippets embedded in posts which were located between <code>...</code> tags. A Stack Overflow thread contains a question and several answers. An answer can also be marked as **accepted answer** by the questioner if the solution fixes his/her problem. We collected Java code snippets using two criteria. First, we only focused on code snippets in accepted answers. We chose the snippets in accepted answers because they actually solved the problems in the questions. Moreover, they are always displayed just below the questions which makes them more likely to be reused than other answers. Second, we were only interested in code snippets of at least six lines. Snippets smaller than six lines are usually spurious clones [5]. Each snippet was extracted from the dump and saved to a file. We obtained 144,064 Java code snippets containing 2,332,940 lines[5] of Java source code. The median of the snippet size is 10.

Open source systems: We selected the established **Qualitas** corpus [56]. It is a curated Java corpus that has been used in several software engineering studies [4, 37, 55, 58]. The projects in the corpus represent various domains of software systems ranging from programming languages to visualisation. We selected the release 20130901r of the Qualitas corpus containing 112 Java open source projects. This release contains projects with releases no later than 1st September 2013. We intentionally chose an old corpus from 2013 since we are interested in online code clones in the direction from open source projects to Stack Overflow. The 20130901r snapshot provides Java code that is more than 2 years older than the Stack Overflow snapshot, which is sufficiently long for a number of code snippets to be copied onto Stack Overflow and also to observe if clones become outdated. Out of 112 Qualitas projects, there is one project, jre, that does not contain Java source code due to its licensing limitation [56] and is removed from the study. This resulted in 111 projects analysed in the study, for a total of 166,709 Java files containing 19,614,083 lines of code (see Table 1). The median project size is 60,667 lines of code.

Clone Detection Tools: We use clone detection to discover online code clones. There are a number of restrictions in terms of choosing the clone detection tools for this study. The main restriction is due to nature of code snippets posted on Stack Overflow, as most of them are incomplete Java classes or methods. Hence, a detector must be flexible enough to process code snippets that are not compilable or not complete blocks. Moreover, since the amount of code that has to be processed is in a scale of millions line of code (as shown in Table 1), a clone detector must be scalable enough to report clones in a reasonable amount of time. We have tried 5 state-of-the-art clone detectors including Simian [49], NiCad [11, 45], CCFinder [24], iClones [20], and DECKARD [23] against the Stack Overflow and Qualitas datasets. CCFinder, iClones, and DECKARD failed with execution errors after running for couple of hours. Thus, we removed them from the study. Simian and NiCad are flexible enough to handle code with incomplete methods or classes and successfully completed the detection.

[4]https://archive.org/details/stackexchange

[5]Measured by cloc: https://github.com/AlDanial/cloc

Table 2: Configurations of Simian and NiCad

Tool	Default (*D*)	EvaClone (*E*)
Simian (*S*)	threshold=6, ignoreStringCase, ignoreCharacterCase, ignoreModifiers	threshold=5, ignoreIdentifiers, ignoreIdentifierCase, ignoreStrings, ignoreCharacters, ignoreSubtypeNames, balanceSquareBrackets
NiCad (*N*)	Blocks, UPI=0.30, MinLine=10, MaxLine=2500	Blocks, UPI=0.20, MinLine=5, MaxLine=604, blind renaming, literal abstraction

Simian is a text-based clone detector that locates clones at line-level granularity and has been used extensively in several clone studies [8, 32, 34, 42, 61]. Furthermore, it offers normalisation of variable names and literals (strings and numbers) which enables Simian to detect literal clones (type-1) and parameterised clones (type-2). **NiCad** is also a text-based clone detector which detects clones at either method- or block-level granularity. It can detect clones of type-1, -2 up to type-3 (clones with added and removed statements) and has also been used in several empirical clone studies [34, 42, 45, 47, 54, 61]. NiCad utilises TXL [10] for parsing and pretty-printing source code and provides code normalisation by variable renaming and abstraction. We use a variant of NiCad called *nicadcross*. It offers the same functionalities as the original NiCad but is specialised for detecting code clones between two systems.

We prepared the Java code in both datasets by removing comments and pretty-printing to increase the clone detection accuracy. Then, we deployed the two detectors to locate clones between the two datasets. For each Qualitas project, we ran the tools on the project's code and the entire Stack Overflow data. Simian did not provide an option to detect cross-project clones. Hence the Simian clone report was filtered to contain only clone pairs between Stack Overflow and Qualitas projects, removing all clone pairs within either Stack Overflow or within Qualitas.

Clone Detection Configuration: We are aware of effects of configurations to clone detection results and the importance of searching for optimised configurations in empirical clone studies [42, 43, 54, 60]. However, considering the massive size of the two datasets and the search space of at least 15 Simian and 5 NiCad parameters, we are unable to search for the best configurations of the tools. Thus, we decided to configure Simian and NiCad using two established configurations: (1) the tools' default configurations chosen by the tools' creators (denoted as *D*), and (2) the discovered configurations for Bellon's Java projects from *EvaClone* [61], a study of optimising clone detectors' configurations based on clone agreement (denoted by *E*). Table 2 shows the two configurations. In total, we have four configurations: Simian with the default configuration (S_D), Simian with the EvaClone configuration (S_E), NiCad with the default configuration (N_D) and NiCad with the EvaClone configuration (N_E).

Table 3: Number of online clones reported by Simian (*S*) and NiCad (*N*) with default (*D*) and EvaClone (*E*) configurations

Stats	S_D	S_E	N_D	N_E
Total C_{pairs}	67,570	63,635,844	632,855	251,449,849
Avg. C_{size}	9.11	5.95	10.85	6.28

We encountered NiCad failures with a few Qualitas projects. N_D could not detect clones in hibernate due to clustering errors. N_E generated errors during code normalisation for 5 projects including vuze, hibernate, myfaces, netbeans, and spring. We have contacted the creator of NiCad regarding the issues. They identified the issues as problems in the TXL grammar and as problems of long file paths, and will be fixed in the next NiCad releases.

The number of online clone pairs reported are presented in Table 3. S_D reports 67,570 clone pairs while N_D reports 632,855 clone pairs. S_E and N_E report much larger numbers of clone pairs, 63,635,844 and 251,449,849 pairs respectively. This is expected since EvaClone configurations prefer recall [61]. The average clone size reported by S_D is 9.11 lines which is bigger than for S_E (5.95 lines). Similarly, N_D has an average clone size of 10.85 lines which is bigger than 6.28 lines reported by N_E. The drop in size is due to the EvaClone configuration with a smaller minimum size (Table 2).

2.2 Phase 2: Clone Filtering

As there are in total 315,786,118 clone pairs reported, it is infeasible for humans to manually validate them all. Instead of investigating a random sample, we adopted the idea of **clone agreement** which has been used in clone research studies [16, 43, 61] in situations where a clone oracle is missing or impossible to establish. Clone pairs agreed by multiple clone detection tools have a higher likelihood to be real clones [43]. By using this agreement-based approach, we reduced the number of clone candidates for manual investigation to the ones agreed by multiple tools. To find agreement between two clone pairs reported by two different tools, we used the clone pair matching metric proposed by Bellon et al. [5]. Two clone pairs which have a large enough number of overlapping lines can be categorised as either a good-match or an ok-match pair (denoted as *good* and *ok* respectively) with a confidence value between 0 and 1. A *good* clone pair has stronger agreement than an *ok* pair. We follow Bellon's original definitions of good- and ok-match [5], which are based on how much two clone fragments *CF* overlap[6] and how much of them is contained in each other:

$$overlap(CF_1, CF_2) = \frac{|lines(CF_1) \cap lines(CF_2)|}{|lines(CF_1) \cup lines(CF_2)|}$$

$$contained(CF_1, CF_2) = \frac{|lines(CF_1) \cap lines(CF_2)|}{|lines(CF_1)|}$$

A clone pair *CP* is formed by two clone fragments CF_1 and CF_2, i.e. $CP = (CF_1, CF_2)$, and the *good-value* and the *ok-value* of two clone pairs are defined as

$$good(CP_1, CP_2) = min(overlap(CP_1.CF_1, CP_2.CF_1), overlap(CP_1.CF_2, CP_2.CF_2))$$

[6]The overlap value can be considered as Jaccard similarity.

Table 4: Distribution of the agreed clone pairs

Set	S_D–N_D	S_D–N_E	S_E–N_D	S_E–N_E	Unique
good	23	26	10	2,267	2,313
ok	12,063	993	79	20,021	32,316

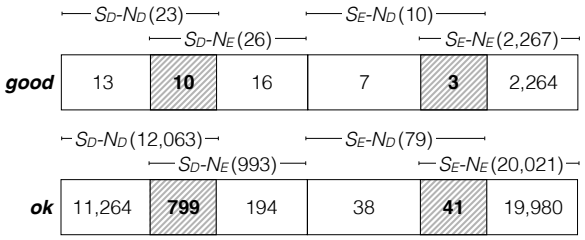

Figure 3: Distribution of *good* (top) and *ok* pairs (bottom) reported by the 4 configurations of Simian and NiCad. The shaded areas are overlapping pairs.

$$ok(CP_1, CP_2) = min(max(contained(CP_1.CF_1, CP_2.CF_1), contained(CP_2.CF_1, CP_1.CF_1)), max(contained(CP_1.CF_2, CP_2.CF_2), contained(CP_2.CF_2, CP_1.CF_2)))$$

Two clone pairs CP_1 and CP_2 are called a *good-match(t)* or *ok-match(t)* iff, for threshold $t \in [0, 1]$ holds

$$good(CP_1, CP_2) \geq t$$
$$ok(CP_1, CP_2) \geq t$$

Using the good- and ok-match criteria with a predefined threshold *t*, we can prune the 315 million candidate clone pairs for manual investigation. The *good* pairs are the ones with the highest confidence, followed by the *ok* pairs, and followed by clone pairs without agreement. We call the *good*- and *ok*- pairs **agreed** clone pairs and the clone pairs reported by Simian or NiCad without agreement **disagreed** clone pairs.

Having two clone detectors, Simian (denoted as *S*) and NiCad (denoted as *N*), with two chosen configurations (*D* and *E*) each, we used Bellon's good- and ok-match for agreement in four possible pair-wise combinations: S_D–N_D, S_D–N_E, S_E–N_D, and S_E–N_E.

Agreed clone pairs: As shown in Table 4, after we applied clone agreement filtering using Bellon's criteria[7], the number of *good* clone pairs is 2,313 and the number of *ok* clone pairs is 32,316, 0.1‰ of all reported clone pairs. Agreed clone pairs are clone pairs that pass Bellon's *good* or *ok* criteria, and therefore are selected for manual classification. Due to NiCad's failures, we do not have NiCad clones from five Qualitas projects in the agreed clone pairs. Simian clones of the same projects also disappeared from the agreed clone pairs since they could not be matched.

The distribution of *good* clone pairs between the four combinations of *D* and *E* configurations are listed in Table 4. There are 2,326 (2,313 unique) *good* pairs consisting of 23 pairs from S_D–N_D, 26 pairs from S_D–N_E, 10 pairs from S_E–N_D, and 2,267 pairs from

[7]Similar to the original study, we selected a threshold of 0.7 for *good* and *ok* pairs [5].

Table 5: Disagreed clone pairs before and after filtering.

Set	Before Filtering	Filters*		After Filtering
		L	S	
S_D	55,274	✓		17,801
N_D	620,680	✓	✓	83

*L=size ≥ 10, S=sim ≥ 85%.

S_E–N_E. There are 33,156 (32,316 unique) *ok* pairs[8]. We obtained 12,063; 993; 79; and 20,021 pairs from S_D–N_D, S_D–N_E, S_E–N_D, and S_E–N_E respectively. Between the four configuration sets, there is considerable amount of clone pairs shared between two adjacent sets (as depicted in Figure 3), but there is no clone pair that is agreed by all four combinations.

Disagreed clone pairs: The disagreed clone pairs are clone pairs that are reported by a single tool, either Simian or NiCad, and do not have agreement with the other tool. The disagreement can be from a misalignment of clone lines or completely non-overlapping clones. The disagreed pairs from Simian include clone pairs in the projects with NiCad's errors (1 project from N_D and 5 from N_E). With the four configuration combinations, we decided to investigate only S_D, and N_D, and dropped S_E and N_E due to their enormous amount of clone pairs (60 and 250 millions respectively). Moreover, S_E and N_E contain a large number of false positives due to the recall preference of their EvaClone configurations.

Even choosing only the default configurations, the number of disagreed clone pair candidates is still too large for manual inspection (as shown in Table 5). We applied two filters: clone size and similarity threshold. For the *clone size filter*, we raised the minimum clone size to 10 lines since larger clones are more interesting, while smaller ones tend to be false clones [46]. The second filter, *similarity threshold*, applies only to NiCad clone pairs since Simian does not provide a similarity configuration.

The two filters reduced the number of S_D clones to 17,801 pairs. We filtered N_D clones by additionally raising the similarity threshold. We increased NiCad's similarity threshold from 70% to 85% (by adjusting NiCad's UPI parameter from 0.3 to 0.15). We tried varying the threshold from 70% (default, 632,855 pairs), 80% (25,003 pairs), 85% (185 pairs) and 90% (83 pairs) and found that 85% similarity provided the best balance in terms of the amount of reported clones and precision for small clones. Using this filter along with the other three filters, we reduced the number of N_D clones to 83 pairs.

Selection of clones: We selected all *good* pairs (2,313) and the *ok* pairs for S_D–N_D (12,336), S_E–N_D (993), and S_D–N_E (79), but we ignored 20,021 $S_E N_E$ *ok* clone pairs in the classification due to their large number of clones and the very small average clone sizes of 5.95 and 6.28 lines (see Table 3), leading to a high probability of being trivial clones. We refer to the set of clone pairs as the *good* set and the *ok'* set. Together with the 17,801 (referred as S'_D set) and 83 (referred as N'_D set) clone pairs after filtering, we ended up with a set of 32,533 online clone pair candidates.

Before performing the manual investigation, we created and used an automatic classifier to classify boiler-plate code of `equals()`, `hashCode()` and getters/setters (depicted with a grey line in Figure 2).

[8]This excludes the subsumed 2,313 *good* pairs because *ok* pairs subsume *good* pairs.

Table 6: The seven patterns of online code cloning

Patt.	Description
QS	Cloned from Qualitas project to Stack Overflow (Q → S).
SQ	Cloned from Stack Overflow to Qualitas project (S → Q).
EX	Cloned from an external source X (X → S ∧ X → Q).
UD	Cloned from each other or from an external source X outside the project (unknown) (S ↔ Q ∨ (X → S ∧ X → Q)).
BP	Boiler-plate or IDE auto-generated
IN	Inheritance, interface implementation
AC	Accidental similarity, false clones

The classifier could automatically classify 28,897 clone pairs and left 3,636 clone pairs for manual investigation in phases 3 to 6.

2.3 Phase 3-4: Validation and Classification

We used the 3,636 filtered clone pairs after automatic classification for manual validation and classification. The validation and classification of the pairs was done at the same time. The clone validation process (phase 3 in Figure 2) involves checking if a clone pair is a true positive or a false positive. Moreover, we are also interested in the patterns of code cloning so we can gain more insights into how these clones are created (phase 4 in Figure 2).

Manual investigation: The first author, who has been working on clone detection research for two years, took the role of the main investigator performing a manual validation and classification of the filtered clone pairs. The main investigator went through each clone pair candidate, looked at the clones, and decided if they are a true positive or a false positive and classified them into an appropriate pattern. To mitigate the human error of the manual investigation, the second author took the role of the validating investigator, performing a validation on 10% (364 pairs) of the clone pairs judged by the main investigator. After the validation, the results from the two investigators were compared and 61 conflicts were discussed and resolved. There were three conflicts where the two investigators could not find a consensus and the third author was involved for the final judgement.

The online cloning classification patterns: We studied the eight patterns of cloning from Kapser et al. [25, 27] and performed a preliminary study to evaluate its applicability to our study. We tried to classify 697 online clone pairs from the reported clones in phase 1 using Kapser's cloning patterns. We found that Kapser's patterns are too broad for our study and a more suitable and fine-grained classification scheme is needed. After a preliminary study, we adopted one of Kapser's cloning patterns, *boiler-plate code*, and defined six new cloning patterns. The seven patterns include QS, SQ, EX, UD, BP, IN, and AC as presented in Table 6. Pattern QS (**Q**ualitas to **S**tack Overflow) represents clones that have evidence of being copied from a Qualitas project to Stack Overflow. The evidence of copying can be found in comments in the Qualitas source code or in the Stack Overflow post's contents. Pattern SQ (**S**tack Overflow to **Q**ualitas) is cloning, with evidence, in the opposite direction from Stack Overflow to a Qualitas project. Pattern EX (**Ex**ternal Sources) is cloning that has evidence of copying from a single or multiple external sources to Stack Overflow and to a Qualitas project. Pattern UD (**U**nknown **D**irection) is cloning that creates identical or highly similar clones between Qualitas and Stack Overflow but where we

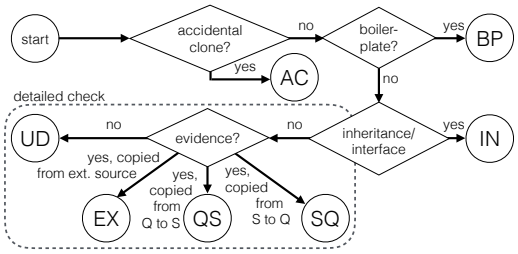

Figure 4: Online code clone classification process

could not find any attribution of copying. Pattern BP (**B**oiler-**P**late) represents clones containing boiler-plate code of `equals()` methods, getters/setters, or IDE-generated code such as GUI components. Pattern IN (**In**heritance/Interface) is cloning by inheritance of the same super class or implementation of the same interface. These two activities usually result in similar overriding methods. The last pattern, AC (**A**ccidental **C**lones), represents accidentally similar clone pairs. These are mainly false positive clones from the clone detectors such as similar `try-catch` statements.

The classification of the filtered online clone pairs followed the steps depicted in Figure 4. First, we look at a pair of clone fragments to see their similarity. If they are accidentally similar clones after code normalisation or false positive clones from the clone detection tools, we classify the pair into **AC**. If the two fragments are boiler-plate code, the pair is classified into **BP**. If they implement the same interface or inherited the same class and share similar overriding methods, the pair is classified into **IN**. If the pair is not **BP**, **IN**, or **AC**, we start a detailed investigation. We check the corresponding Stack Overflow post, read through it carefully and look for any evidence mentioning code copying. If evidence of copying has been found from a Qualitas project, the pair is classified in **QS**. In several occasions, we used extra information such as questions' contents, name of posters, and tags to gain a better understanding. On the other hand, if the source code from the Qualitas project mentions copying from Stack Overflow, the pair is classified into **SQ**. If there is evidence of copying from an external source instead of a Qualitas projects, the pair is classified into **EX**. Lastly, if there is no evidence of copying in any direction but the clone fragments are highly similar, we classify them into **UD**.

2.4 Phase 5: Outdated Clones

Outdated code occurs when a piece of code has been copied from its origin to another location and later the original has been updated [62]. Usually code clone detection is used to locate clone instances and update them to match with the originals [5]. However, online code clones are more difficult to detect than in regular software projects due to its large search space and a mix of natural and programming languages combined in the same post.

To search for outdated online code clones, we focused on the **QS** clone pairs that were cloned from Qualitas to Stack Overflow and compared them with their latest versions. We downloaded the latest version of the Qualitas projects from their repositories on September 26, 2016. For each **QS** online clone pair, we used the clone from Qualitas as a proxy. We searched for its latest version by the file name and located the cloned region in the file based on the method name. We then compared the Stack Overflow snippet to its latest version line-by-line to find if any change has been made to the source code. We also made sure that the changes did not come from the modifications made to the Stack Overflow snippets by the posters but from the updates in the projects themselves. When we found inconsistent lines between the two versions, we used `git blame` to see who modified those lines of code and the timestamps.

2.5 Phase 6: Licensing Analysis

Software licensing plays an important role in software development. Violation of software licenses impacts software delivery and also leads to legal issues [51]. One can run into a licensing issue if one integrates third-party source code into their software without checking. A study by An et al. [2] reports 1,279 cases of potential license violations between 399 Android apps and Stack Overflow code snippets.

We analysed licensing conflicts of the online clones in the **QS**, **EX**, and **UD** set. The licenses were extracted by *Ninka*, an automatic license identification tool [18]. Since Ninka works at file level, we report the findings based on Stack Overflow snippets and Qualitas source files instead of the clone pairs (duplicates were ignored). For the ones that could not be automatically identified by Ninka and have been reported as `SeeFile` or `Unknown`, we looked at them manually to see if any license can be found.

3 RESULTS AND DISCUSSION

We followed the 6 phases in the experimental framework (Figure 2) to answer our four research questions. To answer RQ1, we rely on the number of manually validated true positive online clone pairs in phase 3. We use the results of the manual classification by the seven patterns of online code cloning to answer RQ2 (phase 4). For RQ3, we looked at the true positive clone pairs that are classified as clones from Qualitas to Stack Overflow and checked if they have been changed after cloning (phase 5). Similarly, for RQ4, we looked at the license of each clone in the pattern QS, EX, UD and checked for a possibility of licensing violation (phase 6).

3.1 RQ1: Online Code Clones

The statistics on clones obtained from the different data sets are presented in Table 7. Simian in the default configuration (S_D) reported clones in 1,086 snippets, approximately 1% of the 144,064 Stack Overflow snippets, associated with 92 Qualitas projects. Simian in the EvaClone configuration (S_E) reported 1,531 snippets containing clones associated with all 111 Qualitas projects. NiCad in the default configuration (N_D) reported clones between 1,392 Stack Overflow code snippets and 88 Qualitas projects. NiCad in the EvaClone configuration (N_E) reported the largest number of 12,886 cloned snippets mainly due to the more relaxed configuration. The N_E snippets are associated with 88 Qualitas projects. For the cloned Stack Overflow snippets, the average ratio of cloned code for S_D, S_E, N_D, and N_E is 38%, 50%, 30%, and 33% respectively.

However, using Bellon's criteria for matching clone pairs, only 0.1‰ of all pairs are agreed by two tool/configuration combinations. As shown in Table 7, for the 2,313 *good* pairs, there are 97 Stack Overflow snippets which are associated with 67 Qualitas projects. The 12,336 *ok*' pairs are clones between 139 Stack Overflow snippets

Table 7: Investigated online clone pairs and corresponding snippets and Qualitas projects for the different data sets

Data Set	Pairs	Snippets	Projects	Avg. C%
Reported Clones				
S_D	67,570	1,086	92	38%
S_E	63,635,844	1,531	111	50%
N_D	632,855	1,392	88	30%
N_E	251,449,849	12,886	88	33%
Total	315,786,118	14,856	111	38%
Filtered Data Set				
good	2,313	97	67	23%
ok'	12,336	139	43	28%
S'_D	17,081	460	74	31%
N'_D	83	44	19	27%
Total	32,533	623	96	27%
True Positives in Manually Validated Clones				
good/AC	131	51	24	22%
ok'/AC	152	83	22	21%
S'_D/AC	873	337	60	35%
N'_D/AC	60	33	15	27%
Total	1,216	516	75	26%

and 43 Qualitas projects. Moreover, for the disagreed pairs, we found 17,081 clone pairs reported by S'_D are clones between 460 Stack Overflow snippets and 74 Qualitas projects and 83 N'_D clone pairs are clones between 44 Stack Overflow snippets and 19 Qualitas projects. In total we found that the filtered data sets contains 32,533 clone pairs which are clones between 623 Stack Overflow snippets and 96 Qualitas projects.

During the manual investigation of 3,636 clone pairs, we identified 2,420 pairs as being accidential clones (AC), i.e. false positives. After removing them, the set still contains 1,216 true positive clone pairs between 516 Stack Overflow snippets and 75 Qualitas projects.

To answer RQ1, we found most Qualitas projects to share similar code with Stack Overflow code snippets. In the manually confirmed data set of 3,636 clone pairs, we found 1,216 pairs as true positives which account for 75 Qualitas projects and 516 Stack Overflow code snippets.

3.2 RQ2: Patterns of Online Code Cloning

As shown in Table 8, the automatic classifier helped us to classify 28,897 clone pairs into pattern **BP**. Then, we relied on a manual classification on the remaining 3,636 pairs by following the classification process in Figure 4. The classification results are shown in Table 9 and explained in the following.

QS: Qualitas → Stack Overflow. We found 112 online clone pairs with evidences of cloning from 19 Qualitas projects to Stack Overflow. They include 3, 14, 86, and 9 pairs from the *good*, *ok'*, S'_D, and N'_D set respectively. Most of them have a statement in the Stack Overflow post saying that the code is "copied", "borrowed" or "modified" from a specific file or class in a Qualitas project. For example, according to the motivating example in Figure 1, we found evidence in the Stack Overflow Post 22315734 saying that *"Actually, you can learn how to compare in Hadoop from WritableComparator. Here is an examle that borrows some ideas from it."*

Table 8: No. of online clones candidates for the classification

Set	Candidates	Classification	
		Auto	Manual
good	2,313	10	2,303
ok'	12,336	12,077	259
S'_D	17,801	16,800	1,001
N'_D	83	10	73
Total	32,533	28,897	3,636

Table 9: Classifications of online clone pairs.

Set	QS	SQ	EX	UD	BP	IN	AC	Total
good	3	0	4	15	111	8	2,172	2,313
ok'	14	0	44	20	12,128	23	107	12,336
S'_D	86	0	146	293	17,074	74	128	17,801
N'_D	9	0	6	31	17	7	13	83
Total	112	0	200	359	29,330	112	2,420	32,533

The most cloned projects are hibernate and spring with 20 clone pairs, followed by eclipse (15 pairs), and hadoop (12 pairs). The clones are used as examples and are very similar to their original Qualitas code with limited modifications.

SQ: Stack Overflow → Qualitas. We did not find any pairs with evidence of cloning from Stack Overflow to Qualitas projects. This is very likely due to the age of the Qualitas corpus as another study [2] showed the presence of clones from Stack Overflow in newer open source data sets.

EX: External Sources. We found 200 clone pairs with evidence of cloning from external sources to Stack Overflow and to Qualitas. They include 4, 44, 146, and 6 pairs from the *good*, *ok'*, S'_D, and N'_D set respectively. All of them contain statements giving an external source of the cloned code. For example, Stack Overflow Post 9549009 contains a code comment saying *"Copied shamelessly from org.bouncycastle.crypto.generators.PKCS5S2ParametersGenerator"*.

These findings complement a study of clones between software projects [54]. We found that cloning can also happen among different sources on the Internet just like software projects. There are 30 clone pairs that originated from programming websites. For example, we found a clone of a snippet to convert numbers into words which is copied from www.rgagnon.com/javadetails/java-0426.html. Both the Stack Overflow and the compiere project code contain an attribution to the original source. Another example is a snippet to generate graphical *Perlin noise*. It is copied from http://mrl.nyu.edu/~perlin/noise/ and is used on Stack Overflow and in the aoi project with attribution. There are other EX pairs which are copied from third-party libraries such as zxing, jasper, java.util, javax.servlet, and the bouncycastle cryptography project.

UD: Unknown Direction. We found 359 online clone pairs with no evidence of cloning between Qualitas and Stack Overflow but with a high code similarity that suggests cloning. The most cloned projects is netbeans with 93 clone pairs. Most of the clones are a large chunk of code handling GUI components. Although these GUI clones might be auto-generated by IDEs, we did not find

Table 10: Examples of the outdated QS online clones

Stack Overflow		Qualitas						Changes	
Post	Date	Project	Ver.	File	Start	End	Date	Type*	Date
22315734	11-Mar-14	hadoop	1.0.0	WritableComparator.java	44	54	25-Aug-11	*S*	20-Nov-14
23520731	7-May-14	hibernate	4.2.2	SchemaUpdate.java	115	168	22-May-13	*S*	5-Feb-16
21734562	12-Feb-14	tomcat	7.0.2	FormAuthenticator.java	51	61	4-Aug-10	*R*	4-Aug-16
12593810	26-Sep-12	poi	3.6	WorkbookFactory.java	18	28	7-Dec-09	*R*	26-Jun-13
8037824	7-Nov-11	jasperreports	3.7.4	JRVerifier.java	1221	1240	31-May-10	*D*	20-May-11
3758110	21-Sep-10	spring	3.0.5	DefaultAnnotationHandlerMapping.java	78	92	20-Oct-10	*D*	20-Jan-12

* *S*: modified/added/deleted statements, *D*: file has been deleted, *R*: method has been rewritten completely

any evidence. The second most cloned project is jung2 (62 pairs), a network and graph framework, followed by eclipse-SDK (37 pairs).

BP: Boiler-Plate. There were a large amount of boiler-plate clone pairs found in this study. We observed 29,330 such clone pairs. The BP clone pairs account for 90% of all clone pairs we classified. The majority of them are `equals()` methods.

IN: Inheritance/interface. There were 112 clone pairs found to be similar because they implement the same interface or inheriting from the same class. An example is the two implementations of `MouseListener` that share similar `@Override` methods of `mousePressed`, and `mouseReleased`.

AC: Accidental Clones. There were 2,420 accidental clone pairs. Mainly, they are false positive clones caused by code normalisation. Other clone instances include `finally` or `try-catch` clauses that were accidentally the same due to their very small sizes, and similar `switch-case` statements.

To answer RQ2, we found that a large amount of the clone pairs between Stack Overflow and Qualitas projects are boiler-plate code (29,330) or accidental clones (2,420) with no evidence that the code has actually been copied. Nevertheless, we found 112 pairs with strong evidences to be cloned from 19 Qualitas projects to Stack Overflow and 200 pairs were found to be cloned to Stack Overflow from external sources.

3.3 RQ3: Outdated Online Code Clones

We discovered 64 outdated online clone pairs out of 112 pairs. As shown in Figure 5, spring has the highest number of 17 outdated pairs, followed by 7 from hibernate and hadoop, and 6 from junit. Besides the example of outdated code in `WritableComparator.java` from hadoop shown in Figure 1, we also found a few outdated code elements which contained a large amount of modifications. For example, the code snippet in Stack Overflow post 23520731 is a copy of `SchemaUpdate.java` in hibernate. The code has been heavily modified on February 5, 2016.

In the 64 outdated pairs, we found 5 "dead" snippets. These snippets cannot be located in the latest version of the projects. For example, the snippet in Stack Overflow post 3758110, a copy of `DefaultAnnotationHandlerMapping.java` in spring, was deleted in the commit `02a4473c62d8240837bec297f0a1f3cb67ef8a7b` by Chris Beams on January 20, 2012, two years after it was posted.

Table 10 shows examples of the outdated online clones on Stack Overflow. The table displays information of the clones from both Stack Overflow and Qualitas side including the dates. We summarise the changes that make the clones outdated into three types,

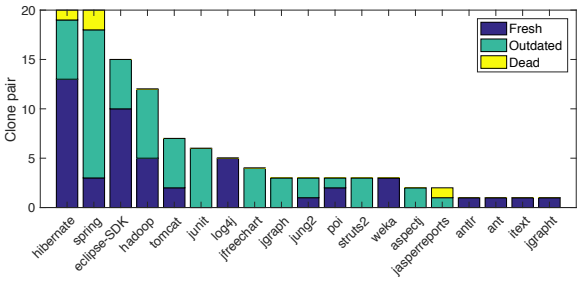

Figure 5: Status of the 112 QS online clone pairs

modified/added/deleted statements (*S*), file deletion (*D*), and method rewriting (*R*), along with the date of change. The complete set of 64 outdated online clones can be found from the study website.

The outdated online code clones cause problems ranging from uncompilable code (due to modifications and different API usage in the outdated code) to introducing vulnerabilities to software [62]. An outdated code with a subtle change (e.g. Figure 1) may be copied and reused without awareness from developers. Although Stack Overflow has a voting mechanism that may mitigate this issue, the accepted answer is still used by naive developers who copy and reuse the outdated code.

For RQ3, our results show that 57% (64) of QS clone pairs on Stack Overflow are outdated. 60 pairs differ from their newest versions by modifications applied to variable names or method names, added or deleted statements, to a fully rewritten code with new method signatures. 4 pairs are dead snippets.

3.4 RQ4: Software Licensing Violation

In our study, we reveal another possible situation of software licensing issues caused by code cloned to Stack Overflow. We found evidence that 112 pieces of code have been copied from Qualitas projects to Stack Overflow as examples. Their status of accepted answers increase their chances of being reused. Even though most of the Qualitas projects came with a software license, we found that the license information were frequently missing after the code was copied to Stack Overflow. The licensing terms on top of source code files are not copied because usually only a small part of the file was cloned. In overall, we can see that most of the Stack Overflow snippets do not contain licensing terms while their clone counterparts in Qualitas projects do. The summary of licensing information is listed in Table 11.

Table 11: License mapping of online clones (file-level)

Type	Stack Overflow (CC BY-NC-SA)	Qualitas	QS	EX	UD
Compatible License	Apache2	Apache2	0	0	1
	EPLv1	EPLv1	1	0	1
	Sun (proprietary)	Sun (proprietary)	0	1	0
	MIT-like	MIT-like	0	1	0
	No license	CC BY-NC-SA	0	5	0
	No license	No license	5	4	21
Total			6	11	23
Incompat. License	BSD3	GPLv2+	0	0	1
	No license	AGPLv3/3+	1	0	2
	No license	Apache-2	41	6	23
	No license	BSD3	3	4	28
	No license	CDDL or GPLv2	1	0	58
	No license	CDDLv1	0	1	0
	No license	EPLv1	7	0	10
	No license	GPLv2/2+/3+	2	62	32
	No license	LGPLv2/2.1+/3+	24	0	32
	No license	MPLv1.1	0	0	1
	No license	spdxBSD3	0	0	10
	No license	SPLv1	0	1	0
	No license	Unknown	0	3	3
	spdxBSD3	AGPLv3+	0	0	2
Total			79	77	202

Compatible license: There are 40 pairs which have compatible licenses such as *Apache license v.2*; *Eclipse Public License v.1 (EPLv1)*; or a pair of no license vs. *Creative Common Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)*. These clones are safe for being reused. Since source code and text on Stack Overflow are protected by *CC BY-NC-SA 3.0* , we can treat the Stack Overflow code snippets without licensing information as having *CC BY-NC-SA 3.0* by default. The *CC BY-NC-SA 3.0* license is relaxed and it only requests an attribution when reused.

Incompatible license: there are 358 clone pairs which do not contain licensing information after they are posted on Stack Overflow or contain a different license from their Qualitas clone counterparts. More than half (79) of **QS** clone pairs have their licensing terms removed or changed when posted on Stack Overflow. For **EX** clone pairs, we searched for licensing terms of the original source code from the external sources. We found that 77 out of 88 EX clone pairs have incompatible licenses. Similarly, the license statement was removed from Stack Overflow snippets. Of 226 **UD** clone pairs, 202 pairs have incompatible licenses. Again, most clones in Qualitas contain a license while the Stack Overflow snippets do not.

For RQ4, we found 358 code snippets on Stack Overflow that violate the license of their original software. The majority of them do not contain licensing statements after they have been copied to Stack Overflow.

3.5 Overall Discussion

Our study discovers links from code in open source projects to code snippets on Stack Overflow using clone detection techniques. These links enable us to discover outdated code and licensing problems. The links can be exploited further to mitigate the problems of reusing outdated online clones and incompatible license on Stack Overflow code snippets. We propose the following actionable items:

- **Preventive measure:** We encourage Stack Overflow to enforce attribution when source code snippets have been copied "from" licensed software projects to Stack Overflow. Moreover, an IDE plug-in that can automatically detect pasted source code and follow the link to Stack Overflow and then to the original open source projects, could also prevent the issue of license violation.
- **Detective measure:** A system to detect outdated source code snippets on Stack Overflow is needed. The system can leverage the online clone detection techniques in this study to periodically check if the cloned snippets are still up-to-date with their originals. With such a system, the poster can be notified when the code has been updated in the original project so that he/she can update their code on Stack Overflow accordingly. On the other hand, with a crowdsourcing solution using an IDE plug-in, developers can also report the corrected version of outdated code back to the original Stack Overflow threads when they reuse outdated code and make corrections to them.

The implementation and the evaluation of the above actionable items is part of the agenda of our future work.

4 THREATS TO VALIDITY

Internal validity: We could not verify all the 315,786,118 clone pairs reported by the clone detection tools in this study. Nevertheless, we applied different mechanisms to ensure the validity to the subset of clone pairs we classified. First, we used two widely-used clone detection tools, Simian and NiCad. We tried three other clone detectors but could not add them to the study due to their scalability issues and susceptibility to incomplete code snippets. We mitigated the problem of clone detector's parameter sensitivity by adding another established configuration from the EvaClone study [61]. Second, we adopted Bellon's agreement metric [5] to prioritise and filters clone pairs for the manual classification. The agreed clone pairs that passed *good-* and *ok-* criteria had a higher chance to be true positive. Nevertheless, we might still suffers from false negatives, i.e. online code clones that are not reported by the tools or are filtered out by the automatic classifier before the manual investigation.

Our seven patterns of online code cloning may not cover all possible online cloning patterns. However, instead of defining the patterns beforehand, we resorted to extract them from the data sets. We derived them from a manual investigation of 679 online clone pairs and adopted one pattern from the study by Kapser et al. [26].

The 3,636 clone pairs classified by the first author are subject to manual judgement and human errors. Although we tried our best to be careful on searching for evidence and classifying the clones, some errors may still exist. We mitigated this problem by having the second author to validate the classifications with a random sampling. The third author made a final judgement when no consensus could be found by the two authors. This validation process can be even improved by employing an external investigator.

External validity: We carefully chose the data sets for our experiment so the findings could be generalise as much as possible.

We selected Stack Overflow because it is one of the most popular programming Q&A websites available with approximately 6.8 million users. There are a large number of code snippets reused from the site [2] and there are also several studies encouraging of doing so (e.g. [28, 38–40]). Nonetheless, it may not be representative to all the programming Q&A websites.

Regarding the code snippets, we downloaded a full data dump and extracted Java accepted answers since they are the most likely ones to be reused. Our findings are limited to these restrictions. They may not be generalised to all programming languages and all answers on Stack Overflow. We chose the curated Qualitas corpus for Java open source projects containing 111 projects [56]. The projects span over several areas of software and has been used in several empirical studies [4, 37, 55, 58]. Although it is a curated and well-established corpus, it may not fully represent all Java open source software available.

5 RELATED WORK

Stack Overflow is a gold mine for software engineering research and has been put to use in several previous studies. In terms of developer-assisting tools, Seahawk is an Eclipse plug-in that searches and recommends relevant code snippets from Stack Overflow [39]. A follow up work, Prompter, by Ponzanelli et al. [40] achieves the same goal but with improved algorithms. The code snippets on Stack Overflow are mostly examples or solutions to programming problems. Hence, several code search systems use whole or partial data from Stack Overflow as their code search databases [14, 28, 38, 52, 53]. Furthermore, Treude et al. [57] use machine learning techniques to extract insight sentences from Stack Overflow and use them to improve API documentation.

Another research area is knowledge extraction from Stack Overflow. Nasehi et al. [36] studied what makes a good code example by analysing answers from Stack Overflow. Similarly, Yang et al. [63] report the number of reusable code snippets on Stack Overflow across various programming languages. Wang et al. [59] use Latent Dirichlet Allocation (LDA) topic modelling to analyse questions and answers from Stack Overflow so that they can automatically categorise new questions. There are also studies trying to understand developers' behaviours on Stack Overflow, e.g. [6, 9, 35, 44].

Code clone detection is a long-standing research topic in software engineering. Whether clones are good or bad for software is still controversial [19, 21, 22, 26, 27, 31, 47]. Code clones have several applications such as software plagiarism detection [41], source code provenance [12], and software licensing conflicts [17].

Two code fragments are clones if they are similar enough according to a given definition of similarity [5]. Given an open interpretation of "definition of similarity", there are various clone detection tools and their siblings, code plagiarism detectors, invented based on plethora of different code similarity measurements [42, 45, 54]. Many tools do not work on original source code directly but detect clones at an intermediate representation such as tokens [7, 15, 20, 24, 33, 41, 47, 48, 50], AST [3, 23] or program dependence graphs [29, 30].

Cloning patterns is initially defined by Kapser et al. [26, 27] by studying clones in Linux file systems and deriving 11 patterns of code cloning. Our study adopted one of the patterns into our online code cloning patterns.

Clone agreement is useful when a clone oracle is absent. By exploiting the different behaviours of clone detectors, one can look for their agreement and obtain highly-confident clones [5, 61]. Clone pairs that are agreed by multiple tools are more assured to be true clones than the ones reported by only a single tool [16, 43, 61].

Software licensing is crucial for open source and industrial software development. Di Penta et al. [13] studied the evolution of software licensing in open source software and found that licensing statements change over time. German et al. [17] found that licensing conflicts occur between the clone siblings, i.e. clones among different systems that come from the same source. Later, German et al. [18] created an automated tool for software license identification, Ninka, which is used in our online clone license analysis.

Reusing of outdated third-party source code occurs in software development. Xia et al. [62] show that a large number of open source systems reuse outdated third-party libraries from popular open source projects. Using the outdated code give detrimental effects to the software since they may introduce vulnerabilities. Our study similarily finds outdated code on Stack Overflow.

The work that is closely similar to us is a study by An et al. [2] where the authors investigated clones between 399 Android apps and Stack Overflow posts. They found 1,226 code snippets which were reused from 68 Android apps. They also observed that there are 1,279 cases of potential license violations. The authors rely on the timestamp to judge whether the code has been copied from/to Stack Overflow along with confirmations from six developers. Instead of Android apps, we investigated clones between Stack Overflow and 111 open source projects. Their results are similar to our findings that there exist clones from software projects to Stack Overflow with potential licensing violations.

6 CONCLUSIONS

Online code clones are clones that have been copied to Q&A websites such as Stack Overflow. We classified 32,533 clone pairs using seven patterns of online code cloning. After automatically removing boiler-plate clones, we manually investigated 3,636 clone pairs. In the 1,216 manually confirmed clone pairs, we discovered 112 clone pairs that have been copied, with evidence, from Qualitas projects to Stack Overflow, 197 clone pairs that have been copied from external sources besides Qualitas to Stack Overflow, and 359 clone pairs that are highly similar but without evidence of copying.

We performed a detailed analysis of the online clone pairs on two aspects: outdated code and licensing violation. The investigation of the 112 clone pairs copied, with evidence, from Qualitas to Stack Overflow reveals that 64 of them are outdated. Moreover, we found 358 code snippets on Stack Overflow that violate the license of their original software.

This study is among, if not the first, to address the important issues of outdated and license-violating online code clones on programming Q&A websites using a hybrid methodology of automatic code clone detection and a manual clone investigation.

REFERENCES

[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.
[2] Le An, Ons Mlouki, Foutse Khomh, and Giuliano Antoniol. 2017. Stack Overflow: A Code Laundering Platform?. In *SANER '17*.
[3] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. 1998. Clone detection using abstract syntax trees. In *ICSM'98*. 368–377.
[4] Nels E. Beckman, Duri Kim, and Jonathan Aldrich. 2011. An Empirical Study of Object Protocols in the Wild. In *ECOOP '11*. 2–26.
[5] Stefan Bellon, Rainer Koschke, Giuliano Antoniol, Jens Krinke, and Ettore Merlo. 2007. Comparison and evaluation of clone detection tools. *Transactions on Software Engineering* 33, 9 (2007), 577–591.
[6] Amiangshu Bosu, Christopher S. Corley, Dustin Heaton, Debarshi Chatterji, Jeffrey C. Carver, and Nicholas A. Kraft. 2013. Building reputation in StackOverflow: An empirical investigation. In *MSR '13*. 89–92.
[7] Steven Burrows, S. M. M. Tahaghoghi, and Justin Zobel. 2007. Efficient plagiarism detection for large code repositories. *Software: Practice and Experience* 37, 2 (2007), 151–175.
[8] Wai Ting Cheung, Sukyoung Ryu, and Sunghun Kim. 2015. Development nature matters: An empirical study of code clones in JavaScript applications. *Empirical Software Engineering* (2015), 517–564.
[9] Morakot Choetkiertikul, Daniel Avery, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2015. Who Will Answer My Question on Stack Overflow?. In *ASWEC '15*. 155–164.
[10] James R. Cordy. 2006. The TXL source transformation language. *Science of Computer Programming* 61, 3 (2006), 190–210.
[11] James R. Cordy and Chanchal K. Roy. 2008. The NiCad Clone Detector. In *ICPC '11*. 3–4.
[12] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. 2013. Software Bertillonage: Determining the provenance of software development artifacts. *Empirical Software Engineering* 18 (2013), 1195–1237.
[13] Massimiliano Di Penta, Daniel M. German, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2010. An exploratory study of the evolution of software licensing. In *ICSE '10*, Vol. 1. 145.
[14] Themistoklis Diamantopoulos and Andreas L. Symeonidis. 2015. Employing source code information to improve question-answering in Stack Overflow. In *MSR '15*. 454–457.
[15] Z. Duric and D. Gasevic. 2012. A Source Code Similarity System for Plagiarism Detection. *Comput. J.* 56, 1 (2012), 70–86.
[16] Marco Funaro, Daniele Braga, Alessandro Campi, and Carlo Ghezzi. 2010. A hybrid approach (syntactic and textual) to clone detection. In *IWSC '10*. 79–80.
[17] Daniel M. German, Massimiliano Di Penta, Yann-Gael Gueheneuc, and Giuliano Antoniol. 2009. Code siblings: Technical and legal implications of copying code between applications. In *MSR '09*. 81–90.
[18] Daniel M. German, Yuki Manabe, and Katsuro Inoue. 2010. A sentence-matching method for automatic license identification of source code files. In *ASE '10*. 437.
[19] Nils Göde and Jan Harder. 2011. Clone Stability. In *CSMR '11*. 65–74.
[20] Nils Göde and Rainer Koschke. 2009. Incremental Clone Detection. In *CSMR '09*. 219–228.
[21] Jan Harder and Nils Göde. 2013. Cloned code: stable code. *Journal of Software: Evolution and Process* 25, 10 (2013), 1063–1088.
[22] Keisuke Hotta, Yukiko Sano, Yoshiki Higo, and Shinji Kusumoto. 2010. Is duplicate code more frequently modified than non-duplicate code in software evolution?. In *IWPSE-EVOL '10*. 73.
[23] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. 2007. DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones. In *ICSE '07*. 96–105.
[24] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. 2002. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *Transactions on Software Engineering* 28, 7 (2002), 654–670.
[25] Cory Kapser and Michael Godfrey. 2006. "Cloning Considered Harmful" Considered Harmful. In *WCRE '06*. 19–28.
[26] Cory Kapser and Michael W Godfrey. 2003. Toward a Taxonomy of Clones in Source Code: A Case Study. In *ELISA '03*. 67–78.
[27] Cory J. Kapser and Michael W. Godfrey. 2008. "Cloning considered harmful" considered harmful: patterns of cloning in software. *Empirical Software Engineering* 13, 6 (2008), 645–692.
[28] Iman Keivanloo, Juergen Rilling, and Ying Zou. 2014. Spotting working code examples. In *ICSE '14*. 664–675.
[29] Raghavan Komondoor and Susan Horwitz. 2001. Using Slicing to Identify Duplication in Source Code. In *SAS'01*. 40–56.
[30] Jens Krinke. 2001. Identifying similar code with program dependence graphs. In *WCRE '01*. 301–309.
[31] Jens Krinke. 2008. Is Cloned Code More Stable than Non-cloned Code?. In *SCAM '08*. 57–66.
[32] Jens Krinke, Nicolas Gold, Yue Jia, and David Binkley. 2010. Cloning and copying between GNOME projects. In *MSR '10*. 98–101.
[33] Zhenmin Li, Shan Lu, Suvda Myagmar, and Yuanyuan Zhou. 2006. CP-Miner: Finding copy-paste and related bugs in large-scale software code. *Transactions on Software Engineering* 32, 3 (2006), 176–192.
[34] Manishankar Mondal, Md. Saidur Rahman, Ripon K Saha, Chanchal K. Roy, Jens Krinke, and Kevin A Schneider. 2011. An Empirical Study of the Impacts of Clones in Software Maintenance. In *ICPC '11*. 242–245.
[35] Dana Movshovitz-Attias, Yair Movshovitz-Attias, Peter Steenkiste, and Christos Faloutsos. 2013. Analysis of the reputation system and user contributions on a question answering website: StackOverflow. In *ASONAM '13*. 886–893.
[36] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. 2012. What makes a good code example?: A study of programming Q&A in StackOverflow. In *ICSM '12*. 25–34.
[37] Cyrus Omar, Young Seok Yoon, Thomas D. LaToza, and Brad A. Myers. 2012. Active code completion. In *ICSE '12*. 859–869.
[38] Jin-woo Park, Mu-Woong Lee, Jong-Won Roh, Seung-won Hwang, and Sunghun Kim. 2014. Surfacing code in the dark: an instant clone search approach. *Knowledge and Information Systems* 41, 3 (2014), 727–759.
[39] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *ICSE '13*. 1295–1298.
[40] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In *MSR '14*. 102–111.
[41] Lutz Prechelt, Guido Malpohl, and Michael Philippsen. 2002. Finding Plagiarisms among a Set of Programs with JPlag. *Journal of Universal Computer Science* 8, 11 (2002), 1016–1038.
[42] Chaiyong Ragkhitwetsagul, Jens Krinke, and David Clark. 2016. Similarity of Source Code in the Presence of Pervasive Modifications. In *SCAM '16*. 117–126.
[43] Chaiyong Ragkhitwetsagul, Matheus Paixao, Manal Adham, Saheed Busari, Jens Krinke, and John H Drake. 2016. Searching for Configurations in Clone Evaluation – A Replication Study. In *SSBSE '16*.
[44] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? A large scale study using Stack Overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223.
[45] Chanchal K. Roy and James R. Cordy. 2008. An Empirical Study of Function Clones in Open Source Software. In *WCRE '08*.
[46] Vaibhav Saini, Hitesh Sajnani, and Cristina Lopes. 2016. Comparing Quality Metrics for Cloned and non cloned Java Methods: A Large Scale Empirical Study. In *ICSE '16*. 256–266.
[47] Hitesh Sajnani, Vaibhav Saini, Jeffrey Svajlenko, Chanchal K Roy, and Cristina V Lopes. 2016. SourcererCC: Scaling Code Clone Detection to Big-Code. In *ICSE '16*. 1157–1168.
[48] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. 2003. Winnowing: Local Algorithms for Document Fingerprinting. In *SIGMOD'03*.
[49] Simon Harris 2015. Simian–Similarity Analyser. http://www.harukizaemon.com/simian. (2015).
[50] Randy Smith and Susan Horwitz. 2009. Detecting and Measuring Similarity in Code Clones. In *IWSC'09*.
[51] Christopher Jon Sprigman. 2015. Oracle v. Google. *Commun. ACM* 58, 5 (2015), 27–29.
[52] Kathryn T Stolee, Sebastian Elbaum, and Daniel Dobos. 2014. Solving the Search for Source Code. *Transactions on Software Engineering and Methodology* 23, 3 (2014), 1–45.
[53] Siddharth Subramanian and Reid Holmes. 2013. Making sense of online code snippets. In *MSR '13*. 85–88.
[54] Jeffrey Svajlenko and Chanchal K. Roy. 2014. Evaluating Modern Clone Detection Tools. In *ICSME'14*. 321–330.
[55] Craig Taube-Schock, Robert J. Walker, and Ian H. Witten. 2011. Can We Avoid High Coupling?. In *ECOOP '11*. 204–228.
[56] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. 2010. Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. In *APSEC '10*. 336–345.
[57] Christoph Treude and Martin P Robillard. 2016. Augmenting API documentation with insights from Stack Overflow. In *ICSE '16*. 392–403.
[58] Bogdan Vasilescu, Alexander Serebrenik, and Mark van den Brand. 2011. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *ICSM '11*. 313–322.
[59] Shaowei Wang, David Lo, and Lingxiao Jiang. 2013. An Empirical Study on Developer Interactions in StackOverflow. In *SAC '13*. 1019–1024.
[60] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. 2014. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. In *ICSME '14*. 291–300.
[61] Tiantian Wang, Mark Harman, Yue Jia, and Jens Krinke. 2013. Searching for Better Configurations: A Rigorous Approach to Clone Evaluation. In *FSE '13*. 455–465.
[62] Pei Xia, Makoto Matsushita, Norihiro Yoshida, and Katsuro Inoue. 2014. Studying Reuse of Out-dated Third-party Code. *Information and Media Technologies* 9, 2 (2014), 155–161.
[63] Di Yang, Aftab Hussain, and Cristina Videira Lopes. 2016. From Query to Usable Code: An Analysis of Stack Overflow Code Snippets. In *MSR '16*. 391–402.