Are We Reusing Outdated Code from Stack Overflow?

[1]Chaiyong Ragkhitwetsagul, [1]Jens Krinke, [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 and also valuable for educational purposes [**?**]. On Stack Overflow, each conversation contains a question and answer(s). The answers normally 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. As a result, 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 [4]. The benefits and drawbacks of clones are still controversial. Several authors state that clones lead to bug propagations and software maintenance issues [15], while some others have proofs that in some cases clones are not harmful than normal code or even beneficial [31, 16]. Code cloning can also have side effects of violating software licenses. Carelessly cloning code from one project and reusing it in another project with different license may cause software licensing violation [11].

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**. There are three 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; and (3) code is implicitly cloned from one software project to another by having Stack Overflow as a medium. Online code clones can similarly lead to a problem of bug propagation as classical code clones. Unfortunately, they are more difficult to locate and fix since the search space from online corpora is larger and no longer confined in a local repository.

A motivating example of problems caused by online code clones can be found in 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 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);`, 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 and outdated. This example shows that there can be inconsistencies between online code clones and its original. This is an emerging and challenging problem. Since studies in this area are still limited, we aim to gain more insight of the problem in this study.

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

MSR '17 May 20–21, 2017, Buenos Aires, Argentina

DOI: 10.1145/1235

```
/* Code in Stack Overflow #22315734 */
1 public int compare (byte[] b1,int s1,int l1,
2                     byte[] b2,int s2,int l2) {
3   try {
4     buffer.reset(b1,s1,l1); /* parse key1 */
5     key1.readFields(buffer);
6     buffer.reset(b2,s2,l2); /* parse key2 */
7     key2.readFields(buffer);
8   } catch (IOException e) {
9     throw new RuntimeException(e);
10   }
11   return compare(key1,key2); /* compare them */
12 }
```

```
/* WritableComparator.java (2016-09-26) */
1 public int compare(byte[] b1,int s1,int l1,
2                    byte[] b2,int s2,int l2) {
3   try {
4     buffer.reset(b1,s1,l1); /* parse key1 */
5     key1.readFields(buffer);
6     buffer.reset(b2,s2,l2); /* parse key2 */
7     key2.readFields(buffer);
8     buffer.reset(null,0,0); /* clean up reference */
9   } catch (IOException e) {
10    throw new RuntimeException(e);
11  }
12  return compare(key1, key2); /* compare them */
13 }
```

Figure 1: The same code fragments, WritableComparator.java, on Stack Overflow post 22315734 and latest version in hadoop code base

This paper makes the following primary contributions:

1. A manual study of online code clones: We used two clone detection tools to discover 266,837,480 similar code fragment pairs and manually investigated 7,840 candidate clone pairs between Java code fragments obtained from Stack Overflow accepted answers and 109 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 238 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. Furthermore, 50 out of the 84 clones are outdated and questionable for being reused.

2. EMPIRICAL STUDY

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

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

RQ2 (classification of online code clones): *What are the main characteristics among the set of online code clones?* We group them into seven groups according to our pre-defined classification scheme so we can differentiate and understand the motivation of cloning. Some of the clones are copy from open source projects to Stack Overflow, while some are copied from a third-party location, and some are accidental clones containing boiler-plate code, and code stubs generated by IDE.

RQ3 (effects of online code clones): *what are the effects derived from online code clones? can they be harmful to software development?* Is there observable problems caused by clones between Stack Overflow and open source projects?

2.1 Experimental Framework

To answer the three research questions, an experimental framework is designed as depicted in Figure 2. We process two datasets, Stack Overflow and open source projects from Qualitas corpus. Java code fragments are extracted from Stack Overflow posts using regular expressions. We preprocess Java code in both datasets by removing comments and pretty-printing to increase accuracy of clone detection. Then, we deploy two clone detection tools, Simian [1] and NiCad [30, 7], to locate clones between the two datasets. Due to a technical limit of Simian and NiCad to scale to large datasets, we partition the input and run the tools multiple times. Each run is composed of the whole Stack Overflow data and a single Qualitas project. We repeat the process until we cover 111 projects.

We then convert the clone reports to General Clone Format (GCF) [39] and combine them into a single file. GCF provides a common format for clones which enable us to reuse scripts that analyse clone reports from Simian and NiCad. Moreover, using GCF, other additional clone detectors can be adopted, if needed, without any changes in the analysis. Simian do not provide an option to detect inter clones between two locations. Hence the Simian GCF clone report is pruned to contain only inter clone pairs between Stack Overflow and Qualitas project. In this step, all intra clone pairs within Stack Overflow and open source projects are removed. NiCad provides an option to detect inter clones so no pruning is needed. Next, clone pairs reported from the two clone detectors are pair-wise matched to find agreements using Bellon's clone overlapping criteria [4]. This step generates **agreed clone pairs**. They are clones with highest confidence since they receive agreement from both tools. Then, clone pairs reported by Simian and NiCad that do not find agreement are filtered by size of minimum 10 lines. This step generates **disagreed clone pairs**. The disagreed clone pairs are clones with less confidence than agreed ones. Finally, agreed and disagreed clone pairs are looked at and classified manually by the first author.

In the manual inspection process, we classify clones into categories according to their properties. This process takes approximately a months until we successfully classified 7,840 clone pairs into categories. Some of the clone candidates are false clones due to being boiler-plate code or IDE-generated and are discarded for further analysis. By ignoring the false clones, we compare licensing information of 477 remaining clone pairs for possibility of software licensing violations. Moreover, we look forward through history of the clones from the projects' git versioning systems. This is to see if there is any changes made to the clones after it has been copied, hence resulting in outdated clones on Stack Overflow.

2.2 Experimental Setup

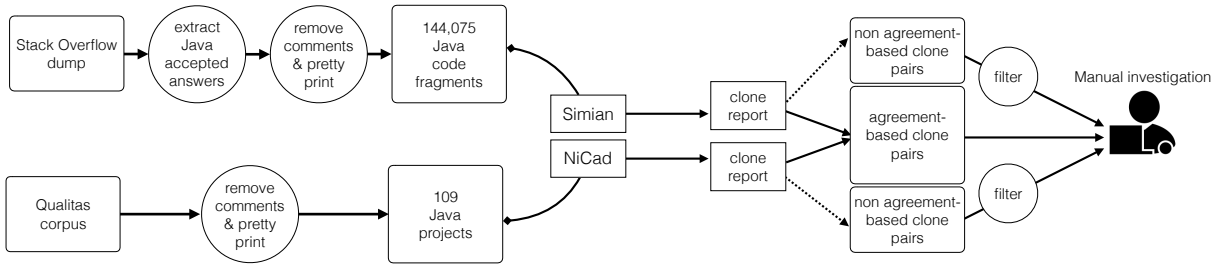

Figure 2: Experimental framework

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

2.2.1 Datasets

Stack Overflow: we extracted Java code snippets from accepted answers in a snapshot of Stack Overflow dump [2] in January 2016. The archived dump has a size of 9 gigabytes. The data dump is in XML format containing information of *Posts* (questions and answers) and supporting data such as user accounts and timestamps of the posts. We are interested in code snippets embedded in posts which are pieces of code located between `<code></code>` tags. We filtered the snippets with two filtering criteria. First, we ignore snippets that are less than 6 lines since they are usually not considered as clones by clone detection research [**?**]. Second, we are only interested in code snippets from posts that are marked accepted answer since they have high chances to be reused than snippets in questions and other answers. Each snippet is extracted from the dump using regular expressions and saved to a file using its post ID as the file's name. We use `.java` extension so that the clone detectors can recognise them. If a Stack Overflow conversation has more than one code snippet in the accepted answer, we append an indexing number starting from zero after the post ID (e.g. 45051109_0.java, and 45051109_1.java). With the two filters, we finally obtained 144,075 Java code snippets which contain xxx lines of Java source code excluding comments and blank lines[3].

Open source systems: we selected an established corpus for an empirical software engineering study called **Qualitas** [36]. It is a curated Java corpus that has been used in several software engineering studies [35, 3, 37, 23]. The projects in the corpus represent various domains of software systems ranging from programming language to 3D and visualisation [36]. We selected the 20130901r snapshot 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–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 so it is removed from the study. This results in totally 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]https://archive.org/details/stackexchange

[3]measured by cloc: https://github.com/AlDanial/cloc

2.2.2 Clone Detectors

There is a number of restrictions in terms of choosing clone detection tools for this study. Firstly, they have to support Java. Secondly, 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 process code snippets that are neither a complete block nor compilable. Thirdly, 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 [1], NiCad [7, 30], CCFinder [15], iClones [13], and DECKARD [14] against Stack Overflow and Qualitas datasets. CCFinder, iClones, and DECKARD failed to successfully detect clones between 144,075 Stack Overflow code snippets and 109 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 million-SLOC code corpus with method or class incompleteness. So, we decided to use both of them.

Simian is a text-based clone detector which locate clones at line-level granularity and has been used extensively in several clone studies [28, 39, 20, 6, 19]. 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 clones of type 1 and type 2. **NiCad** is also a text-based clone detector which detects clones at either method- or block-level granularity. It can detect clones up to type 3 and is used in several clone studies [30, 28, 34, 39, 20, 32]. It utilises TXL for parsing and pretty-printing source code. It also provide code normalisation by variable renaming and code 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 Clone Candidate Filtering Using Agreement

Usually a number of clones detected between two large-scale datasets can be huge. In our study, there are totally

266,837,480 clone pairs reported. It is infeasible for human to manually validate all of them. One can do sampling of clones from this huge clone pair set. However, they may end up having most of them false positive clones. Therefore, we adopted an idea of clone agreement which has been used in clone research studies [39, 10, 29] 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 [29]. By using this agreed clone detection approach, we can reduce the number of clone candidates for manual investigation by paying more attention to the ones agreed by multiple tools. To find an agreement of two clone pairs, we resort to the approach of clone pair matching metric proposed by Bellon et al. [4]. Two clone pairs which have large enough overlapping clone lines can be categorised as either a good-match or an ok-match pair. A good-match clone pair has stronger agreement than an ok-match pair. We follow the same following definitions of good- and ok-match introduced in the original paper.

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

2.3 Clone detectors' parameter tuning

We are aware of effects of configurations to clone detection results and the importance of searching for optimised configurations in empirical clone studies [39, 29, 28, 34]. However, considering the size of the two datasets and search space of at least 15 Simian's and 5 NiCad's parameters, we are 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 *df*), 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. [39] (denoted by *EvCl*). The details of the two configurations are described in Table 2. Having two clone detectors multiplied by two chosen configurations, we look for agreements in four possible pair-wise combinations: Simian$_{df}$–NiCad$_{df}$, Simian$_{df}$–NiCad$_{EvCl}$, Simian$_{EvCl}$–NiCad$_{df}$, and Simian$_{EvCl}$–NiCad$_{EvCl}$.

Table 2: Configurations of Simian and NiCad

Tool	Parameters
Simian$_{df}$	threshold=6, ignoreStringCase, ignoreCharacterCase, ignoreModifiers
Simian$_{EvCl}$	threshold=5, ignoreIdentifiers, ignoreIdentifierCase, ignoreStrings, ignoreCharacters, ignoreSubtypeNames, balanceSquareBrackets
NiCad$_{df}$	MinLine=10, MaxLine=1000, UPI=0.30
NiCad$_{EvCl}$	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 and NiCad

Stats	Simian$_{df}$	Simian$_{EvCl}$	NiCad$_{df}$	NiCad$_{EvCl}$
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%

3. RESULTS AND DISCUSSION

We follow the experimental framework and detect clones between Stack Overflow and Qualitas corpus using the two selected clone detectors. To answer RQ1 and RQ2 together, we compute statistics of clone discovered by the tools and the true clone pairs confirmed by manual investigation. In the manual investigation process, we proposed an online clone classification scheme derived from common patterns found in online code clones. For RQ3, we look at the true positive clone pairs carefully and compare them to their latest versions. This is to see if the cloned code is still up-to-date. We also look at the license of each clone and observe a possibility of licensing violation because of cloning.

3.1 RQ1: Online code clones

The clone statistics obtained from running Simian and NiCad with *df* and *EvCl* configurations are presented in Table 3. Preliminary manual investigation of Simian's clone report revealed that there were problematic 11 fragments. These 11 fragments trigger Simian to generate large clone clusters containing a huge number of false clones of array initialisation. Hence, they were removed from Simian's clone reports before the analysis. From Table 3, Simian clones cover approximately 10% of the 144,075 Stack Overflow snippets, 1,406 reported by Simain$_{df}$ and 1,360 from Simain$_{EvCl}$ respectively. NiCad$_{df}$ reports clones in 1,197 Stack Overflow code snippets, while NiCad$_{EvCl}$ reports clones in a larger number of 12,884 snippets mainly due to its relaxed configurations. In terms of number of clone pairs, Simian$_{EvCl}$

Table 4: No. of projects in Qualitas successfully analysed by Simian and NiCad

	Simian$_{df}$	Simian$_{EvCl}$	NiCad$_{df}$	NiCad$_{EvCl}$
Successful	111	111	105	84
Clust. fail	–	–	6	16
Renm. fail	–	–	–	11

and NiCad$_{EvCl}$ report an enormous number of clone pairs of 59,936,722 and 206,718,663 respectively. This is expected since EvaClone configurations prefer recall [39]. The average clone size of Simian$_{df}$ is 7.45 lines which is bigger than its Simian$_{EvCl}$ counterpart of 4.81. Similarly, NiCad$_{df}$ has an average clone size of 9.54 lines which is bigger than 5.31 reported by NiCad$_{EvCl}$. 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 to Simian$_{df}$, Simian$_{EvCl}$, NiCad$_{df}$, and NiCad$_{EvCl}$ is 28%, 29%, 25%, and 21% accordingly.

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-match[4]. The number of projects processed by the clone detection tools are listed in Table 4. We found that NiCad generates errors and stopped processing while it is analysing some Qualitas projects. **FIXME: Report the errors to NiCad creator.** NiCad$_{df}$ could not detect clones in 6 projects due to clustering errors. For NiCad$_{EvCl}$, the tool generated renaming errors for 4 projects, and clustering errors for 13 projects. As a result, we did not have NiCad clone candidates in the agreed clone pairs for 6 and 27 projects with *df* and *EvCl* configurations respectively. This affected Simian clone candidates of the same projects to be removed from the agreed clone pairs as well since they could not have matching clone pairs.

The distributions of good-match clone pairs between four combinations of *df* and *EvCl* configurations are listed in Table 5 and depicted visually in Figure 3. There are 2,261 unique good-match pairs consisting of 10 pairs from Simian$_{df}$–NiCad$_{df}$, 26 pairs from Simian$_{df}$–NiCad$_{EvCl}$, 10 pairs from Simian$_{EvCl}$–NiCad$_{df}$, and 2,228 pairs from Simian$_{EvCl}$–NiCad$_{EvCl}$. According to the definition, ok-match clone pairs always subsume the good-match pairs. As a result, there are 29,944 ok-match clone pairs. 9,062 pairs are from Simian$_{df}$–NiCad$_{df}$, 1,017 pairs are from Simian$_{df}$–NiCad$_{EvCl}$, 88 are pairs from Simian$_{EvCl}$–NiCad$_{df}$, and 19,777 pairs are from Simian$_{EvCl}$–NiCad$_{EvCl}$. Between the four configuration sets, there are considerable amount of clone pairs shared between two adjacent sets, but there is no clone pair that is agreed by all four combinations.

Table 6: Statistics of disagreed clone pairs (Simian$_{df}$ and NiCad$_{df}$).

Tool	Filter	C$_{pairs}$	good/ok	remaining
Simian$_{df}$	*minline* ≥ 10 reg. expressions	67570	2546	871
NiCad$_{df}$	*similarity* $\geq 80\%$	229176	450	168

Table 5: Distribution of agreed clone pairs using Bellon's criteria for *df* and *EvCl* settings

Tool		No. of clone pairs	
Simian	NiCad	good-match	ok-match
df	*df*	10	9062
df	*EvCl*	26	1017
EvCl	*df*	10	88
EvCl	*EvCl*	2228	19777
Total		2274	29944
Total (unique)		2261	29091

3.1.2 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 another tool. The disagreement can be from misalignment of clone lines or different configurations that result in different clones reported. They are also clone pairs in projects with NiCad's errors (6 projects for *df* and 27 for *EvCl* configurations) that are missing from the agreed clone pairs. With the four configuration combinations, we decided to investigate only two, Simian$_{df}$, and NiCad$_{df}$, and drop Simian$_{EvCl}$ and NiCad$_{EvCl}$ due to their enormous amount of clone pairs (59 millions and 206 millions respectively). They 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 two pruning filters: clone size, 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 [31]. 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 second filter, similarity threshold, applies only to NiCad clone pairs since Simian does not provide this similarity threshold configuration.

For Simian$_{df}$, there are 20,348 clone pairs after using 10–line filter. Out of 20,348 pairs, 2,546 of them are ok-match pairs which are discarded. We filtered the results further by removing false positives such as similar `equals()`, `hashCode()` methods, getters and setters out by using regular expressions. We managed to reduce the number to 871 clone pairs remaining for manual investigation. For NiCad$_{df}$, 10-line threshold is already a minimum NiCad's clone size and regular expressions could not be used effectively as in Simian's case since NiCad detects type-3 clones as well. We thus filter NiCad's clones by using similarity threshold. We increase NiCad's similarity threshold from 70% to 80% (by adjusting NiCad's UPI from 0.3 to 0.2) and obtain 618 clone pairs. There are 450 ok-match pairs which we ignore, thus 168 clone pairs remaining for manual investigation. The statistics of the clones and classification results are reported in Table 6.

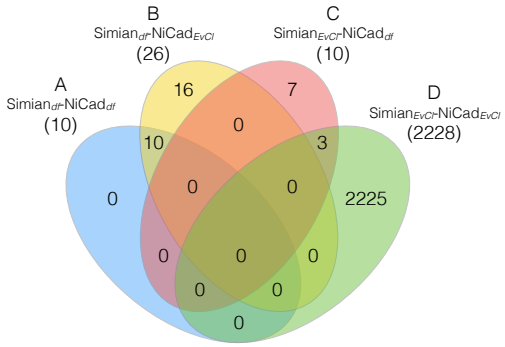

Figure 3: Distributions of good-match(0.7) pairs

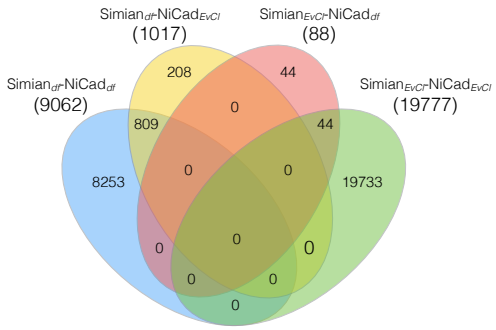

Figure 4: Distributions of ok-match(0.7) pairs

Table 7: Seven categories of online code clones

Cat.	Descriptions
A	Cloned from Qualitas to Stack Overflow (Q $\rightarrow$ S).
A'	Cloned from Stack Overflow to Qualitas (S $\rightarrow$ Q).
B	Cloned from each other or from an external source X (unknown) (S $\leftrightarrow$ Q $\vee$ (X $\rightarrow$ S $\wedge$ X $\rightarrow$ Q)).
C	Cloned from an external source (X $\rightarrow$ S $\wedge$ X $\rightarrow$ Q).
D	Boiler-plate or IDE auto-generated
E	Inheritance, interface implementation
F	Accidental similarity, false clones by normalisation

3.2 RQ2: Classifications of online code clones

We need an appropriate clone classification scheme to meaningfully categorise the selected clone candidates by hand. We started by studying the 8 patterns of cloning from Kapser et al. [16]. 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. Using Simian$_{df}$ 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 categories observed as common cloning patterns from the manual investigation. The seven categories ranging from A–F presented in Table 7. Category A is a clone pair that has evidence to be copied from Qualitas to Stack Overflow (by having comments in source code and explanation or links in Stack Overflow post), and vice versa in category A'. Category B is a clone pair that is exactly identical or highly similar but without any attribution of copying. Category C is a clone pair that has information confirming of copying from the same external source. Category D is a clone pair that is either boiler-plate code (e.g. `equals()` methods, or getters and setters) or IDE-generated (e.g. GUI initialisation). Category E are clones created by inheriting the same super class or implementing the same interface. They usually share similar overriding methods. The last category, category 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 classification scheme, we consider clone pairs in category A, A', B, C as true positive, and clone pairs in category D, E, F as false positive.

3.2.1 *Manual investigation of clone pair candidates*

The first author who has been working on clone detection research for two years takes the role of an investigator performing manual investigation. Following the classification scheme, the investigator manually goes through each agreed clone pair, looks at the clones, and chooses the most appropriate category 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-match clone pairs. There is one clone pair in category 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 category B. We found 3 pairs in category C that are copied from external sources. The rest are false positive clones. 58 clone pairs are found to be `equals()` methods, or getters and setters, category D. Six pairs are similar code from inheritance of the same superclass or implementing the same interface, category E. Finally, 2,189 clone pairs are categorised to E which means they are false clones.

For the ok-match clone pairs, we could not feasibly investigate all 23,868 pairs manually. According to the manual investigation of good-match pairs, we found that Simian$_{EvCl}$–NiCad$_{EvCl}$ produces a large number of 2,253 false positive results (accounts for 99.87% of Simian$_{EvCl}$–NiCad$_{EvCl}$ clone pairs) due to its recall preference. We thus decided to leave this configuration out of the ok-match manual investigation. There are totally 4,625 ok-match pairs that were investigated. The 47 true positive pairs found are combinations of 8, 29, and 10 clone pairs in category A, B, and C respectively.

FIXME: Maybe remove? We cannot be certain about the direction of copying in the category-B pairs, since there is no solid information of copying. We thus checked the timestamp of each Java file in Qualitas project and compa-

red it to their respective timestamp of Stack Overflow posts. We found that all Stack Overflow posts were created after their respectively Qualitas Java files **FIXME: check again since the dataset is updated**. This means that the copying can only be either (1) from Qualitas to Stack Overflow or (2) from an external source to both Stack Overflow and Qualitas independently. There is no clone pairs found in A' category.

Disagreed clone pairs: we performed a manual investigation and classification of 871 filtered clone pairs reported by Simian_{df} and 168 by NiCad_{df} in the same way as the agreed clone pairs. The results of investigation is reported in Table 8. There are 432 true positive clone pairs found from Simian_{df} consisting of 19 pairs in category A, 336 pairs in B, and 17 pairs in C. For NiCad_{df}, there are 36 true positive pairs consisting of 8 pairs in category A, 27 pairs in B, and 1 pair in C. The total number of true positive clone pairs is 523 as described in Table 9.

Table 9: Numbers of true positive online clone pairs found by manual investigation

Tool	A	A'	B	C	Total
good-pairs	1	0	4	3	8
ok-pairs	8	0	29	10	47
Simian_{df} pairs	79	0	336	17	432
NiCad_{df} pairs	8	0	27	1	36
Total	96	0	396	31	523

3.3 RQ3: Effects of online code clones

In this study, we are interested in effects of having code clones between open source software systems and Stack Overflow. From the manual investigation, we found 523 true positive online clone pairs. With this set of true clones, we investigated further and found that there are two potential issues, outdated code and software licensing violation.

3.3.1 Outdated 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. This is a normal issue driving the development of clone detection. We need to locate clones to be able to update them according to their updated original code [**?**, **?**]. 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 including bug fixing, this poses a problem if developers reuse a cloned code snippet from Stack Overflow without knowing that it is outdated. 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 the same later-be-fixed bug(s) into their software.

To check if the discovered Stack Overflow clones are outdated, we focus on the 96 category-A clone pairs that are copied in the direction from Qualitas to Stack Overflow (see Table 9). For each code snippet, we can track its origin from its Qualitas counterpart. Using the file in Qualitas project, we locate the latest version of the file in its version control repository (all of them use git) and compare the snippet to its latest version to see if any change has been made to the source code. If we find any change, we use *git blame* command to see who modified the source code and when.

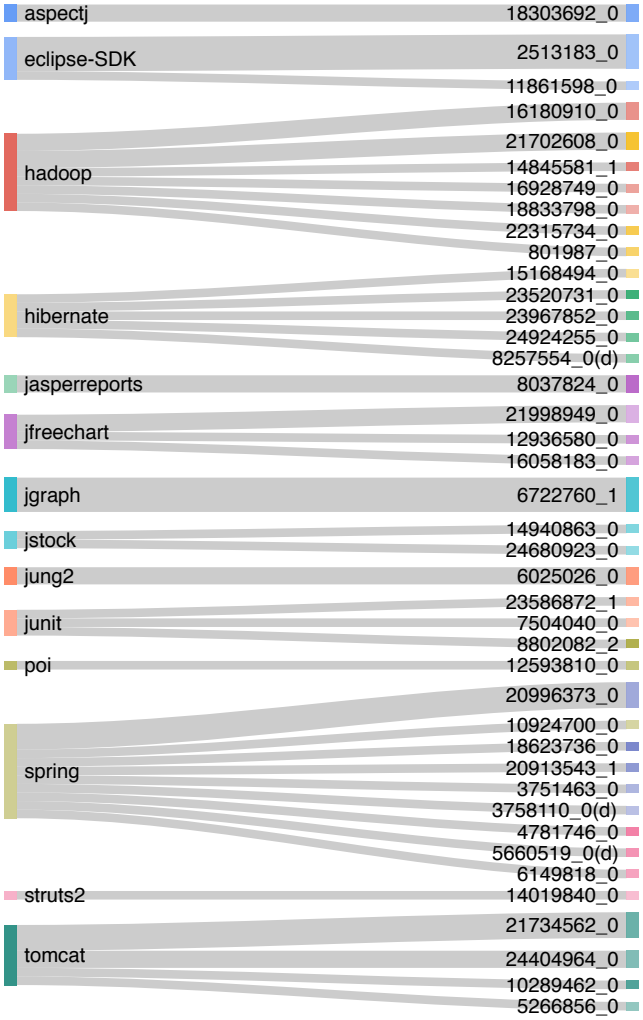

Figure 5: Relationships of 58 Stack Overflow clone pairs to their original projects. 55 are outdated and 3 are deleted (shown using (d) suffix).

Table 8: Classification results of agreed and disagreed clone pairs. S_u denotes a number of unique Stack Overflow snippets. Q_u and Q_{up} denote a number of unique Qualitas Java files and unique Qualitas projects respectively.

Classificaiton	A	A'	B	C	**Sum**	S_u	Q_u	Q_{up}	D	E	F	**Sum**	S_u	Q_u	Q_{up}	**Total**	S_u	Q_u	Q_{up}
good-match(0.7)	1	0	4	3	**8**	7	6	6	58	6	2189	**2253**	81	693	58	**2261**	87	699	59
ok-match(0.7)	8	0	29	10	**47**	12	14	10	9158	35	82	**9275**	100	204	30	**9322**	112	218	35
$Simian_{df}$	19	0	336	17	**432**	148	229	38	209	63	167	**439**	136	214	49	**871**	271	428	58
$NiCad_{df}$	8	0	27	1	**36**	16	15	5	77	3	82	**132**	52	55	17	**168**	67	69	19

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), GLGPLv2.1+ (31) PublicDomain (1), None (1,850), SeeFile (4), Unknown (4,324)
jasperreports	3.7.4	GLGPLv2.1+ (3), GLGPLv3+ (1,581), None (1), Unknown (4)
jfreechart	1.0.13	GLGPLv2.1+ (989)
jgraph	5.13.0.0	GLGPLv2.1+ (4), spdxBSD (2), Unknown (151), None (24), SeeFile (9)
jstock	1.0.7c	GLGPLv2.1+ (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)

Figure 7 shows the findings of outdated clone investigation. The comparison with the clones' latest code reveals that there are 55 clone pairs that are outdated. These are clone pairs that were copied from Qualitas projects to Stack Overflow and marked as accepted answers, and later have been changed during the development. *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 contains more disruptive changes. The code snippet in Stack Overflow Post 23520731 which is a copy of *SchemaUpdate.java* in *hibernate* has been heavily modified (as shown in Figure 6). Moreover, there is a code snippet in Stack Overflow post 3758110 which is a copy of *DefaultAnnotationHandlerMapping.java* in Spring. It was deleted in commit 02a4473c62d8240837bec297f0a1f3cb67ef8a7b by Chris Beams on 20th January 2012 at 22:51:02 two years after it was posted and the latest version of Spring does not contain the Java file anymore.

The outdate code clones without latest changes can cause problems ranging from uncompilable code due to version difference to bug propagation. An outdated code with a subtle change such as the one in *WritableComparator.java* can be copied and reused without awareness from developers. Fortunately, Stack Overflow has a natural voting mechanism that can mitigate this kind of outdated code. 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 vote for the answer with outdated code. However, if the answer is marked as accepted by the person who asks the question (which is our case), it is still attractive to naive developers who are not aware of the voting mechanism.

3.3.2 Software licensing violation

Software licensing is a paramount factor in software development. Violation of software license can cause a major impact to the delivery of the software and also lead to legal issues. It is an emerging area that software engineering research community is paying attention to. For example, there are studies of automatic technique to identify software licensing from source code files [12] and the evolution of licenses in open source projects [8].

In our study, we reveal another possible situation of software licensing issue caused by code cloning to Stack Overflow. We found that there are at least 84 pieces of code have been copied from 13 open source projects in Qualitas dataset to Stack Overflow as examples. They are also marked as accepted answers which increase their probability of being reused. These 13 open source projects come with

Table 11: 58 code clones in Stack Overflow that were altered, rewritten, or removed from the project after posted and their respective licenses

No.	Project	File	License	SO Post	Changes	Date
1	aspectj-1.6.9	Agent.java	–	18303692	alteration	2015-09-08
2	aspectj-1.6.9	Agent.java	–	18303692	alteration	2015-09-08
3	eclipse-SDK	GenerateToStringAction.java	EPLv1	2513183	alteration	2015-03-17
4	eclipse-SDK	GenerateToStringAction.java	EPLv1	2513183	alteration	2015-03-17
5	eclipse-SDK	GenerateToStringAction.java	EPLv1	2513183	alteration	2015-03-17
6	eclipse-SDK	GenerateToStringAction.java	EPLv1	2513183	alteration	2011-03-01
7	eclipse-SDK	WizardDialog.java	EPLv1	11861598	alteration	2011-02-03
8	hadoop-1.0.0	DBCountPageView.java	Apache-2	21702608	alteration	2011-06-12
9	hadoop-1.0.0	DBCountPageView.java	Apache-2	21702608	alteration	2011-06-12
10	hadoop-1.0.0	JobSubmissionFiles.java	Apache-2	14845581	alteration	2012-06-25
11	hadoop-1.0.0	LineRecordReader.java	Apache-2	16180910	alteration	2011-07-25
12	hadoop-1.0.0	LineRecordReader.java	Apache-2	16180910	alteration	2011-07-25
13	hadoop-1.0.0	StringUtils.java	Apache-2	801987	alteration	2013-02-04
14	hadoop-1.0.0	TestJobCounters.java	Apache-2	18833798	alteration	2011-06-12
15	hadoop-1.0.0	TextOutputFormat.java	Apache-2	16928749	alteration	2011-06-12
16	hadoop-1.0.0	WritableComparator.java	Apache-2	22315734	alteration	2014-11-20
17	hibernate-4.2.2	ConnectionProviderInitiator.java	–	15168494	alteration	2012-06-24
18	hibernate-4.2.2	Example.java	–	24924255	alteration	2013-04-23
19	hibernate-4.2.2	SchemaUpdate.java	–	23520731	alteration	2016-02-05
20	hibernate-4.2.2	SettingsFactory.java	–	8257554	removal	2011-03-11
21	hibernate-4.2.2	SQLServer2005LimitHandler.java	–	23967852	alteration	2015-03-12
22	jasperreports-3.7.4	JRVerifier.java	GLGPLv3+	8037824	alteration	2008-04-17
23	jasperreports-3.7.4	JRVerifier.java	GLGPLv3+	8037824	alteration	2011-05-20
24	jfreechart-1.0.13	AbstractXYItemRenderer.java	GLGPLv2.1+	12936580	alteration	2016-02-19
25	jfreechart-1.0.13	KeyToGroupMap.java	GLGPLv2.1+	16058183	alteration	2013-07-03
26	jfreechart-1.0.13	SpiderWebPlot.java	GLGPLv2.1+	21998949	alteration	2008-06-02
27	jfreechart-1.0.13	SpiderWebPlot.java	GLGPLv2.1+	21998949	alteration	2008-06-02
28	jgraph-5.13.0.0	HelloWorld.java	GLGPLv2.1+	6722760	rewriting	2014-04-13
29	jgraph-5.13.0.0	HelloWorld.java	GLGPLv2.1+	6722760	rewriting	2014-04-13
30	jgraph-5.13.0.0	HelloWorld.java	GLGPLv2.1+	6722760	rewriting	2014-04-13
31	jgraph-5.13.0.0	HelloWorld.java	GLGPLv2.1+	6722760	rewriting	2014-04-13
32	jstock-1.0.7c	GoogleMail.java	GPLv2+	14940863	alteration	2015-12-13
33	jstock-1.0.7c	GoogleMail.java	GPLv2+	24680923	alteration	2015-12-13
34	jung2-2_0_1	ShortestPathDemo.java	–	6025026	alteration	2010-04-13
35	jung2-2_0_1	ShortestPathDemo.java	–	6025026	alteration	2010-04-13
36	junit-4	Assert.java	–	23586872	alteration	2015-05-12
37	junit-4	ExternalResource.java	–	7504040	alteration	2016-06-25
38	junit-4.11	ExpectException.java	–	8802082	alteration	2014-05-26
39	poi-3.6-20091214	WorkbookFactory.java	Apache-2	12593810	alteration	2015-04-29
40	spring-framework-3.0.5	AnnotationMethodHandlerExceptionResolver.java	Apache-2	5660519	removal	2012-01-20
41	spring-framework-3.0.5	AutowireUtils.java	Apache-2	20913543	alteration	2014-10-28
42	spring-framework-3.0.5	CustomCollectionEditor.java	Apache-2	18623736	alteration	2013-11-21
43	spring-framework-3.0.5	DefaultAnnotationHandlerMapping.java	Apache-2	3758110	removal	2012-01-20
44	spring-framework-3.0.5	DefaultPropertiesPersister.java	Apache-2	6149818	alteration	2013-03-19
45	spring-framework-3.0.5	DelegatingServletInputStream.java	Apache-2	20996373	alteration	2016-07-15
46	spring-framework-3.0.5	DelegatingServletInputStream.java	Apache-2	20996373	alteration	2008-12-18
47	spring-framework-3.0.5	DelegatingServletInputStream.java	Apache-2	20996373	alteration	2008-12-18
48	spring-framework-3.0.5	DispatcherServlet.java	Apache-2	4781746	alteration	2011-08-08
49	spring-framework-3.0.5	Jaxb2Marshaller.java	Apache-2	10924700	alteration	2012-08-28
50	spring-framework-3.0.5	ScheduledTasksBeanDefinitionParser.java	Apache-2	3751463	alteration	2016-07-05
51	struts2-2.2.1	DefaultActionMapper.java	Apache-2	14019840	alteration	2013-10-18
52	tomcat-7.0.2	BasicAuthenticator.java	Apache-2	21734562	alteration	2016-08-04
53	tomcat-7.0.2	BasicAuthenticator.java	Apache-2	21734562	alteration	2016-08-04
54	tomcat-7.0.2	CoyoteAdapter.java	Apache-2	24404964	alteration	2012-11-18
55	tomcat-7.0.2	CoyoteAdapter.java	Apache-2	24404964	alteration	2012-11-18
56	tomcat-7.0.2	FormAuthenticator.java	Apache-2	21734562	alteration	2016-08-04
57	tomcat-7.0.2	HttpServlet.java	Apache-2	5266856	alteration	2011-10-22
58	tomcat-7.0.2	JspRuntimeLibrary.java	Apache-2	10289462	alteration	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 6: Outdated code fragment on Stack Overflow post 23520731 copied from SchemaUpdate.java and its latest version in hibernate code base with heavy modifications

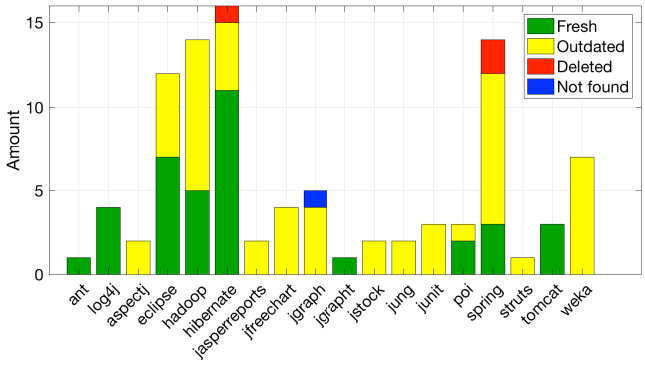

Figure 7: Findings from outdated code investigation of 96 category-A online clone pairs

their respective software licenses. However, the licensing information are mostly missing from these clones when they are posted on Stack Overflow. Mostly one or a few methods from the full source file are cloned. This makes the license information at the top of the file mostly left uncopied. If developers copy and reuse these pieces of code in their projects, a licensing conflict can quietly happen without realisation of the developers.

3.3.3 Interesting finding: external clones

4. THREATS TO VALIDITY

5. RELATED WORK

- Stack Overflow
  - Code example [22]
  - Search for code in Stack Overflow [9, 18, 24, 33]
  - Stack Overflow to help developers [26, 27]
  - Improving Stack Oveflow [9, 38, 5]
  - Developers' behaviours on Stack Overlow [39, 21]
- Code clones
  - Definition: Baxter et al. [2]
  - Comparison of clone detectors: [30, 28, 34]
  - NiCad [30, 7]
  - Simian [1]
  - Clone taxonomy [17]
  - Clone evolution [25, 20]
  - Comparing Quality Metrics for Cloned and non cloned Java Methods: A Large Scale Empirical Study [31].
- Agreement-based Clone Detection
  - Bellon's framework [4].
  - EvaClone [39]
  - Hybrid [10]
- Software licensing
  - Code siblings [11], Ninka – Automatic indification of SW license [12], Evolution of SW licensing [8]

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] Simian. http://www.harukizaemon.com/simian. Accessed: 07.04.2016.

[2] 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.

[3] N. E. Beckman, D. Kim, and J. Aldrich. An Empirical Study of Object Protocols in the Wild. In *European Conference on Object-Oriented Programming (ECOOP'11)*, volume 6813 LNCS, pages 2–26. 2011.

[4] 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.

[5] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in StackOverflow: An empirical investigation. *IEEE International Working Conference on Mining Software Repositories*, pages 89–92, 2013.

[6] 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.

[7] J. R. Cordy and C. K. Roy. The NiCad Clone Detector. In *ICPC '11 Proceedings of the 2011 IEEE 19th International Conference on Program Comprehension*, pages 3–4, 2008.

[8] M. Di Penta, D. M. German, Y.-G. Guéhéneuc, and G. Antoniol. An exploratory study of the evolution of software licensing. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, volume 1, page 145, 2010.

[9] T. Diamantopoulos and A. L. Symeonidis. Employing source code information to improve question-answering in stack overflow. In *MSR '15 Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 454–457, 2015.

[10] M. Funaro, D. Braga, A. Campi, and C. Ghezzi. A hybrid approach (syntactic and textual) to clone detection. In *Proceedings of the 4th International Workshop on Software Clones - IWSC '10*, pages 79–80. ACM Press, 2010.

[11] D. M. German, M. Di Penta, Y.-G. Gueheneuc, and G. Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 81–90, 2009.

[12] D. M. German, Y. Manabe, and K. Inoue. A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM international conference on Automated software engineering - ASE '10*, page 437, 2010.

[13] N. Göde and R. Koschke. Incremental Clone Detection. In *2009 13th European Conference on Software Maintenance and Reengineering*, pages 219–228, 2009.

[14] L. Jiang, G. Misherghi, Z. Su, and S. Glondu. DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones. In *Proceeding of the 29th International Conference on Software Engineering (ICSE'07)*, pages 96–105, 2007.

[15] 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.

[16] C. Kapser and M. Godfrey. "Cloning Considered Harmful" Considered Harmful. In *Proceeding of the 13th Working Conference on Reverse Engineering*, pages 19–28, 2006.

[17] C. Kapser and M. W. Godfrey. Toward a taxonomy of clones in source code: A case study. In *Proceedings of the ELISA workshop – Evolution of Large-scale Industrial Software Evolution*, pages 67–78, 2003.

[18] I. Keivanloo, J. Rilling, and Y. Zou. Spotting working code examples. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, pages 664–675, 2014.

[19] J. Krinke, N. Gold, Y. Jia, and D. Binkley. Cloning and copying between GNOME projects. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 98–101, 2010.

[20] 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 *2011 IEEE 19th International Conference on Program Comprehension*, pages 242–245, 2011.

[21] 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. *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining – ASONAM'13*, pages 886–893, 2013.

[22] 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 *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 25–34. IEEE, 2012.

[23] C. Omar, Y. S. Yoon, T. D. LaToza, and B. A. Myers. Active code completion. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 859–869, 2012.

[24] 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.

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

[26] L. Ponzanelli, A. Bacchelli, and M. Lanza. Seahawk: Stack Overflow in the IDE. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1295–1298, 2013.

[27] 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 *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, pages 102–111, 2014.

[28] C. Ragkhitwetsagul, J. Krinke, and D. Clark. Similarity of Source Code in the Presence of Pervasive Modifications. In *16th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'16)*. IEEE, 2016.

[29] C. Ragkhitwetsagul, M. Paixao, M. Adham, S. Busari, J. Krinke, and J. H. Drake. Searching for Configurations in Clone Evaluation A Replication Study. In *8th Symposium on Search-Based Software Engineering (SSBSE'16)*, 2016.

[30] C. K. Roy and J. R. Cordy. An Empirical Study of Function Clones in Open Source Software. In *2008 15th Working Conference on Reverse Engineering*, 2008.

[31] 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 Proceedings of the 32th International Conference on Software Maintenance and Evolution*, pages 256–266, 2016.

[32] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes. SourcererCC: Scaling Code Clone Detection to Big-Code. In *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, pages 1157–1168, 2016.

[33] 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.

[34] J. Svajlenko, I. Keivanloo, and C. K. Roy. Big data clone detection using classical detectors: an exploratory study. *Journal of Software: Evolution and Process*, 2014.

[35] C. Taube-Schock, R. J. Walker, and I. H. Witten. Can We Avoid High Coupling? In *European Conference on Object-Oriented Programming (ECOOP'11)*, volume 6813 LNCS, pages 204–228, 2011.

[36] 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 *2010 Asia Pacific Software Engineering Conference (APSEC2010)*, pages 336–345, Dec. 2010.

[37] 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 *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, pages 313–322, 2011.

[38] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 291–300, 2014.

[39] T. Wang, M. Harman, Y. Jia, and J. Krinke. Searching for Better Configurations: A Rigorous Approach to Clone Evaluation. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 455–465, 2013.