Are We Reusing Outdated and License-violating Code from Stack Overflow?

[1]Chaiyong Ragkhitwetsagul, [1]Jens Krinke, [1]Matheus Paixao, [2]Giuseppe Bianco
[1]University College London, London, UK
[2]Università degli Studi del Molise, Campobasso, Italy

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an *alternate* style which produces a *tighter-looking* paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document *Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

1. INTRODUCTION

Stack Overflow is a popular online programming community with 6.3 million users. It allows programmers to ask questions and give answers to programming problems. The website has found to be useful for software development [40, 41, 14, 28, 38, 53, 54, 14, 58] and also valuable for educational purposes [36]. On Stack Overflow, each conversation contains a question and answer(s). The answers frequently contain at least one code snippet as a solution to the question asked. The code snippet is usually not written directly on Stack Overflow website but copied from another location. It can be copied and modified from the problematic code snippet in the question, copied from an answerer's own code, or borrowed from other locations including open source software (OSS) systems. The process of posting and answering questions on Stack Overflow, which involves copying and pasting 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% 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 have proofs that in some cases clones are not harmful than normal code or even beneficial [47, 25].

Code cloning can also have side effects of violating software licenses and software vulnerabilities. Carelessly cloning code from one project to another project with different license may cause software licensing violation [17]. This also happens within the context of online Q&A website such as Stack Overflow. An et al. [2] showed that 1,279 cloned snippets between Android apps and Stack Overflow have potential of violating software licenses. Security is also among the main concerns when code is copied from online source. Stack Overflow helps developers to solve Android programming problems more quickly than other resources while, at the same time, gives less secure code than books and the official Android documentation [1]. Additionally, outdated and vulnerable third-party libraries from famous open source projects are cloned into in a large number of open source systems [63].

In this study, we treat code snippets that are copied from software systems to Stack Overflow, and vice versa, as code clones. We call them **online code clones** (or sometimes only call **clones** for brevity). There are four ways to create online code clones: (1) code is cloned from a software project to Stack Overflow as an example; (2) code is cloned from Stack Overflow to a software project to obtain a functionality, perform a particular task, or fixing a bug; (3) code is implicitly cloned from one software project to another by having Stack Overflow as a medium; and (4) code is cloned from an external source to both a software project and Stack Overflow. **Online code clones can similarly lead to a problem of outdated code, bug propagation, licensing violation, and software vulnerability as classical code clones.** Unfortunately, they are more difficult to locate and fix since the search space in online code corpora is larger and no longer confined in a local repository.

A motivating example of outdated code in online code clones can be found in the a Stack Overflow post regarding how to implement *RawComparator* in Hadoop[1]. In Figure 1, the left hand side shows a code snippet embedded as a part of

[1]http://stackoverflow.com/questions/22262310

ESEC/FSE 2017 September 4–8, 2017, Paderborn, Germany

DOI: 10.1145/1235

```
/* Code in Stack Overflow #22315734 */
public int compare (byte[] b1,int s1,int l1,
                    byte[] b2,int s2,int l2) {
  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,
                   byte[] b2,int s2,int l2) {
  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 same code fragments, WritableComparator.java. The one on Stack Overflow post 22315734 is outdated when compared to its latest version in Hadoop code base

accepted answer to the question. The snippet shows how Hadoop implements *compare* method in its *WritableComparator* class. The code snippet on the right hand side shows another version of the same *compare* method in *WritableComparator* class but it is extracted from the latest version of Hadoop. We can obviously see that they are highly similar except one line, `buffer.reset(null,0,0);`, is added in the latest version after `key2.readFields(buffer);`. The added line is intended for cleaning up the reference in `buffer` variable. While this change has already been introduced into *compare* method in the latest version of Hadoop, the code example in Stack Overflow post is still unchanged. This example shows that there can be inconsistencies between online code clones and its original. The code snippet on Stack Overflow can be outdated.

While a lot of research has focused on reusing code snippets "from Stack Overflow" (e.g. [28, 2, 64]), less work has been conducted on finding the origins of code examples copied "to Stack Overflow" and effects of reusing them such as outdated code and software licensing violation.

To fill this gap, this paper makes the following primary contributions:

1. A manual study of online code clones: We used two clone detection tools to discover 270,429,422 similar code fragment pairs and manually investigated 3,450 candidate clone pairs between Java code fragments obtained from Stack Overflow accepted answers and 111 Java open source projects.

2. Addressing the problems of reusing source code between open source projects and Stack Overflow: Our study shows that there are at least 523 clones that have been obviously copied from open source projects or external online sources to Stack Overflow as code examples which potentially violate their software licenses. Importantly, 53 out of the 96 online code clones on Stack Overflow are outdated and questionable for being reused.

3. Online code clone oracle: The 3,450 manually investigated and validated online clone pairs are available for download on the study website[2] and can be used as a clone oracle.

2. EMPIRICAL STUDY

We performed an empirical study of online code clones between Stack Overflow and 109 Java open source projects to answer the following research questions:

[2]https://some.where

RQ1 (online code clones): *To what extent source code is 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 according to our online code cloning pattern scheme. This gave us more 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 correctness of Stack Overflow code examples since millions of users are potentially reusing them.

RQ4 (Software licensing violation): *Do licensing conflicts occur between Stack Overflow clones and their originals?* We investigated whether online code clones can be legally harmful to software reusing them.

2.1 Experimental Framework

To answer the four research questions, we followed the experimental framework depicted in Figure 2. We processed two datasets, Stack Overflow and 111 open source projects from Qualitas corpus. Java code fragments were extracted from Stack Overflow posts using regular expressions. We prepared Java code in both datasets by removing comments and pretty-printing to increase accuracy of clone detection. Then, we deployed two clone detection tools, Simian [50] and NiCad [46, 11], to locate clones between the two datasets. Due to scalability of Simian and NiCad, we partitioned the input and run the tools multiple times. Each run was composed of the whole Stack Overflow data and single Qualitas project. We repeated the process until we cover 111 projects.

We then converted the clone reports to General Clone Format (GCF) [62] and combined the 111 reports into a single GCF file. GCF provides a common format for clones which enabled us to reuse the same scripts to analyse clone reports from both Simian and NiCad. Moreover, using GCF, other additional clone detectors could be adopted, if needed, without any changes in the analysis. Simian did not provide an option to detect inter clones between two locations. Hence the Simian GCF clone report was pruned to contain only inter clone pairs between Stack Overflow and Qualitas projects. In this step, all intra clone pairs within either Stack Overflow or Qualitas were removed. NiCad provided an option to detect inter clones so no pruning is needed. We did

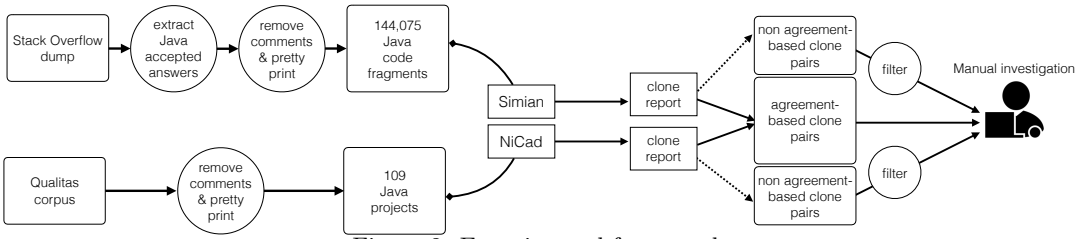


Figure 2: Experimental framework

not have an oracle of clones between the two data sets so we rely on a manual investigation to validate the reported clone candidates. However, the large amount of clone pairs hindered us from looking at all of them. Random sampling of clones could be done but may return mostly non-meaningful clones. Instead, we selected clone pair candidates by relying on agreement of clone detectors. If a clone pair was similarly reported by multiple tools, we had a higher confidence that it was a real clone pair. To achieve this, clone pairs from the two clone detectors were pair-wise matched to find agreements using Bellon's clone overlapping criteria [5]. This step generated **agreed clone pairs**. They were pairs with high confidence to be true clones since they obtained agreement from both tools. Then, pairs reported by Simian and NiCad that did not find agreement were **disagreed clone pairs**. The disagreed clone pairs were clones with less confidence than the agreed ones. Finally, agreed and disagreed clone pairs had been looked at manually by the first and the third author.

In the manual inspection process, we classified clones into seven categories according to information observed from code comments and natural text in Stack Overflow answer. This process took approximately a months until we successfully investigated 3,450 clone pairs. Clone pairs that were classified as false clones, boiler-plate code, or IDE-generated were not very interesting and were ignore in the analysis of outdated code and licensing violation. We compared licensing information of 523 remaining clone pairs for possibility of software licensing violations. Moreover, we looked forward through history of each clone from its git repository. With this method, we discovered **outdated code**, code that changes are made to them after it had been copied to Stack Overflow.

2.2 Experimental Setup

2.2.1 Datasets

Stack Overflow: we extracted Java code snippets from a snapshot of Stack Overflow dump [3] in January 2016. The archived dump has a size of 9 gigabytes. The data dump was in XML format containing information of *Posts* (questions and answers) and supporting data such as user accounts and timestamps of the posts. We were interested in code snippets embedded in posts which were located between `<code></code>` tags. A Stack Overflow thread contains a post of question and several posts of answers. Every answer has voting score from its users. 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 were only interested in code snippets with at least six lines. Snippets which were smaller than six lines were usually spurious clones [5]. Second, we only focused on code snippets in accepted answers. We chose the snippets in accept answers because they actually solved the problems in the questions and had a check mark sign on them to notify other users. Moreover, they were always displayed just below the questions which made them attractive to be reused than other answers. Each snippet was extracted from the dump using regular expressions and saved to a file using its post ID as the file's name. If an accepted answer had more than one code snippet, we appended an indexing number, starting from zero, after the post ID (e.g. 45051109_0, and 45051109_1.). Lastly, we added `.java` extension to the file's names so that clone detectors could recognise them. We finally obtained 144,075 Java code snippets which contain 2,347,269 lines of Java source code excluding comments and blank lines[4].

Table 1: Stack Overflow and Qualitas datasets

Dataset	No. of files	SLOC
Stack Overflow	144,075	2,347,269
Qualitas	160,937	19,086,883

Open source systems: we selected an established corpus for an empirical software engineering study called **Qualitas** [57] for this study. It is a curated Java corpus that has been used in several software engineering studies [56, 4, 59, 37]. The projects in the corpus represent various domains of software systems ranging from programming language to 3D and visualisation [57]. We selected the 20130901r release of Qualitas corpus containing 112 Java open source projects. This release contains projects with releases no later than 1st September 2013. We chose a snapshot late back in 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 at least 3 years old from the time of the experiment (January to December 2016). The time difference is sufficiently long for a number of code snippets to be copied onto Stack Overflow. Out of 112 Qualitas projects, there is one project, *jre*, that does not contain Java source code due to its licensing limitation [57] so it is removed from the study. This resulted in total of 111 projects analysed in the study. As shown in Table 1, the 111 Qualitas project have 160,937 Java files containing 19,086,883 lines of code.

2.2.2 Clone Detectors

There is a number of restrictions in terms of choosing clone detection tools for this study. First, they have to support Java. Second, due to nature of code snippets posted on Stack Overflow, some of them are not complete Java classes or methods. Hence, the tool must be flexible enough to

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

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

process code snippets that are neither a complete block nor compilable. Third, since the amount of code that have to be processed are in a scale of millions line of code (as shown in Table 1), a clone detector must be scalable enough to successfully complete the execution and report clones in a reasonable amount of time. We have tried running 5 state-of-the-art clone detectors including Simian [50], NiCad [11, 46], CCFinder [24], iClones [20], and DECKARD [23] against Stack Overflow and Qualitas datasets. CCFinder, iClones, and DECKARD failed to report clones between 144,075 Stack Overflow code snippets and 111 Qualitas projects. All of them reported execution errors after running for couple of hours. Thus, we removed them from the study. Simian and NiCad completed the detection with success. We found that both of them are also flexible enough to handle code with incomplete methods or classes.

Simian is a text-based clone detector which locate clones at line-level granularity and has been used extensively in several clone studies [43, 62, 34, 8, 32]. It is a command-line tool which enables us to automate the detection. Furthermore, it offers normalisation of variable names and literals (strings, and numbers) which enable 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/removed/relocated/changed statements) and is also used in several empirical clone studies [46, 43, 55, 62, 34, 48]. It utilises TXL [10] for parsing and pretty-printing source code. It also provide 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. NiCad is also a command-line tool which makes it suitable for automation.

2.2.3 Prioritise Clone Candidates Using Agreement

A number of clones detected in large-scale datasets can be huge. In our study, there are totally 266,837,480 clone pairs reported. It is almost infeasible for human to manually validate them all. One can do a random sampling of clones. However, due to a very high number of false positives, they may end up looking at most of the false clones. Therefore, we adopted an idea of **clone agreement** which has been used in clone research studies [62, 16, 44] in a situation that clone oracle is missing or impossible to establish. Clone pairs agreed by multiple clone detection tools have higher confident to be real clones [44]. By using this agreement-based approach, we could reduce the number of clone candidates for manual investigation by paying more attention to the ones agreed by multiple tools. To find an agreement between two clone pairs report by two different tools, we used clone pair matching metric proposed by Bellon et al. [5]. Two clone pairs which have large enough overlapping clone lines can be categorised as either a good-match or an ok-match pair (denoted as *good* and *ok* respectively) with a confident value between 0 and 1. A *good* clone pair has stronger agreement than an *ok* pair. We follow the following definitions of good- and ok-match introduced in the original paper.

Given that a clone pair CP is formed by two clone fragments CF_1 and CF_2 with a pre-defined similarity threshold t, i.e. $CP = (CF_1, CF_2, t)$, we can define *overlap* and *contained* value of two clone pairs as

$$overlap(CP_1, CP_2) = \frac{|lines(CF_1) \cap lines(CF_2)|}{|lines(CF_1) \cup lines(CF_2)|} \quad (1)$$

$$contained(CP_1, CP_2) = \frac{|lines(CF_1) \cap lines(CF_2)|}{|lines(CF_1)|}. \quad (2)$$

good-value of two clone pairs is then defined as

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

ok-value is defined as

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

Two clone pairs CP_1 and CP_2 are called a *good(p)* iff, for $p \in [0, 1]$ holds

$$good(CP_1, CP_2) \geq p. \quad (3)$$

Similarly for an *ok-match(p)* pair

$$ok(CP_1, CP_2) \geq p. \quad (4)$$

Using this good and ok-match criteria with a predefined threshold p, we can prune the 266-million candidate clone pairs for manual investigation. *good* pairs are the ones with the highest confident and ranked the first to be looked at, followed by *ok* pairs, and followed by clone pairs without agreement.

2.3 Clone Detector's Parameter Tuning

We were aware of effects of configurations to clone detection results and the importance of searching for optimised configurations in empirical clone studies [61, 44, 43, 55]. However, considering the massive size of the two datasets and search space of at least 15 Simian's and 5 NiCad's parameters, we were hindered from searching 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*, a study of optimising clone detectors' configurations based on clone agreement, by Wang et al. [62] (denoted by E). The details of the two configurations are described in Table 2. Having two clone detectors, Simian (denoted as S) and NiCad (denoted as N), with two chosen configurations each, we looked for Bellon's good and ok-match in four possible pair-wise combinations: S_DN_D, S_DN_E, S_EN_D, and S_EN_E.

3. RESULTS AND DISCUSSION

We followed the experimental framework and detected clones between Stack Overflow and Qualitas corpus using the two selected clone detectors. To answer RQ1, we computed statistics of clone discovered by the tools. To answer RQ2, we manually investigated online clone pair candidates and categorised them using our patterns of online code cloning. The patterns were derived from Kapser et al. [26] augmented by common patterns found in our data set. For RQ3, we

Table 2: Configurations of Simian and NiCad

Tool	Parameters
S_D	threshold=6, ignoreStringCase, ignoreCharacterCase, ignoreModifiers
S_E	threshold=5, ignoreIdentifiers, ignoreIdentifierCase, ignoreStrings, ignoreCharacters, ignoreSubtypeNames, balanceSquareBrackets
N_D	MinLine=10, MaxLine=1000, UPI=0.30
N_E	MinLine=5, MaxLine=604, UPI=0.20, blind renaming, literal abstraction

Table 3: Statistics of clones found between Stack Overflow and Qualitas projects using Simian (S) and NiCad (N) with default (D) and EvaClone (E) configurations

Stats	S_D	S_E	N_D	N_E
Snippets	1,086	1,530	1,240	12,886
Total C_{pairs}	67,570	63,372,599	229,176	206,760,077
Avg. C_{pairs}	62	41,447	185	16,047
Avg. C_{size}	7.72	4.79	9.64	5.32
Avg. $C_{\%}$	29%	28%	25%	21%

looked at the true positive clone pairs and check if they are still up-to-date. Similarly, for RQ4, we also looked at the license of each clone and observed a possibility of licensing violation.

3.1 RQ1: Online Code Clones

The clone statistics obtained from running Simian and NiCad with D and E configurations are presented in Table 3. From Table 3, Simian clones cover approximately 10% of the 144,075 Stack Overflow snippets, 1,086 reported by S_D and 1,530 from S_E respectively. N_D reports clones in 1,240 Stack Overflow code snippets, while N_E reports clones in a larger number of 12,886 snippets mainly due to its relaxed configurations. In terms of number of clone pairs, S_E and N_E report a large number of clone pairs of 63,372,599 and 206,760,077 respectively. This is expected since EvaClone configurations prefer recall so it tends to remote more clones [62]. The average clone size ofS_D is 7.72 lines which is bigger than its S_E counterpart of 4.79. Similarly, N_D has an average clone size of 9.64 lines which is bigger than 5.32 reported by N_E. We can see from the statistics that EvaClone tunes the tools in the way that they report smaller clones. The average percentage of Stack Overflow code snippets that are cloned according toS_D, S_E, N_D, and N_E is 29%, 28%, 25%, and 21% accordingly. Preliminary manual investigation of Simian's clone report revealed that there were problematic 11 fragments. These 11 fragments triggered Simian to generate large clone clusters containing a huge number of false clones from array initialisation. Hence, their clones were removed from further analyses.

As displayed in Table 4, after we applying clone agreement filtering using Bellon's criteria, the number of *good* agreed clone pairs is 2,261 while *ok* is 26,838. The number of disagreed clone pairs (S_D, N_D) after filtering is 17,801 and 168 for Simian and NiCad respectively. The details are as follows.

3.1.1 Agreed clone pairs

Agreed clone pairs are clone pairs that pass Bellon's good- or ok-match criteria and selected for manual investigation. Similar to the original study, we select a threshold p of 0.7 for both *good* and *ok* pairs [5]. We encountered NiCad failures with a few Qualitas projects. N_D could not detect clones in Hibernate due to clustering errors. N_E generated renaming errors for 5 projects including Vuze, Hibernate, Myfaces, Netbeans, and Spring. We had contacted the author of NiCad regarding the issues. They found that it was a problem of long file paths which will be fixed in the next NiCad releases. As a result, we did not have NiCad clones from these project in the agreed clone pairs. Simian clones of the same projects also disappeared from the agreed clone pairs since they could not find matches.

The distributions of *good* clone pairs between four combinations of D and E configurations are listed in Table 6 and depicted visually in Figure 3. There are 2,261 unique *good* pairs consisting of 10 pairs from S_DN_D, 26 pairs from S_DN_E, 10 pairs from S_EN_D, and 2,228 pairs from S_EN_E. There are 26,838 unique *ok* pairs (excluding the subsumed 2,261 *good* pairs).[5] We obtained 9,052; 991; 78; and 17,549 pairs from S_DN_D, S_DN_E, S_EN_D, and S_EN_E respectively. Between the four configuration sets, there are considerable amount of clone pairs shared between two adjacent sets (as depicted in Figure 4), but there is no clone pair that is agreed by all four combinations.

3.1.2 Disagreed clone pairs

The disagreed clone pairs are clone pairs that are reported by a tool, either Simian or NiCad, and do not have agreement with clones from another tool. The disagreement can be from misalignment of clone lines or different configurations that result in dissimilar clones reported. They also cover clone pairs in projects having NiCad's errors (6 projects for N_D and 27 for N_E) that are automatically missing from the agreed clone pairs. With the four configuration combinations, we decided to investigate only S_D, and N_D, and drop S_E and N_E due to their enormous amount of clone pairs (59 and 206 millions respectively). S_E and N_E also have a high possibility of containing a large number of false positives due to relaxation of their EvaClone configurations.

Even choosing only the default configurations, the number of clone pair candidates are still very large. We hence apply three filters: **clone size, regular expression, and similarity threshold**. For the clone size filter, we raise the minimum clone size to 10 line as larger clones tend to be more interesting while smaller ones tend to be false clones [47]. The 10–line threshold is already the default configuration for NiCad, thus this filter affects Simian clone pairs only (Simian's default configurations consider a minimum of 6 lines). The regular expression filter captures uninteresting boiler-plate clones of `hashCode()` methods, getters and setters using regular expressions and removes them. The third filter, similarity threshold, applies only to NiCad clone pairs since Simian does not provide this similarity threshold configuration.

As shown in Table 4, for S_D we filtered the results using clone size and regular expressions. The combination of the two filters reduce the number of S_D clones to 17,801 pairs for manual investigation. For N_D, 10-line threshold is already a minimum NiCad's clone size. However, NiCad still reported much more clones than Simian since it can detect type-3 clones. Thus, we filtered NiCad's clones by using similarity threshold and regular expressions. We increase NiCad's similarity threshold from 70% to 80% (by adjusting NiCad's

[5]According to the definition, *ok* clone pairs always subsume the *good* pairs.

Table 4: The number of online clone pair condidates (mutually exclusive) before and after filtering. The last column shows the number of true positive clone pairs after the manual investigation.

Clone set	Before Filters	Filters*				Candidates	`equals()`	Manual	TP
		L	*S*	*R*	*G*				
good						2,261	9	2,252	72
ok						9,322	9,122	200	9,240
S_D	67,570	✓		✓	✓	17,801	16,930	871	704
N_D	229,176		✓	✓	✓	168	41	127	116
Total	296,746					47,068	26,102	3,450	10,132

Filters*: L=$Line \geq 10$, S=$Sim \geq 80\%$, R=regular expressions, and G=$good/ok$ pairs.

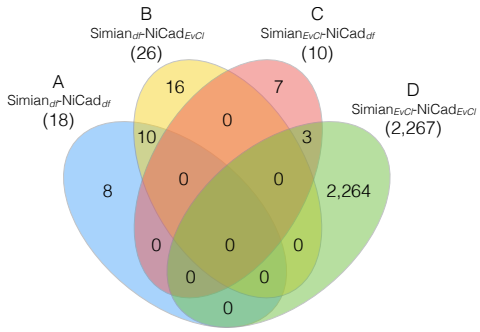


Figure 3: Distributions of *good* pairs

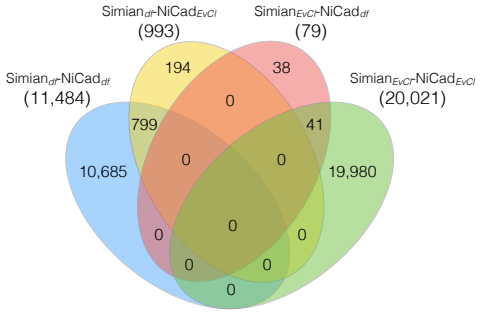


Figure 4: Distributions of *ok* pairs

Table 5: No. of successfully analysed Qualitas projects

Execution	S_D	S_E	N_D	N_E
Successful	111	111	100	96
Clustering fail	–	–	1	–
Renaming fail	–	–	–	5

Table 6: Distribution of agreed clone pairs

Set	S_DN_D	S_DN_E	S_EN_D	S_EN_E	Total	Unique
good	10	26	10	2,228	2,321	2,308
ok	9,042	991	78	17,549	27,670	26,838

UPI from 0.3 to 0.2). We have tried varying the UPI from 70%, 80% and 90% and we found that 80% similarity provides the best balance in terms of the amount of reported clones and feasibility for manual investigation. Using the two filters, we reduced the size of N_D clones to 168 pairs.

In our manual investigation process, we used a script to automatically classify `equals()` clone pairs which helped classifying 26,102 pairs out of 47,068 (more than half of the `equals()` pairs came from S_D). Finally, there are 3,450 pairs remaining which we manually investigated them. **To answer RQ1, we found 47,068 clone candidates between 111 Qualitas projects and 144,075 Stack Overflow snippets. They are 2,261 *good* pairs, 26,838 *ok* pairs. XXX After a manual investigation, we found at least 523 online code clones pairs are true clones.**

3.2 RQ2: Patterns of Online Code Cloning

Table 7: Seven patterns of online code cloning

Patt.	Descriptions
A	Cloned from Qualitas project to Stack Overflow (Q → S).
A'	Cloned from Stack Overflow to Qualitas project (S → Q).
B	Cloned from each other or from an external source X outside the project (unknown) (S ↔ Q ∨ (X → S ∧ X → Q)).
C	Cloned from an external source (X → S ∧ X → Q).
D	Boiler-plate or IDE auto-generated
E	Inheritance, interface implementation
F	Accidental similarity, false clones by normalisation

Besides the quantitative analysis, we are also interested in a qualitative analysis of the online clones. Why are they created and how? Due to an absence of clone oracle of the two data sets, we resort to manual investigation to validate the clone pair candidates as either true or false positives. At the same time, we select a reason that stimulates a creation of each online clone pair.

We started by studying the 8 patterns of cloning from Kapser et al. [25, 27]. We use the patterns to classify 697 clone pairs associated with 34 sampled Stack Overflow fragments. This preliminary investigation aims to evaluate the applicability of Kapser's cloning patterns to our study. UsingS_D clone report, fragments are ranked according to (1) frequency, (2) popularity (i.e. number of associated Qualitas projects), (3) clone size in SLOC, and (4) clone percentage compared to the fragment size. We then picked the top 10 fragments from the 4 groups resulting in 34 Stack Overflow fragments chosen (there were some overlaps). The 34 selected snippets generate 697 clone pairs. Using Kapser's cloning patterns, the 697 clone pairs were categorised into either Customisation or Templating. Clearly **Kapser's cloning patterns are too broad for our study and a more suitable and fine-grained classification scheme is needed.**

We adopted one of Kapser's cloning patterns, boiler-plate code and API/library protocol as category D, and added 6 new patterns observed as common cloning patterns from our preliminary investigation. The seven patterns ranging from A–F as presented in Table 7. Pattern A is a cloning that has evidence to be copied from Qualitas to Stack Overflow (by having comments in source code and explanation/links in Stack Overflow post). Pattern A' is the opposite of A. Pattern B is a cloning that is exactly identical or highly similar but without any attribution of copying. Pattern C is a cloning that has an evidence of copying from the same external source. Pattern D is either boiler-plate code (e.g. `equals()` methods, or getters and setters) or IDE-generated (e.g. GUI initialisation). Pattern E is inheritance of the same super

Table 8: Classification results of agreed and disagreed clone pairs.

Set	A	A'	B	C	D	E	F	Total
good	1	0	4	3	58	6	2,189	2,261
ok	8	0	29	10	9,158	35	82	9,322
S_D	79	0	336	17	209	63	167	871
N_D	8	0	27	1	77	3	82	168
Total	96	0	396	31	9,502	107	2,520	12,622

class or implementation of the same interface. They usually share similar overriding methods. The last pattern, F, is a false clone pair. They can be either accidentally similar (e.g. similar try-catch statements) or similar after code normalisation process. Using this cloning pattern classification scheme, we consider clone pairs in created by pattern A, A', B, C, D, and E as true positive, and F as false positive.

3.2.1 Manual Investigation of Clone Candidates

The first author who has been working on clone detection research for two years takes the role of a main investigator performing manual investigation of all 3,450 clone pairs. The third author takes the role of a second investigator performing 10% validation of the clone pairs judged by the main investigator. Following the cloning patterns, the main investigator manually goes through each agreed clone pair, looks at the clones, and chooses the most appropriate cloning pattern for the pair. A relevant and useful observation is also recorded for each clone pair.

Agreed clone pairs: the classification results are shown in Table 8. We have manually investigated 2,261 *good* clone pairs. There is one clone pair in pattern A which is found to be copied from a Qualitas project to Stack Overflow. Four pairs are highly similar or identical but without any evidence of copying (no comments in neither Stack Overflow post nor Qualitas source code) and are classified as pattern B. We observed 3 pairs in pattern C since they are copied from external sources. 58 clone pairs are found to be pattern D: `equals()` methods, or getters and setters. Six pairs are similar code from inheriting the same superclass or implementing the same interface, pattern E. Finally, 2,189 clone pairs are categorised to F which means they are false clones.

For the *ok* clone pairs, we could not feasibly investigate all 23,868 pairs manually. According to the manual investigation of *good* pairs, we found that $S_E N_E$ produces a large number of 2,253 false positive results (accounts for 99.87% of $S_E N_E$ clone pairs) due to its recall preference. We thus decided to leave this configuration out of the manual investigation. There are totally 4,625 *ok* pairs that were investigated. We found 8, 29, and 10 in pattern A, B, and C respectively. There were 77 boilter-plate pairs, and 3 inherit/interface pairs. 82 pairs were false positives.

Disagreed clone pairs: we performed a manual investigation and classification of 871 S_D and 168 N_D clone pairs in the same way as the agreed ones. There were 19 pairs in pattern A, 336 pairs in B, 17 pairs in C, 209 in D, and 63 in E. 167 pairs are false clones. For N_D, there are 8 pairs in pattern A, 27 pairs in B, 1 pair in C, 77 pairs in D, and 3 pairs in E. N_D reported 82 false positive clones.

3.3 RQ3: Outdated Online Code Clones

Outdated code occurs when a piece of code has been copied from its origin to another location and later the original code has been updated [63]. Usually code clone detection is used to locate clone instances and update them according to their updated original code [5]. However, in this situation, the clones are more pervasive on Stack Overflow posts and more difficult to detect due to its large scale and the mix of natural language and source code. Since code can be updated due to various possible reasons, e.g. bug fixing or API changes, it is risky if developers reuse outdated code snippets from Stack Overflow. They might later find out that the copied code does not work any more due to different API versions. Even worse, they might also introduce bugs or vulnerabilities into their software.

To check if the discovered Stack Overflow clones are outdated, we focused on the 96 pattern-A clone pairs that are copied in the direction from Qualitas to Stack Overflow (see Table 8). For each code snippet, we could track its origin from its Qualitas counterpart. Relying on git repositories of Qualitas projects, we located the latest version of the each file and compared to the Stack Overflow snippet to see if any change has been made to the source code. If we found any change, we used *git blame* command to see who modified the source code and when.

Figure 6 shows the findings of outdated clone investigation. The comparison with the clones' latest versions revealed that 55 clone pairs were outdated. These were clone pairs that were copied from Qualitas to Stack Overflow and later marked as accepted answers. Hadoop and Spring have the maximum number of 9 outdate code clones, followed by 6 clones from Tomcat, and 5 clones from Eclipse-SDK. An example of outdated code in hadoop's *WritableComparator.java* is shown in Figure 1. The newest version has one line added. We also found outdated code which contained heavy modifications. For example, the code snippet in Stack Overflow post 23520731, a copy of *SchemaUpdate.java* in Hibernate, it had been heavily modified in the latest version (as shown in Figure 5). Moreover, the code snippet in Stack Overflow post 3758110, a copy of *DefaultAnnotationHandlerMapping.java* in Spring, was deleted in the commit `02a4473c62d8240837bec297f0a1f3cb67ef8a7b` by Chris Beams on 20th January 2012 at 22:51:02 two years after it was posted. The latest version of Spring does not contain the file anymore.

The outdate online code clones can cause problems ranging from uncompilable code due to different API usage to bug propagation. An outdated code with a subtle change (e.g. Figure 1) may be copied and reused without awareness from developers. Stack Overflow has a voting mechanism that may mitigate this outdated code issue. Usually when developers reuse a code snippet from a Stack Overflow post and find that it does not work nor compatible with their environment. They can cast a down vote to that answer resulting in low votes for the answer (which might be outdated). However, if the answer is marked as accepted by the person who asks the question (which is our case), it is always ranked on the top and still attractive to naive developers who are ignorant.

3.4 RQ4: Software Licensing Violation

Software licensing is important in software development. Violation of software license can cause major impacts to software delivery and also lead to legal issues [52]. Software licenses keep evolving in open source projects [13] and clones can have different license [17]. This can lead to licensing

Table 9: 58 code clones in Stack Overflow (SO) that were altered, rewritten, or removed from the project after posted and their respective licenses. Files can be changed (C) by modifications (M), deletion (D), and rewriting (R).

No.	Project	File	Start	End	License	SO Post	License	C	C_{date}
1	aspectj-1.6.9	aspectjtools/../Agent.java	7	18	–	18303692	–	M	2015-09-08
2	aspectj-1.6.9	aspectjweaver/../Agent.java	7	18	–	18303692		M	2015-09-08
3	eclipse-SDK	GenerateToStringAction.java	113	166	EPLv1	2513183	EPLv1	M	2015-03-17
4	eclipse-SDK	GenerateToStringAction.java	117	126	EPLv1	2513183	EPLv1	M	2015-03-17
5	eclipse-SDK	GenerateToStringAction.java	143	165	EPLv1	2513183	EPLv1	M	2015-03-17
6	eclipse-SDK	GenerateToStringAction.java	178	187	EPLv1	2513183	EPLv1	M	2011-03-01
7	eclipse-SDK	WizardDialog.java	377	394	EPLv1	11861598	–	M	2011-02-03
8	hadoop-1.0.0	DBCountPageView.java	275	287	Apache-2	21702608	–	M	2011-06-12
9	hadoop-1.0.0	DBCountPageView.java	289	309	Apache-2	21702608	–	M	2011-06-12
10	hadoop-1.0.0	JobSubmissionFiles.java	46	55	Apache-2	14845581	–	M	2012-06-25
11	hadoop-1.0.0	mapred/../LineRecordReader.java	47	60	Apache-2	16180910	–	M	2011-07-25
12	hadoop-1.0.0	mapreduce/../LineRecordReader.java	75	99	Apache-2	16180910	–	M	2011-07-25
13	hadoop-1.0.0	StringUtils.java	40	56	Apache-2	801987	–	M	2013-02-04
14	hadoop-1.0.0	TestJobCounters.java	186	192	Apache-2	18833798	–	M	2011-06-12
15	hadoop-1.0.0	TextOutputFormat.java	75	99	Apache-2	16928749	–	M	2011-06-12
16	hadoop-1.0.0	WritableComparator.java	44	54	Apache-2	22315734	–	M	2014-11-20
17	hibernate-4.2.2	ConnectionProviderInitiator.java	65	93	–	15168494	–	M	2012-06-24
18	hibernate-4.2.2	Example.java	224	243	–	24924255	–	M	2013-04-23
19	hibernate-4.2.2	SchemaUpdate.java	115	168	–	23520731	–	M	2016-02-05
20	hibernate-4.2.2	SettingsFactory.java	244	255	–	8257554	–	D	2011-03-11
21	hibernate-4.2.2	SQLServer2005LimitHandler.java	43	61	–	23967852	–	M	2015-03-12
22	jasperreports-3.7.4	JRVerifier.java	982	998	LGPLv3+	8037824	–	M	2008-04-17
23	jasperreports-3.7.4	JRVerifier.java	1221	1240	LGPLv3+	8037824	–	M	2011-05-20
24	jfreechart-1.0.13	AbstractXYItemRenderer.java	532	569	LGPLv2.1+	12936580	–	M	2016-02-19
25	jfreechart-1.0.13	KeyToGroupMap.java	18	30	LGPLv2.1+	16058183	–	M	2013-07-03
26	jfreechart-1.0.13	SpiderWebPlot.java	502	520	LGPLv2.1+	21998949	–	M	2008-06-02
27	jfreechart-1.0.13	SpiderWebPlot.java	522	536	LGPLv2.1+	21998949	–	M	2008-06-02
28	jgraph-5.13.0.0	HelloWorld.java	16	22	LGPLv2.1+	6722760	–	R	2014-04-13
29	jgraph-5.13.0.0	HelloWorld.java	28	40	LGPLv2.1+	6722760	–	R	2014-04-13
30	jgraph-5.13.0.0	HelloWorld.java	31	36	LGPLv2.1+	6722760	–	R	2014-04-13
31	jgraph-5.13.0.0	HelloWorld.java	39	56	LGPLv2.1+	6722760	–	R	2014-04-13
32	jstock-1.0.7c	GoogleMail.java	18	42	GPLv2+	14940863	–	M	2015-12-13
33	jstock-1.0.7c	GoogleMail.java	18	42	GPLv2+	24680923	–	M	2015-12-13
34	jung2-2_0_1	ShortestPathDemo.java	106	117	–	6025026	–	M	2010-04-13
35	jung2-2_0_1	ShortestPathDemo.java	158	172	–	6025026	–	M	2010-04-13
36	junit-4	Assert.java	33	52	–	23586872	–	M	2015-05-12
37	junit-4	ExternalResource.java	4	23	–	7504040	–	M	2016-06-25
38	junit-4.11	ExpectException.java	11	29	–	8802082	–	M	2014-05-26
39	poi-3.6	WorkbookFactory.java	18	28	Apache-2	12593810	–	M	2015-04-29
40	spring-3.0.5	AnnotationMethodHandler ExceptionResolver.java	224	233	Apache-2	5660519	–	D	2012-01-20
41	spring-3.0.5	AutowireUtils.java	32	42	Apache-2	20913543	–	M	2014-10-28
42	spring-3.0.5	CustomCollectionEditor.java	33	71	Apache-2	18623736	–	M	2013-11-21
43	spring-3.0.5	DefaultAnnotation HandlerMapping.java	78	92	Apache-2	3758110	–	D	2012-01-20
44	spring-3.0.5	DefaultPropertiesPersister.java	69	80	Apache-2	6149818	–	M	2013-03-19
45	spring-3.0.5	org.springframework.test/../ DelegatingServletInputStream.java	6	20	Apache-2	20996373	–	M	2016-07-15
46	spring-3.0.5	org.springframework.web/../ DelegatingServletInputStream.java	6	20	Apache-2	20996373	–	M	2008-12-18
47	spring-3.0.5	org.springframework.web.servlet/../ DelegatingServletInputStream.java	6	20	Apache-2	20996373	–	M	2008-12-18
48	spring-3.0.5	DispatcherServlet.java	91	103	Apache-2	4781746	–	M	2011-08-08
49	spring-3.0.5	Jaxb2Marshaller.java	253	269	Apache-2	10924700	–	M	2012-08-28
50	spring-3.0.5	ScheduledTasksBean DefinitionParser.java	42	52	Apache-2	3751463	–	M	2016-07-05
51	struts2-2.2.1	DefaultActionMapper.java	91	103	Apache-2	14019840	–	M	2013-10-18
52	tomcat-7.0.2	BasicAuthenticator.java	25	73	Apache-2	21734562	–	M	2016-08-04
53	tomcat-7.0.2	BasicAuthenticator.java	33	43	Apache-2	21734562	–	M	2016-08-04
54	tomcat-7.0.2	CoyoteAdapter.java	543	553	Apache-2	24404964	–	M	2012-11-18
55	tomcat-7.0.2	CoyoteAdapter.java	557	573	Apache-2	24404964	–	M	2012-11-18
56	tomcat-7.0.2	FormAuthenticator.java	51	61	Apache-2	21734562	–	M	2016-08-04
57	tomcat-7.0.2	HttpServlet.java	111	124	Apache-2	5266856	–	M	2011-10-22
58	tomcat-7.0.2	JspRuntimeLibrary.java	252	296	Apache-2	10289462	–	M	2012-09-12

```
/* Code in Stack Overflow #23520731 */
public void execute (Target target) {
  LOG.runningHbm2ddlSchemaUpdate();
  Connection connection = null;
  Statement stmt = null;
  Writer outputFileWriter = null;
  exceptions.clear();
  try {
    DatabaseMetadata meta;
    ...
```

```
/* SchemaUpdate.java (2016-09-26) */
public void execute(EnumSet<TargetType> targetTypes,
                    Metadata metadata, ServiceRegistry serviceRegistry) {
  if ( targetTypes.isEmpty() ) {
    LOG.debug(""Skipping SchemaExport as no targets were specified"");
    return;
  }
  exceptions.clear();
  LOG.runningHbm2ddlSchemaUpdate();
  ...
```

Figure 5: Outdated code fragment on Stack Overflow post 23520731. This code has been copied from SchemaUpdate.java and its latest version in hibernate code base contains heavy modifications.

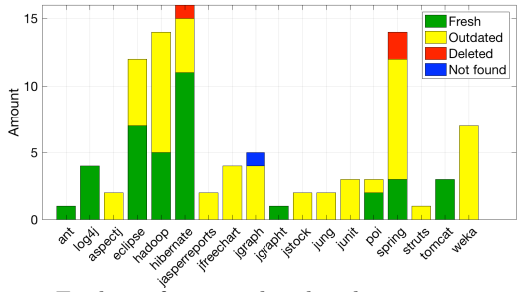


Figure 6: Findings from outdated code investigation of 96 pattern-A online clone pairs

violation if one does not check the license before integration third-party source code into their software. A study by An et al. [2] found 1,219 cases of potential license violations between 399 Android apps and Stack Overflow code.

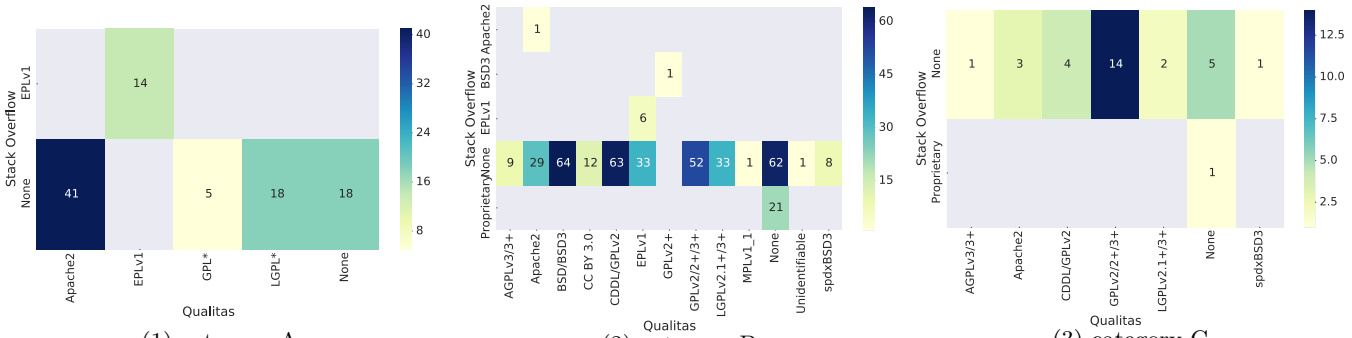
In our study, we reveal another possible situation of software licensing issues caused by code cloning "to Stack Overflow". We found with evidence that 96 pieces of code had been copied from 14 Qualitas projects to Stack Overflow as code examples. They are also marked as accepted answers which increased their probabilities of being reused. The 14 open source projects came with their respective software licenses (shown in Table 10). However, the licensing information were mostly missing from these clones after they were posted on Stack Overflow. The licensing terms on the top of source code are not copied because mostly a small part of the file was cloned to Stack Overflow. If developers copy and reuse these licensed pieces of code in their projects, conflicts may happen without their realisation.

The summary of licensing information is listed in Table 11. The licenses were extracted by Ninka, an automatic license identification tool [18]. For the ones that could not be automatically identified by Ninka and had been reported as `SeeFile` and `Unknown`, we looked at them manually. The table shows license of Stack Overflow snippets (denoted as SO) and Qualitas source code. The number of clone pairs of each license matching has been grouped according to online cloning patterns A, B, and C. In overall, we can see that most of the Stack Overflow snippets do not have licensing terms with them while their clone counterparts in Qualitas project do.

No license or compatible license: There are 85 clone pairs in type *NL* which do not have licensing information (None vs. None) and 13 pairs in type *CL* which have compatible licenses (Apache2 vs. Apache2; EPLv1 vs. EPLv1; None vs. CC BY 3.0). They are safe for being reused. Since source code and text on Stack Overflow posted before 1st March 2016 are protected by CC BY 3.0 (and MIT license

Table 10: 14 Qualitas projects with the highest number of true clones on Stack Overflow with their respective licenses

Project	Version	Licenses (no. of files)
aspectj	1.6.9	Apache-1.1 (182), CPLv1 (3), EPLv1 (2011), None (23), SeeFile (6), Unknown (286)
eclipse-SDK	4.1	Apache-2 (2183), BSD3NoWarranty (107), CDDLorGPLv2 (296), EPLv1 (18823), MPLv1.1 (93), None (224), SeeFile (8), spdxBSD3 (185), spdxMIT (1), Unknown (714)
hadoop	1.0.0	Apache-2 (1,935), spxdBSD/Apache-2 (8) None (33), Unknown (14)
hibernate	4.2.2	Apache-2 (20), LGPLv2.1+ (31) PublicDomain (1), None (1,850), SeeFile (4), Unknown (4,324)
jasperreports	3.7.4	LGPLv2.1+ (3), LGPLv3+ (1,581), None (1), Unknown (4)
jfreechart	1.0.13	LGPLv2.1+ (989)
jgraph	5.13.0.0	LGPLv2.1+ (4), spdxBSD (2), Unknown (151), None (24), SeeFile (9)
jstock	1.0.7c	LGPLv2.1+ (1), GPLv2+ (239), Apache-2 (1), BSD3 (1), None (23), SeeFile (1), spdxMIT (3), Unknown (5)
jung2	2-0-1	N/A
junit	4.11	None (160), Unknown (4)
poi	3.6	Apache-2 (2,002), None (5)
spring	3.0.5	Apache-2 (2,982)
struts2	2.2.1-all	Apache-2 (1,717), spdxBSD3 (6), None (118), SeeFile (1), Unknown (2)
tomcat	7.0.2	Apache-2 (1,313), None (11)



(1) category A (2) category B (3) category C

Figure 7: Licensing between Stack Overflow code snippets and Qualitas Java file of 523 true online clone pairs

Table 11: Licenses of 523 clone pairs in category A, B, and C

Type	SO	Qualitas	A	B	C
No license (*NL*)	None	None	18	62	5
Compatible License (*CL*)	Apache2	Apache2	0	1	0
	EPLv1	EPLv1	14	6	0
	None	CC BY 3.0	0	12	0
Incompat. License (*IL*)	None	Apache2	41	29	3
	None	BSD/BSD3	0	64	0
	None	CDDLorGPLv2	0	63	4
	None	EPLv1	0	33	0
	None	GPLv2/2+/3+	5	52	14
	None	LGPL2.1+/3+	18	33	2
	None	AGPLv3/3+	0	9	1
	None	MPLv1_1	0	1	0
	None	spdxBSD3	0	8	1
	None	Unidentifiable	0	1	0
	Proprietary	None	0	21	1
	BSD3	GPLv2+	0	1	0
Total			96	396	31

from 1st March 2016 onwards), we can treat Stack Overflow code snippets without licensing information as having CC BY 3.0 by default. CC BY 3.0 license is relaxed and it only request attribution when reuse.

Incompatible license: there are a large number of clone pairs in type *IL* which do not contain licensing information after they are posted on Stack Overflow or having different license that its Qualitas clone counterpart. 96 category-A clones are the pairs that we have evidence of being copied from Qualitas to Stack Overflow. 64 of them do not contain licensing terms after posted and are controversial for being reused. 396 clone pairs in category B are highly similar clones that we do not have evidence of copying. 315 of them have incompatible licenses. Interestingly, there are 21 clone pairs that Stack Overflow snippets contain Sun and MIT-like license while Qualitas clones do not have any license information. There is also one clone pair having BSD3 license on Stack Overflow while having GPLv2+ in Qualitas source code. Lastly, 26 out of 31 clone pairs in category C have incompatible license. They are clones that have been identified to be copied from an external source. Again, most of the clones in Qualitas contain license while the Stack Overflow snippets do not.

3.4.1 *Interesting Finding: External Clones*

We found 31 clone pairs which either one of the clones or both of them are found to be a copy of code snippet from somewhere else besides Qualitas project. This compliments the findings of inter clones between software projects [55] by showing that clones can also be copied over different websites on the Internet. There are 15 clone pairs that are originated from programming websites. For example, we found a clone of code snippet to convert numbers into words which is copied from http://www.rgagnon.com/javadetails/java-0426.html. Both Stack Overflow and *Compiere* project source code contain attribution to the original source. Another example is a code of algorithm to generate graphical *Perlin noise* copied from http://mrl.nyu.edu/~perlin/noise/. This code is used in both Stack Overflow and *AoI* project with attribution. The original code of the 15 clone pairs do not have licensing information so these clones do not violate any license. However, there are 4 clone pairs that are cloned from *JUnit* to both Stack Overflow and *Eclipse* and 12 pairs cloned from other Java projects outside Qualitas corpus. They are originated from *zxing* projects, Apache projects (ant, jasper), Java packages (`java.util`, `javax.servlet`, JUnit, and `org.bouncycastle.crypto` project.

Table 12: 31 external clones found in Stack Overflow posts

Type	FOSS	Web
StackOverflow only	1	1
StackOverflow + Qualitas	15	14

4. THREATS TO VALIDITY

5. RELATED WORK

Stack Overflow is a gold mine for software engineering research. Its rich and developer-driven data are invaluable. Since posts on Stack Overflow may contain code snippets embedded within natural language text, they become a huge database for source code and code-relevant information. The Stack Overflow data set 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 [40]. A follow up work, Prompter, by Ponzanelli et al. [41] 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, 53, 54, 14]. Furthermore, Treude et al. use machine learning techniques to extract insight sentences from Stack Overflow and use them to improve API documentation [58].

Another aspect is knowledge extraction from Stack Overflow. Nasehi et al. studied what makes a good code example by analysing answers from Stack Overflow [36]. Similarly, Yang et al. [64] analysed Stack Overflow snippets across various programming languages and observed that code snippets in Python and Javascript are the most usable. Wang et al. [60] 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. [35, 45, 9, 6], while some studies aim to improve Stack Overflow itself [14, 61].

Code clone detection is a long-standing research topic in software engineering. Whether clones are good or bad for software is still controversial [48, 26, 27, 31, 22, 19, 21]. However, by only knowing how many code clones residing in software and how they evolve [39, 34] can provide several valuable insights into the software systems. Besides clones, clone detection has its several applications such as software plagiarism detection [42], 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 [46, 43, 55]. Some tools use string comparison techniques such as Simian [50]. NiCad [46, 11] also exploits Longest Common Subsequence (LCS) string similarity measure to discover clones after applying code pretty-printing using TXL [10]. Many tools do not work on original source code directly but transform them into an intermediate representation such as tokens and apply similarity measurement on them. These tools include SourcererCC [48], CCFinder [24], CP-Miner [33], iClones [20] and a few more [7, 51, 15, 42, 49]. To find more challenging clones such as clones with added/deleted/reordered statements or equivalent loop and conditional statements (i.e. type-3 clones), structural similarity of clones is needed. This structural similarity can be discovered by comparing AST as found in CloneDR [3] and Deckard [23] or by using program dependence graphs [30, 29].

Kapser et al. studied clones in Linux file systems and create 11 patterns of code cloning based on four groups: Forking, Templating, Customization and Exact match [26, 27]. Our study partially adapted their patterns for our online code clone classification scheme.

Clone agreement is useful in when clone oracle is absent. Since clone detectors are different in their detection approaches, they may behave differently and report different clones even on the same data set. Some researchers exploit these different behaviours of clone detectors by finding their agreement and obtain highly-confident clones [5, 62]. Using the same data set, clone pairs that are agreed by several tools (or several code similarity measurement techniques) are highly potential to be true clones than the ones reported by only a single tool [62, 44, 16]. These studies also report that sensitivities of the tools' parameter settings have strong effects to the results [62, 44].

Software licensing is crucial for open source and industrial software development. Di Penta et al. study an evolution of software licensing in six FOSS and found that licensing statements change over time [13]. German et al. [17] performed an empirical study of code siblings (code clones among different systems coming from the same source) and found that licensing conflicts can occur between the clone siblings. Later, German et al. [18] created Ninka, a tool to automate identification of software license from program source code. We use Ninka to analyse software license from Stack Overflow fragments and Qualitas projects.

Reusing of outdated third-party source code occurs in software development. Xia et al. [63] show that a large number of open source systems reuse outdated third-party libraries from famous open source projects. Using outdated code give detrimental effects to the software since they may introduce vulnerabilities. Our study discovers similar findings in the context of outdated code from Stack Overflow.

The work that is closely similar to us is a study by An et al. [2]. 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 1,219 cases of potential license violations. However, 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. Our results confirm their findings that there are clones from software projects to Stack Overflow with potential licensing violations. In our work, we defined seven patterns of online code cloning and performed a large-scale manual check of 3,450 clone pairs. We discovered 96 clone pairs with strong evidences, based on natural text in comments and post contents, that they were copied from Qualitas to Stack Overflow. By comparing the clones to their latest versions in the software, we found that 58 code fragments on Stack Overflow are outdated and harmful for reuse.

6. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

7. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

8. REFERENCES

[1] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *SP '16*, pages 289–305, 2016.

[2] L. An, O. Mlouki, F. Khomh, and G. Antoniol. Stack Overflow: A Code Laundering Platform? In *SANER '17*, 2017.

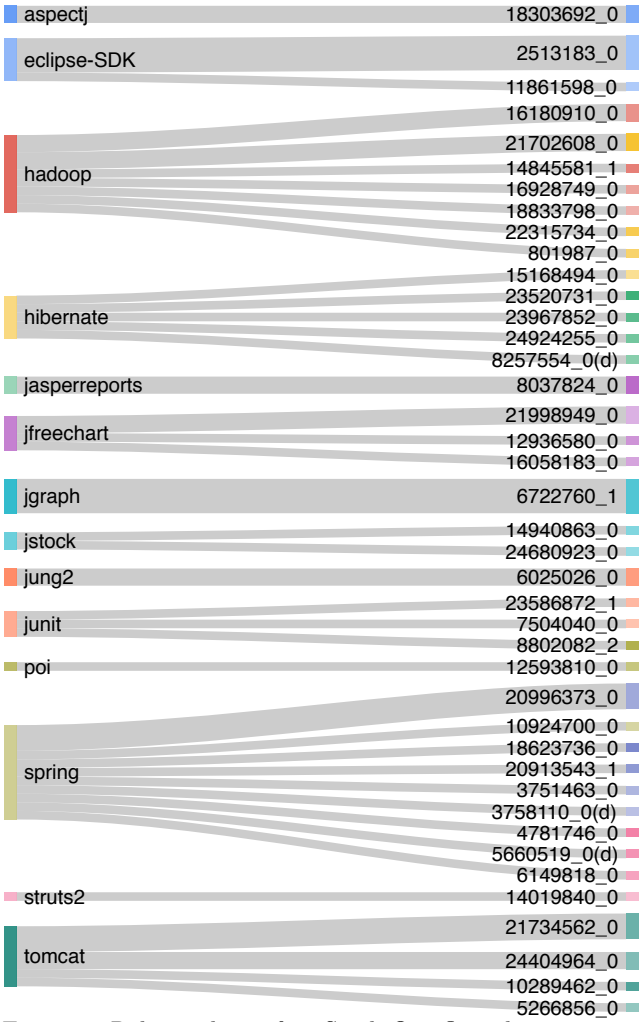


Figure 8: Relationships of 58 Stack Overflow clone pairs to their original projects. 55 are outdated and 3 are deleted (shown using (d) suffix).

[3] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *ICSM'98*, pages 368–377, 1998.

[4] N. E. Beckman, D. Kim, and J. Aldrich. An Empirical Study of Object Protocols in the Wild. In *ECOOP '11*, volume 6813 LNCS, pages 2–26. 2011.

[5] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Transactions on Software Engineering*, 33(9):577–591, 2007.

[6] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in StackOverflow: An empirical investigation. In *MSR '13*, pages 89–92, 2013.

[7] S. Burrows, S. M. M. Tahaghoghi, and J. Zobel. Efficient plagiarism detection for large code repositories. *Software: Practice and Experience*, 37(2):151–175, Feb. 2007.

[8] W. T. Cheung, S. Ryu, and S. Kim. Development nature matters: An empirical study of code clones in JavaScript applications. *Empirical Software Engineering*, pages 517–564, 2015.

[9] M. Choetkiertikul, D. Avery, H. K. Dam, T. Tran, and A. Ghose. Who Will Answer My Question on Stack Overflow? In *ASWEC '15*, pages 155–164, 2015.

[10] J. R. Cordy. The TXL source transformation language. *Science of Computer Programming*, 61(3):190–210, 2006.

[11] J. R. Cordy and C. K. Roy. The NiCad Clone Detector. In *ICPC '11*, pages 3–4, 2008.

[12] J. Davies, D. M. German, M. W. Godfrey, and A. Hindle. Software bertillonage: Determining the provenance of software development artifacts. *Empirical Software Engineering*, 18:1195–1237, 2013.

[13] M. Di Penta, D. M. German, Y.-G. Guéhéneuc, and G. Antoniol. An exploratory study of the evolution of software licensing. In *ICSE '10*, volume 1, page 145, 2010.

[14] T. Diamantopoulos and A. L. Symeonidis. Employing source code information to improve question-answering in stack overflow. In *MSR '15*, pages 454–457, 2015.

[15] Z. Duric and D. Gasevic. A source code similarity system for plagiarism detection. *The Computer Journal*, 56(1):70–86, Mar. 2012.

[16] M. Funaro, D. Braga, A. Campi, and C. Ghezzi. A hybrid approach (syntactic and textual) to clone detection. In *IWSC '10*, pages 79–80. ACM Press, 2010.

[17] D. M. German, M. Di Penta, Y.-G. Gueheneuc, and G. Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *MSR '09*, pages 81–90, 2009.

[18] D. M. German, Y. Manabe, and K. Inoue. A sentence-matching method for automatic license identification of source code files. In *ASE '10*, page 437, 2010.

[19] N. Göde and J. Harder. Clone Stability. In *CSMR '11*, pages 65–74, 2011.

[20] N. Göde and R. Koschke. Incremental Clone Detection. In *CSMR '09*, pages 219–228, 2009.

[21] J. Harder and N. Göde. Cloned code: stable code.

Journal of Software: Evolution and Process, 25(10):1063–1088, oct 2013.

[22] K. Hotta, Y. Sano, Y. Higo, and S. Kusumoto. Is duplicate code more frequently modified than non-duplicate code in software evolution? In *IWPSE-EVOL '10*, page 73, 2010.

[23] L. Jiang, G. Misherghi, Z. Su, and S. Glondu. DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones. In *ICSE '07*, pages 96–105, 2007.

[24] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, 28(7):654–670, 2002.

[25] C. Kapser and M. Godfrey. "Cloning Considered Harmful" Considered Harmful. In *WCRE '06*, pages 19–28, 2006.

[26] C. Kapser and M. W. Godfrey. Toward a taxonomy of clones in source code: A case study. In *ELISA '03*, pages 67–78, 2003.

[27] C. J. Kapser and M. W. Godfrey. "Cloning considered harmful" considered harmful: patterns of cloning in software. *Empirical Software Engineering*, 13(6):645–692, dec 2008.

[28] I. Keivanloo, J. Rilling, and Y. Zou. Spotting working code examples. In *ICSE '14*, pages 664–675, 2014.

[29] R. Komondoor and S. Horwitz. Using slicing to identify duplication in source code. In *SAS'01*, pages 40–56, 2001.

[30] J. Krinke. Identifying similar code with program dependence graphs. In *WCRE '01*, pages 301–309, 2001.

[31] J. Krinke. Is Cloned Code More Stable than Non-cloned Code? In *SCAM '08*, pages 57–66. IEEE, 2008.

[32] J. Krinke, N. Gold, Y. Jia, and D. Binkley. Cloning and copying between GNOME projects. In *MSR '10*, pages 98–101, 2010.

[33] Z. Li, S. Lu, S. Myagmar, and Y. Zhou. CP-Miner: Finding copy-paste and related bugs in large-scale software code. *IEEE Transactions on Software Engineering*, 32(3):176–192, 2006.

[34] M. Mondal, M. S. Rahman, R. K. Saha, C. K. Roy, J. Krinke, and K. A. Schneider. An Empirical Study of the Impacts of Clones in Software Maintenance. In *ICPC '11*, pages 242–245, 2011.

[35] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos. Analysis of the reputation system and user contributions on a question answering website: StackOverflow. pages 886–893, 2013.

[36] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example?: A study of programming Q&A in StackOverflow. In *ICSM '12*, pages 25–34. IEEE, 2012.

[37] C. Omar, Y. S. Yoon, T. D. LaToza, and B. A. Myers. Active code completion. In *ICSE '12*, pages 859–869, 2012.

[38] J.-w. Park, M.-W. Lee, J.-W. Roh, S.-w. Hwang, and S. Kim. Surfacing code in the dark: an instant clone search approach. *Knowledge and Information Systems*, 41(3):727–759, dec 2014.

[39] J. R. Pate, R. Tairas, and N. A. Kraft. Clone evolution: A systematic review. *Journal of software: Evolution and Process*, 25:261–283, 2013.

[40] L. Ponzanelli, A. Bacchelli, and M. Lanza. Seahawk: Stack Overflow in the IDE. In *ICSE '13*, pages 1295–1298, 2013.

[41] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In *MSR '14*, pages 102–111, 2014.

[42] L. Prechelt, G. Malpohl, and M. Philippsen. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11):1016–1038, 2002.

[43] C. Ragkhitwetsagul, J. Krinke, and D. Clark. Similarity of Source Code in the Presence of Pervasive Modifications. In *SCAM '16*. IEEE, 2016.

[44] C. Ragkhitwetsagul, M. Paixao, M. Adham, S. Busari, J. Krinke, and J. H. Drake. Searching for Configurations in Clone Evaluation A Replication Study. In *SSBSE '16*, 2016.

[45] C. Rosen and E. Shihab. What are mobile developers asking about? A large scale study using stack overflow. *Empirical Software Engineering*, 21(3):1192–1223, jun 2016.

[46] C. K. Roy and J. R. Cordy. An Empirical Study of Function Clones in Open Source Software. In *WCRE '08*, 2008.

[47] V. Saini, H. Sajnani, and C. Lopes. Comparing Quality Metrics for Cloned and non cloned Java Methods : A Large Scale Empirical Study. In *ICSE '16*, pages 256–266, 2016.

[48] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes. SourcererCC: Scaling Code Clone Detection to Big-Code. In *ICSE '16*, pages 1157–1168, 2016.

[49] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: Local algorithms for document fingerprinting. In *SIGMOD'03*, 2003.

[50] Simian - similarity analyser. http://www.harukizaemon.com/simian/, Accessed: 2015-08-27.

[51] R. Smith and S. Horwitz. Detecting and measuring similarity in code clones. In *IWSC'09*, 2009.

[52] C. J. Sprigman. Oracle v. Google. *Communications of the ACM*, 58(5):27–29, apr 2015.

[53] K. T. Stolee, S. Elbaum, and D. Dobos. Solving the Search for Source Code. *ACM Transactions on Software Engineering and Methodology*, 23(3):1–45, jun 2014.

[54] S. Subramanian and R. Holmes. Making sense of online code snippets. In *MSR '13*, pages 85–88, 2013.

[55] J. Svajlenko and C. K. Roy. Evaluating modern clone detection tools. In *ICSME'14*, pages 321–330, 2014.

[56] C. Taube-Schock, R. J. Walker, and I. H. Witten. Can We Avoid High Coupling? In *ECOOP '11*, volume 6813 LNCS, pages 204–228, 2011.

[57] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble. Qualitas corpus: A curated collection of java code for empirical studies.

In *APSEC '10*, pages 336–345, Dec. 2010.

[58] C. Treude and M. P. Robillard. Augmenting API documentation with insights from stack overflow. In *ICSE '16*, pages 392–403, 2016.

[59] B. Vasilescu, A. Serebrenik, and M. van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *ICSM '11*, pages 313–322, 2011.

[60] S. Wang, D. Lo, and L. Jiang. An Empirical Study on Developer Interactions in StackOverflow. pages 1019–1024, 2013.

[61] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. In *ICSME '14*, pages 291–300, 2014.

[62] T. Wang, M. Harman, Y. Jia, and J. Krinke. Searching for Better Configurations: A Rigorous Approach to Clone Evaluation. In *FSE '13*, pages 455–465, 2013.

[63] P. Xia, M. Matsushita, N. Yoshida, and K. Inoue. Studying Reuse of Out-dated Third-party Code. *Information and Media Technologies*, 9(2):155–161, 2014.

[64] D. Yang, A. Hussain, and C. V. Lopes. From Query to Usable Code: An Analysis of Stack Overflow Code Snippets. In *MSR '16*, pages 391–402, 2016.