Are We Reusing Outdated Code from Stack Overflow?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 [3]. The benefits and drawbacks of clones are still controversial. Several authors state that clones lead to bug propagations and software maintenance issues [11], while some others have proofs that in some cases clones are not harmful than normal code or even beneficial [21, 12]. 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 [8].

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. However, they are harder to locate and fix since the search scope is no longer contained in a local repository. This is an emerging and challenging problem. Since studies in this area are still limited, we aim to address this problem in this study.

This paper makes the following primary contributions:

1. A manual study of online code clones: We used two clone detection tools to discover 130,644,707 clone pairs and manually investigated 2,371 meaningful clone pairs between Java code fragments obtained from Stack Overflow accepted answers and 63 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 48 clones that have been obviously copied from open source projects to Stack Overflow as code examples which violate their software licenses. Furthermore, 30 out of the 48 clones are outdated and dangerous for being reused.

2. EMPIRICAL STUDY

We perform an empirical study of online code clones between Stack Overflow and 63 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?*

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

DOI: 10.1145/1235

We would like to quantitatively measure the number of online code clones between Stack Overflow and open source projects.

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 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 generated by IDE.

RQ3 (effects of online code clones): *Are online code clones harmful?* Is there observable problems caused by clones between Stack Overflow and open source projects?

2.1 Experimental Framework

The experimental framework is depicted in Figure **??**. We deploy two clone detection tools, Simian [1] and NICAD [20, 4], to locate clones between two locations: code snippets from Stack Overflow, and 63 open source projects. The clone report is pruned to contain only inter clone pairs between the two locations. In this step, all intra clone pairs within Stack Overflow and open source projects are removed. Next, clone pairs reported from the two clone detectors are matched to find agreements using Bellon's clone overlapping criteria [3]. This step generates **agreement-based clone pairs**. They are clones with highest confidence since they receive agreement from both of the tools. Then, clone pairs reported by Simian and NICAD that do not find agreement are ranked and selected. This step generates **non agreement-based clone pairs**. The non agreement-based clone pairs are clones with less confidence than agreement-based ones. Finally, agreement-based and selected non agreement-based clone pairs are looked at and classified manually.

In manual inspection process, we classify clones into categories according to their properties. This process took around three months until we successfully classified 2,371 clone pairs into categories. By ignoring clone pairs manually classified as false clones, we compare licensing information of remaining clone pairs to see if they have any licensing conflicts. Moreover, we look forward through history of the clones from the projects' versioning systems (GIT) to see if there is any changes made to the clones after it has been copied.

2.2 Experimental Setup

2.2.1 Datasets

Stack Overflow: we extracted 144,075 Java code snippets from accepted answers in a snapshot of Stack Overflow dump [1] 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. Code snippet is a piece 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 ignored by clone detection research [**?**]. Second, we are only interested in code snippets from posts that are marked as "accepted answer" since they have high chances to be reused than snippets in questions and other answers. Each snippets is extracted from the dump using regular expressions and stored in a file with post ID as the file's name. We add `.java` as files' suffix so the clone detectors can recognise them. If a Stack Overflow conversation has more than one code snippet in an accepted answer, we append an indexing number starting from zero behind the ID (e.g. 45051109_0.java, and 45051109_1.java).

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

Qualitas corpus: we selected the latest version (20130901r) of Qualitas corpus [24] containing 64 Java open source projects. *eclipse* project does not contain source code so it has been removed from the study resulting in totally 63 projects. The details of the 63 Qualitas projects with their respective licenses are listed in Table 3.

2.2.2 Clone Detectors

We selected two clone detectors for this study: Simian [1] and NICAD [4, 20]. There is a few restrictions in terms of choosing clone detection tools for this study. First, 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 which are not in a complete block or compilable. This restriction forces us to remove two well-known clone detectors; CCFinder [11], and DECKARD [10]; from our study since they failed to analyse the complete 144,075 Stack Overflow snippets. Second, since the amount of code that have to be process are large **FIXME: Find SLOC of 144,075 SO snippets and 63 projects**, the clone detector must be scalable enough to execute and report clones in a reasonable amount of time. We found that both Simian and NICAD are able to handle clones in this situation with an acceptable running time.

2.2.3 Agreement-based Clone Detection

Since we did not have the clone oracle of the two datasets, we adopted an idea of clone agreement used in clone research [25, 7] to help filtering the reported clones for manual investigation.

3. RESULTS

The results of running 2 clone detectors: Simian and NiCad, to detect clones between 144,075 Stack Overflow fragments (Java accepted answers) and 63 open-source projects in Qualitas dataset is presented below. There are 2 tools selected: Simian and NiCad. They are configured using two different settings: default settings, and settings from a study by Wang et al. [25].

Table 1: No. of projects in Qualitas successfully analysed by Simian and NiCad

	Simian_{df}	Simian_{EvCl}	NiCad_{df}	NiCad_{EvCl}
Successful	63	63	57	46
Clust. fail	-	-	6	13
Ren. fail	-	-	0	4

Manual investigation of Simian's clone report showed that there were problematic 11 fragments. These fragments generate false clone containing array initialisation. Hence, they were removed from the result set before analysis.

3.1 Agreement based clone pairs vs. Non agreement based clone pairs

Table 2: Statistics of clones found between Stack Overflow and Qualitas projects using Simian and NiCad

Statistics	Simian$_{df}$			Simian$_{EvCl}$			NiCad$_{df}$			NiCad$_{EvCl}$		
	792 fragments			1229 fragments			1141 fragments			12400 fragments		
	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$
Total	24,929	–	–	16,957,362	–	–	105,118	–	–	113,557,298	–	–
Mean	38	7.54	0.27	14,444	4.80	0.28	107	9.52	0.25	9,397	5.21	0.20
Std Dev.	87	3.21	0.22	281,747	1.22	0.18	198	3.07	0.18	12,098	1.73	0.16
Max	551	49.00	0.94	9,599,676	18.00	0.89	1,792	39.00	0.80	227,077	44.00	0.86
Min	1	5.00	0.01	1	4.00	0.02	1	7.00	0.02	1	3.00	0.01
Median	3	7.00	0.23	22	5.00	0.24	15	8.00	0.19	6,105	5.00	0.15
Mode	1	7.00	0.25	1	4.00	0.50	1	8.00	0.53	1	4.00	0.33

The agreement-based clone pairs are the ones discovered using Bellon's *good*-match(0.7) and *ok*-match(0.7) criteria as listed in Table 4. Non-agreement based clone pairs are the ones that are solely reported by a single tool. The agreement-based pairs provide higher confident that they are real clones than the non-agreement based ones.

3.2 Agreement based clone pairs

For agreement-based clone pairs, we use a threshold of 0.7 for both *good* and *ok*-match. A visualisation of *good*-match common clone pairs between four sets of parameter settings can be seen from Figure 1. There are 1,357 unique *good*-match pairs. The distribution of 10,139 *ok*-match pairs, which subsume the *good*-match pairs, is depicted in Figure 2.

Nevertheless, NiCad produced renaming and clustering errors for some of the settings. This resulted in not all 63 projects had NiCad clone reports. For NiCad default settings (NiCad$_{df}$), 6 projects had clustering failed errors. For NiCad EvaClone settings (NiCad$_{EvCl}$), 4 projects had renaming failed errors and 13 projects had clustering failed errors as depicted in Table 1. So these projects are also missing from agreed clone pairs. **FIXME: Report the errors to NiCad creator.**

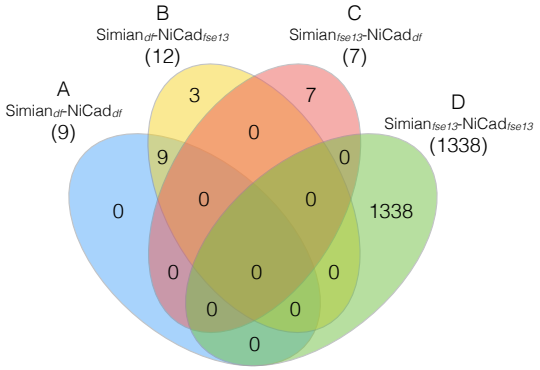

Figure 1: *good*-match(0.7) pairs

Table 4: Distribution of agreement-based clone pairs reported using Bellon's criteria

Tool		Qualitas-O	
Simian	NiCad	*good*-match	*ok*-match
default	*default*	9	473
default	*EvaClone*	12	353
EvaClone	*default*	7	60
EvaClone	*EvaClone*	1,338	9528
Total		1,366	10,414
Total (unique)		1,357	10,139

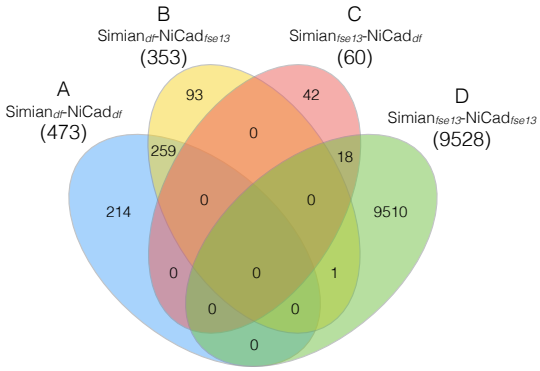

Figure 2: *ok*-match(0.7) pairs

3.3 Manual investigation of agreement-based clone pairs

The classification scheme is described in Table 5 and the classification results are shown in Table 6. We have manually investigated all of the 1,357 *good-match* ones reported by agreement of four different Simian and NiCad settings. However, for the *ok-match*, we could not investigate all of the 10,139 pairs manually. According to the distribution of category from *good-match* results, we can see that Simian$_{EvCl}$–NiCad$_{EvCl}$ produces a large number, 1,338, of false positive results (D, E, and F). Thus, we decided to leave them out of the manual investigation of *ok-match* pairs. There are totally 608 *ok-match* pairs that were investigated. The 39 true positive pairs found are combinations of 8 unique Stack

Table 3: 63 Qualitas projects (new versions retrieved on 2016-09-27)

Projects	Old version	New versions	Latest change	CVS	License	Notes
antlr4	4.0	4.5.4	25/09/2016	git	BSD	
apache-ant	1.8.4	1.10.0	09/04/2016	git	Apache2.0	
argouml	0.34	0.35.4	11/01/2015	svn	Eclipse 1.0	
artofillusion	2.8.1	3.0.2	27/08/2016	svn	GPL 2.0	
aspectj	1.6.9	1.8.9	12/05/2016	git	Eclipse 1.0	
axion	1.0-M2	-	08/03/2013	-	Proprietary (BSD/Apache-style)	
batik	1.7	1.9.0	11/05/2016	svn	Apache , v.2.0	
c-jdbc	2.0.2	-	16/09/2005	-	GLGPL 2.1	
castor	1.3.1	1.4.2	17/08/2016	git	Apache 2.0	
cayenne	3.0.1	4.0.M4	26/09/2016	git	Apache 2.0	
checkstyle	5.1	7.2	23/09/2016	git	GLGPL 2.1 & Apache 2.0	*Cli, Logging and Beanutils* packages are from the Apache Commons project.
cobertura	1.9.4.1	2.1.2	01/06/2016	git	GPL 2.0	
colt	1.2.0	-	09/09/2014	-	Proprietary (CERN)	Found multithreaded v.
columba	1.4	-	20/04/2007	-	Mozilla 1.1	
commons-collections	3.2.1	4.2	12/09/2016	svn	Apache 2.0	
compiere	330	-	-	-	GPL 2.0	No longer OSS
db-derby	10.6.1.0	10.12.1	13/08/2016	svn	Apache 2.0	
displaytag	1.2	2.0	17/08/2014	svn	MIT	
drawswf	1.2.9	-	02/04/2013	-	GPL 2.0	
drjava	20100913-r5387	???	03/09/2014	svn	BSD	Build to see version?
exoportal	???	???		git	GLGPL 3.0 & proprietary	Too many new projects
emma	2.0.5312	2.0.5312	09/05/2013	-	Common 1.0	
findbugs	1.3.9	3.0.1	06/03/2015	-	GLGPL 2.0	
fit-java	1.1	-	04/06/2013	-	GPL 2.0	
fitlibrary	20100806	???	29/07/2014	git	GPL 2.0	
freecol	0.10.7	0.11.6	26/09/2016	git	GPL 2.0	
freecs	1.3.20100406	-	22/04/2013	-	GPL 3.0	
freemind	0.9.0	1.0.0	16/08/2016	git	GPL 2.0+	
galleon	2.3.0	2.5.6	29/04/2013	-	GPL 2.0	
ganttproject	2.0.9	2.8.1	16/08/2016	git	GPL 3.0	
geotools	2.7-M3	16	27/09/2016	git	GLGPL 2.0	
hadoop	1.0.0	3.0.0-alpha2	26/09/2016	git	Apache 2.0	
heritrix	1.14.4	-	05/06/2013	-	GLGPL 2.1	
hibernate	4.2.2	5.2.3	22/09/2016	git	GLGPL 2.1+	
hsqldb	2.0.0	2.3.4	13/09/2016	svn	BSD	
htmlunit	2.8	2.24	26/09/2016	svn	Apache 2.0	
ireport	3.7.5	-	28/05/2014	-	Affero GLGPL 3.0	
itext	5.0.3	5.5.9	27/09/2016	git	Affero GLGPL 3.0	
informa	0.7.0-alpha2	-	07/11/2008	-	GLGPL 2.1 & Apache Software 1.1	
ivatagroupware	0.11.3	-	27/02/2013	-	GPL 2.0	
jfin_datemath	R1_0_1	-	25/04/2013	-	GPL 2.0	
joggplayer	114s	-	15/04/2013	-	GPL 2.0	
jag	6.1	6.2	08/04/2013	-	GPL 2.0 & BSD	BSD is for libraries.
james	2.2.0	2.3.2.1	14/08/2015	-	Apache 2.0	
jasml	0.10	-	08/03/2013	-	Apache Software	
jasperreports	3.7.4	6.3.1	27/09/2016	git	GLGPL 3.0	
javacc	5.0.0	7.0.0	15/08/2016	svn	Proprietary (Sun)	
jboss (wildfly)	5.1.0.GA	11.0.0.Alpha1	27/09/2016	git	GLGPL 2.1	Renamed to Wildfly.
jchempaint	3.0.1	3.4	01/09/2016	git	GLGPL 2.1+	
jedit	4.3.2	5.3.1	20/09/2016	svn	GPL 2.0	
jena	2.6.3	3.1.1	16/09/2016	git	Apache 2.0	
jext	5.0	-	18/08/2004	-	GPL 2.0	
jfreechart	1.0.13	1.5.0	29/08/2016	git	GLGPL 2.0	
jgraph	5.13.0.0	3.6.0.0	07/09/2016	git	Proprietary (mxGraph)	
jgraphpad	5.10.0.2	-	10/11/2006	-	GPL & GLGPL (derivatives)	
jgrapht	0.8.1	1.0.1	23/09/2016	git	GLGPL 2.1 & Eclipse 1.0	
jgroups	2.10.0.GA	4.0.0	26/09/2016	git	Apache 2.0	
jmoney	0.4.4	???	27/12/2015	git	GPL 2.0	
jparse	0.96	-	29/07/2004	-	GLGPL 2.1	
jpf	1.5.1	???	13/01/2012	-	Apache 2.0	
junit	4.11	4.12	04/12/2014	git	Eclipse 1.0	
shiftone-jrat	0.6	1-beta-1	17/11/2007	svn	GLGPL 2.0	
vuze	4812	5730	23/09/2016	svn	GPL 2.0	

Table 5: Classifications of clone creation

Category	Descriptions
A	Code in Stack Overflow is copied from Qualitas (Q → S).
A'	Code in Qualitas is copied from Stack Overflow (S → Q).
B	Code is copied either from each other or a third source (unknown) (S ↔ Q ∨ (T → S ∧ T → Q)).
C	Code in both places are copied from a third source T (known) (T → S ∧ T → Q).
D	Code is a boiler-plate or IDE auto-generated.
E	Code in both places initialise a similar/the same object; extend the same class/its subclass; implement the same interface.
F	Accidental similarity, false clone

Overflow fragments, and 9 unique Qualitas Java files from 6 different projects.

Since we are not certain about the direction of copying in the B-classified pairs, we checked the modification time of each Java file in Qualitas project and compare it to the timestamp of Stack Overflow answers. We found that all Stack Overflow code fragments were posted after their respectively similar Java files in Qualitas project. This means that the copying can only be either (1) Q → S or (2) from a third source to both S and Q independently.

3.4 Non-agreement based clone pairs

In the preliminary stage of our experiment, we found that there are 41 Stack Overflow fragments reported by Simian with default configurations. However, only 10 of them appear in the new results using tool's agreement. Thus, we further investigated the clone pairs reported by Simian and NiCad but *without* an agreement.

With our 4 settings, we decided to investigate only 2 settings, Simian$_{df}$, and NiCad$_{df}$, and drop Simian$_{EvaClone}$ and NiCad$_{EvaClone}$ due to their large number of false positives as shown in Table 7 and **??**. With the 2 selected settings, we investigated clone pairs having the minimum clone size of 10 SLOC as they are meaningful and tend to be real clone in modern clone detection [22].

For Simian$_{df}$, there were 9,383 clone pairs reported by the tool. Out of 9,383 pairs, 140 of them are the ones found in *ok*-pairs using agreement-based detection. We filtered the results further by removing false positives such as similar equals(), hashCode() methods, getters and setters out by using regular expression. We managed to remove 8,956 pairs using this method. Eventually, there were 287 clone pairs remaining for manual investigation. For NiCad$_{df}$, we obtained 7,040 clone pairs to look at which is infeasible for manual investigation. Hence, result filtering was also needed. However, regular expressions could not be used effectively as in Simian's case since NiCad allowed clones that are different at keywords/variable names or even added/deleted lines. So we decided to filter the results by selecting pairs that pass stricter clone criteria with UPI = 0.2. By reducing the UPI to 0.2, there were totally 166 pairs left. Out of 166, 52 are *ok*-pairs and 114 are remaining pairs for manual check (18 pairs are from *cayenne* and *iReport* that could not be analysed using UPI = 0.3). The statistics of the clones and classification results are reported in Table 8 and 9.

3.5 Manual investigation of non agreement-based clone pairs

We performed manual investigation of the clone pairs reported by Simian$_{df}$ and NiCad$_{df}$ in the same way as the agreement-based clone pairs. The results of the manual investigation is reported in Table 9.

Table 10: Numbers of true positives online clone pairs found by manual investigation

Tool	A	A'	B	C	Total
good-pairs	1	0	1	3	5
ok-pairs	8	0	23	8	39
Simian$_{df}$ pairs	35	0	89	7	131
NiCad$_{df}$ pairs	4	0	5	0	9
Total	48	0	118	18	184

4. EFFECTS OF ONLINE CODE CLONE

In this study, we are interested in the effects of online code clones to software development. From the manual investigation of 184 true online clone pairs, we found that there are two potential issues: stale online code, and software licensing violation.

4.1 Issue 1: stale online code clones

Stale online code occurs when a piece of code has been copied from a software project to Stack Overflow, and later it has been changed in the original project. However, in this situation, the copy is still unchanged. Since the code were updated due to various possible reasons including bug fixing, this can cause a problem if developers reuse stale online code from Q&A websites such as Stack Overflow. They might also introduce the same unfixed bug(s) into the software. To discover stale online code, we focus on the true online clone pairs that are copied in the direction of Q → S (class-*A* online clone pairs) in Table 10 which results in 48 pairs selected. We restricted it further to only the ones having versioning system so we can trace changes made to these clone pairs. Fortunately, all of the pairs were from projects with either git or svn so we did not remove any pair from this set.

The manual investigation of 48 class-A online clone pairs reveals that there are 30 stale clones. They are clone pairs that were copied from Qualitas projects to Stack Overflow and marked as *accepted* answers. The investigation results are described in Table 11.

4.2 Issue 2: clones with software licensing violations

Software licensing is vital in software industry. Violation of software license can have a major impact to the delivery of

Table 6: Qualitas-*O*: Classification results of *good*- and *ok*-match pairs which excludes the subsumed *good*-match and Simian$_{EvaClone}$-NiCad$_{EvaClone}$ pairs.

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	1	3	**5**	5	4	4	26	6	1320	**1352**	56	402	31	**1357**	61	406	32
ok-match(0.7)	8	0	23	8	**39**	8	9	6	480	28	61	**569**	76	60	16	**608**	83	68	19

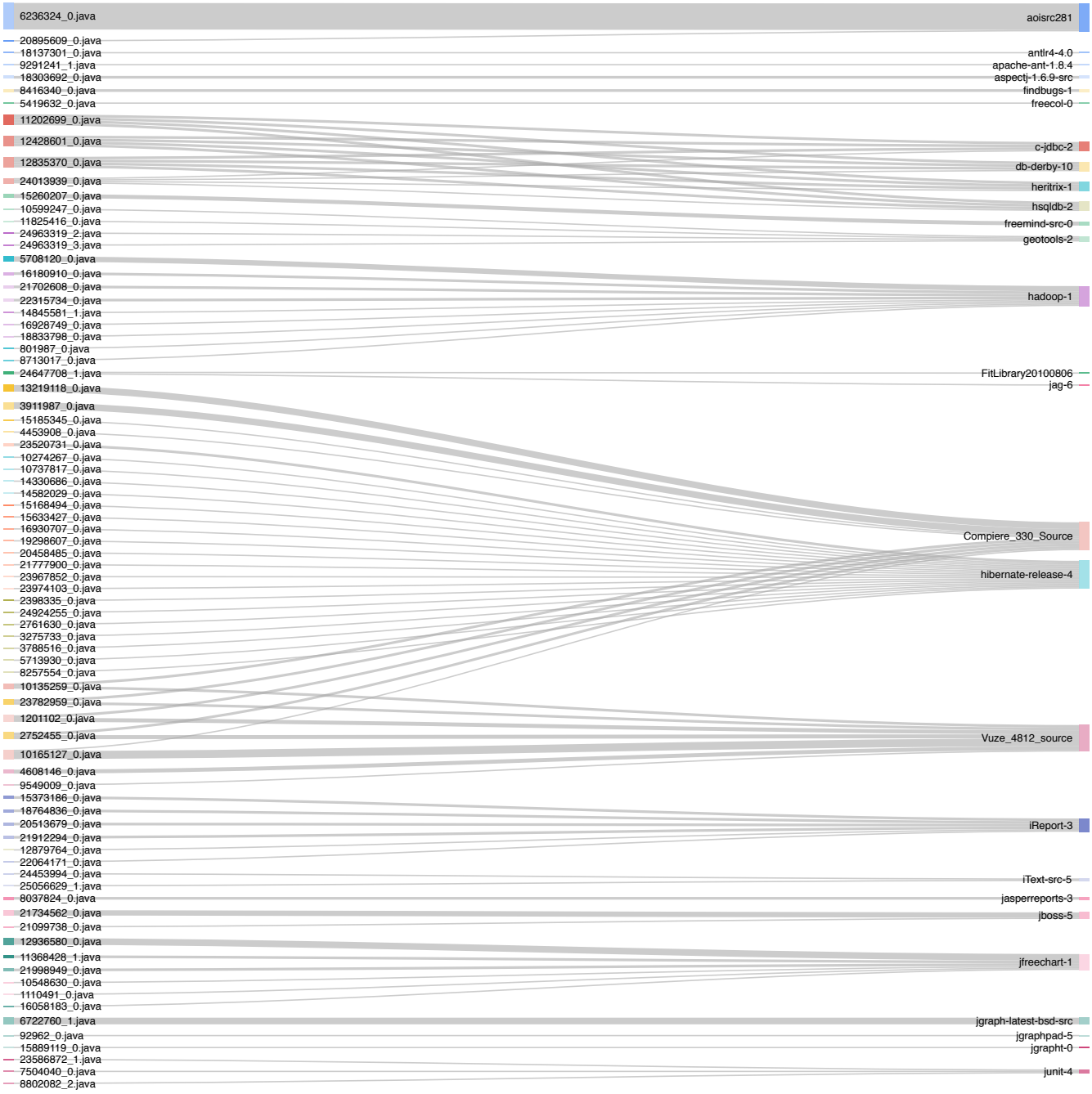

Figure 3: Relationships between 184 true online code clone found between Stack Overflow and Qualitas projects

Table 7: Distribution of classification category A–F according to *good*- and *ok*-match pairs. S denotes Simian and N denotes NiCad tool.

Category	A	A'	B	C	D	E	F	Total
good-match pairs								
S_{df}–N_{df}	1	0	1	3	0	4	0	9
S_{df}–N_{EvCl}	1	0	1	3	1	5	1	12
S_{EvCl}–N_{df}	0	0	0	0	7	0	0	7
S_{EvCl}–N_{EvCl}	0	0	0	0	18	1	1319	1338
Total	2	0	2	6	26	10	1320	1366
Total (unique)	1	0	1	3	26	6	1320	1352
ok-match pairs								
S_{df}–N_{df}	3	0	10	6	433	5	7	464
S_{df}–N_{EvCl}	8	0	22	4	250	25	32	341
S_{EvCl}–N_{df}	0	0	0	0	29	0	24	53
Total	11	0	32	10	712	30	63	858
Total (unique)	8	0	23	8	480	28	61	608

Table 8: Statistics of $Simian_{df}$ and $NiCad_{df}$ non agreement-based clone pairs.

Tool	Clone pairs	*ok*-pairs	filtered	remaining
$Simian_{df}$	9383	140	8956	287
$NiCad_{df}$	7040	226	6700	114

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 [9] and the evolution of licenses in open source projects [5].

In our study, we tackle another possible situation of software licensing issue caused by code cloning to Q&A websites. We found that there are at least 48 pieces of code have been copied from 9 open source projects to Stack Overflow as examples. These 9 open source projects come with software licenses. However, the licensing information are mostly missing from these clones. If developers copy and reuse these pieces of code in their projects, a licensing conflict can quietly happen without realisation of the developers.

5. THREATS TO VALIDITY

6. RELATED WORK

- Code clones
  - Definition: Baxter et al. [2]
  - Comparison of clone detectors: [20, 19, 23]
  - NiCad [20, 4]
  - Simian [1]
  - Clone taxonomy [13]
  - Clone evolution [18, 15]
  - Comparing Quality Metrics for Cloned and non cloned Java Methods : A Large Scale Empirical Study [21].
- Agreement-based Clone Detection
  - Bellon's framework [3].
  - EvaClone [25]
  - Hybrid [7]
- Software licensing
  - Code siblings [8], Ninka – Automatic indification of SW license [9], Evolution of SW licensing [5]
- Stack Overflow
  - Code example [16]
  - Search for code in Stack Overflow [6, 14, 17]

7. 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.

8. 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.

9. 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

Table 9: Classification results of 292 Simian$_{df}$ and 114 NiCad$_{df}$ non agreement-based clone pairs.

Tool/Classification	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}
Simian$_{df}$	35	0	89	7	**133**	68	57	23	13	10	133	**159**	39	69	23	**287**	103	121	31
NiCad$_{df}$	4	0	5	0	**9**	9	5	4	24	3	78	**105**	41	39	12	**114**	48	44	14

Table 11: Summary of stale code clones found

Project	Pairs	Stale	Fresh
apache-ant	1	0	1
aspectj	2	2	0
hadoop	14	9	5
hibernate	16	5	11
jasperreports	2	2	0
jfreechart	4	4	0
jgraph	5	5	0
jgrapht	1	0	1
junit	3	3	0
Total	48	30	18

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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 *Proceeding of the 19th International Conference on Program Comprehension*, pages 242–245. IEEE, 2011.

[16] 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.

[17] 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.

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

[19] 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.

[20] C. K. Roy and J. R. Cordy. NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization. In *2008 16th IEEE International Conference on Program Comprehension*, pages 172–181, 2008.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

Table 12: 30 stale code clones in Stack Overflow found by a manual investigation

No.	File	Stack Overflow Q&A	Changes made	Date
1	aspectjtools1.6.9-src/../Agent.java	18303692	alteration	2015-09-08
2	aspectjweaver1.6.9-src/../Agent.java	18303692	alteration	2015-09-08
3	hadoop-1.0.0/../WritableComparator.java	22315734	alteration	2014-11-20
4	hadoop-1.0.0/../StringUtils.java	801987	alteration	2013-02-04
5	hadoop-1.0.0/../DBCountPageView.java	21702608	alteration	2011-06-12
6	hadoop-1.0.0/../DBCountPageView.java	21702608	alteration	2011-06-12
7	hadoop-1.0.0/../LineRecordReader.java	16180910	alteration	2011-07-25
8	hadoop-1.0.0/../LineRecordReader.java	16180910	alteration	2011-07-25
9	hadoop-1.0.0/../JobSubmissionFiles.java	14845581	alteration	2012-06-25
10	hadoop-1.0.0/../TextOutputFormat.java	16928749	alteration	2011-06-12
11	hadoop-1.0.0/../TestJobCounters.java	18833798	alteration	2011-06-12
12	hibernate-release-4.2.2.Final/../SettingsFactory.java	8257554	removal	2011-03-11
13	hibernate-release-4.2.2.Final/../Example.java	24924255	alteration	2013-04-23
14	hibernate-release-4.2.2.Final/../SQLServer2005LimitHandler.java.java	23967852	alteration	2013-04-23
15	hibernate-release-4.2.2.Final/../ConnectionProviderInitiator.java	15168494	alteration	2016-02-24
16	hibernate-release-4.2.2.Final/../SchemaUpdate.java	23520731	alteration	2016-02-05
17	jasperreports-3.7.4/../JRVerifier.java	8037824	removal	2011-05-20
18	jasperreports-3.7.4/../JRVerifier.java	8037824	alteration	2013-12-08
19	jasperreports-3.7.4/../SpiderWebPlot.java	21998949	alteration	2013-11-22
20	jasperreports-3.7.4/../SpiderWebPlot.java	21998949	alteration	2013-11-22
21	jasperreports-3.7.4/../AbstractXYItemRenderer.java	12936580	alteration	2016-01-16
22	jfreechart-1.0.13/../KeyToGroupMap.java	16058183	alteration	2013-07-03
23	jgraph-latest-bsd-src/../GroupingRemoving.java	6722760	rewriting	2005
24	jgraph-latest-bsd-src/../HelloWorld.java	6722760	rewriting	2005
25	jgraph-latest-bsd-src/../HelloWorld.java	6722760	rewriting	2005
26	jgraph-latest-bsd-src/../HelloWorld.java	6722760	rewriting	2005
27	jgraph-latest-bsd-src/../HelloWorld.java	6722760	rewriting	2005
28	junit-4/org/junit/Assert.java	23586872	alteration	2015-05-12
29	junit-4/../ExpectException.java	8802082	alteration	2014-05-26
30	junit-4/../ExternalResource.java	7504040	alteration	2016-06-25