CloneOverflow: Discovery of Online Code Clones between Stackoverflow and Open Source Projects

[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

- Definition of code clones, online code clones, pros & cons
- Roles of Q&A websites in supporting software development and education
- Problems of code reuse (bug propagation, licensing conflicts)

This paper makes the following primary contributions:

1. A manual study of online code clone between Stackoverflow and open source projects: We manually investigate 2,371 clone pairs found between Java code fragments obtained from Stackoverflow accepted answers, and 63 Java open source projects.

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

DOI: 10.1145/1235

2. Addressing the problems of reusing online code clones: Our study shows that there are at least 48 clones that have been copied from open source projects to Stackoverflow as code examples which may violate their software license. Moreover, out of 48 clones, there are 27 clones in Stackoverflow that are outdated and dangerous for being reused.

2. RESEARCH QUESTIONS

The study aims to answer the following research questions:
RQ1 (online code clone): *How source code are reused between between Q&A sites and open source projects?* We would like to observe whether this phenomenon has happened and at what scale.
RQ2 (flow of online code clone): *what are the directions source code have been reused?* If the code reuse between the two locations exist, we would like to discover the which direction the code that have been copied. Is it from Q&A site to open source projects, or the other way around, or both?
RQ3 (classification of online code clone): *How and why did these online code clone happen?* Can we categorise them?
RQ4 (effects of online code clone): *Is this phenomenon of online code clone harmful and how?* is there any evidence of problems caused by reusing code between Q&A sites and open source projects?

3. EXPERIMENTAL SETUP

3.1 Dataset

In our study, we selected Qualitas corpus containing 64 Java open source projects [21]. We found that *eclipse* project does not contain source code so we removed it from the dataset. This results in totally 63 projects being analysed. The details of the 63 Qualitas projects with their respective licenses are listed in Table 3.

3.2 Clone Detectors

We selected two clone detectors for this study: Simian [1] and NiCad [4, 17]. **FIXME: Add more info about clone detection tools in general and more details of these two tools**

3.3 Agreement-based Clone Detection

We adopted an idea of clone agreement normally used in clone research [22, 7].

4. RESULTS

The results of running 2 clone detectors: Simian and NiCad, to detect clones between 144,075 Stackoverflow 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 another study [22]. Full Simian's parameter names can be found from the footnote[1].

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.

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

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 5. 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.

4.2 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 ($\text{NiCad}_{EvaClone}$), 4 projects had renaming failed errors and 13 projects had clustering failed errors as depicted in Table 4. So these projects are also missing from agreed clone pairs. **FIXME: Report the errors to NiCad creator.**

FIXME: Maybe no longer needed? We are interested in discovering reused code in the latest versions of Qualitas projects. So, we downloaded the newest release of each project and found 44 of them having newer updates. Then, we reran the experiment again on these 44 projects. Several projects triggered NiCad problem of clustering and renaming again as listed in Table 4. The agreed clone pairs using Bellon's *good*-match(0.7) and *ok*-match(0.7) criteria of this new dataset are also listed in Table 5.

4.3 Manual investigation of agreement-based clone pairs

The classification scheme is described in Table 6 and the classification results are shown in Table 7. 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 $\text{Simian}_{EvaClone}$–$\text{NiCad}_{EvaClone}$ 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 Stackoverflow 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 Stackoverflow answers. We found that all Stackoverflow 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.

4.4 Non-agreement based clone pairs

In the preliminary stage of our experiment, we found that there are 41 Stackoverflow 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 $\text{Simian}_{EvaClone}$ and $\text{NiCad}_{EvaClone}$ due to their large number of false positives as shown in Table 8 and 9. 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 [19].

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 10 and 11.

4.5 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 11.

[1]Simian's parameters: iChar = ignoreCharacters, iCurlB = ignoreCurlyBraces, iId = ignoreIdentifiers, iIdC = ignoreIdentifierCase, iMod = ignoreModifiers
iStrC = ignoreStringCase, iStr = ignoreStrings, iSbtNm = ignoreSubtypeNames, bSqBrck=balanceSquareBrackets

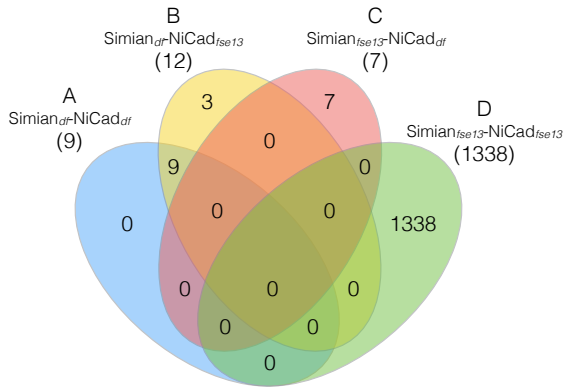

Figure 1: *good*-match(0.7) pairs

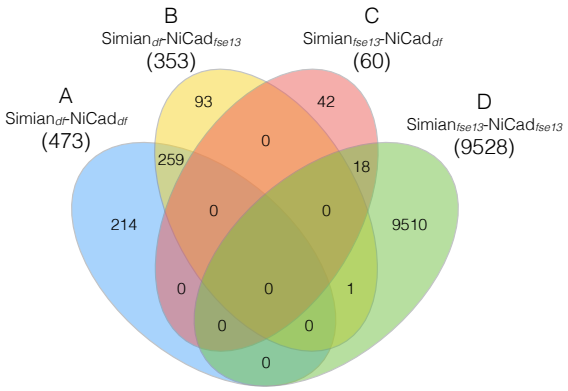

Figure 2: *ok*-match(0.7) pairs

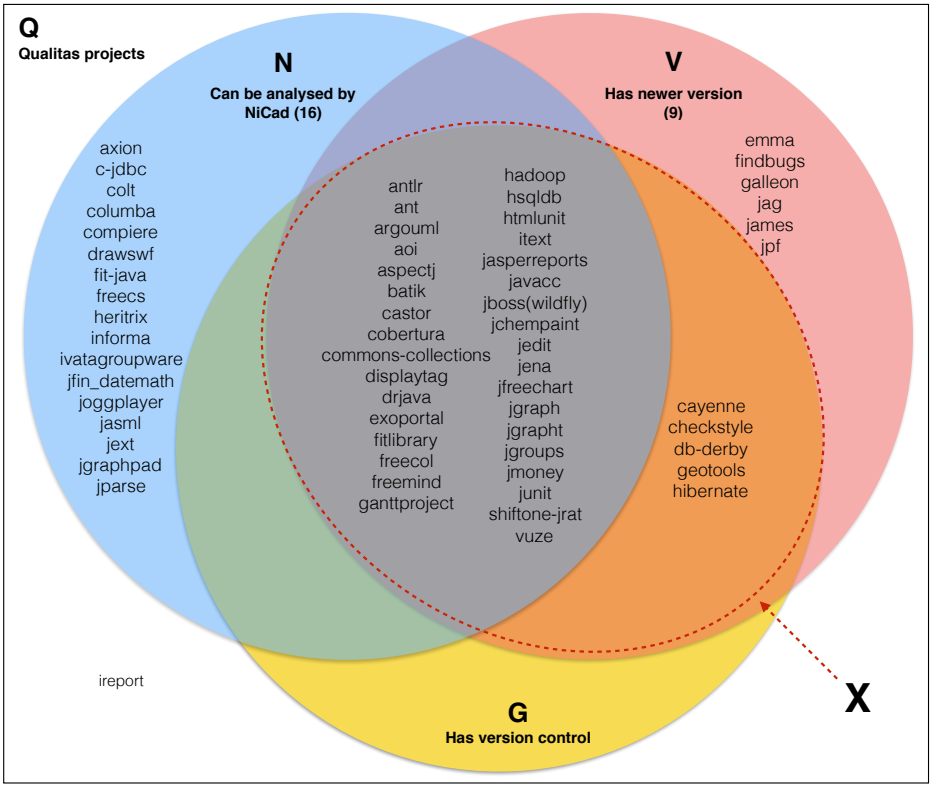

Figure 3: Qualitas projects categorised by NiCad's results, existence of newer versions, and version control

Table 1: Qualitas-*O* (2013-09-01r) clone results

Statistics	Simian Default settings			Simian EvaClone settings			NiCad Default settings			NiCad EvaClone settings		
	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

Table 2: Qualitas-*N* (2016-08-05) clone results (44 new Qualitas projects)

Statistics	Simian Default settings			Simian EvaClone settings			NiCad Default settings			NiCad EvaClone settings		
	707 fragments			1205 fragments			1068 fragments			10231 fragments		
	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$	C_{pairs}	C_{SLOC}	$C_{\%}$
Total	60,536	–	–	22,797,190	–	–	648,165	–	–	573,438,528	–	–
Mean	86	7.37	0.26	18,919	4.85	0.28	607	9.39	0.25	47663	5.22	0.20
Std Dev.	217	2.22	0.21	326,588	1.31	0.19	1,499	2.98	1.81	58,833	2.77	0.15
Max	1,298	32.00	0.94	11,127,286	21.00	0.92	10,550	36.00	0.84	1,246,598	250	0.96
Min	1	5.00	0.01	1	4.00	0.02	1	7	0.02	1	2.00	0.01
Median	2	7.00	0.21	20	5.00	0.25	33	8	0.19	52,077	5.00	0.15
Mode	1	7.00	0.25	1	4.00	0.50	1	8	0.67	1	4.00	0.33

Table 12: Numbers of true online clone pairs (A+A'+B+C) 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

5. 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.

5.1 Issue 1: stale online code

Stale online code occurs when a piece of code has been copied from a software project to Stackoverflow, 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 Stackoverflow. 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 $\rightarrow$ S (class-*A* online clone pairs) in Table 12 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 Stackoverflow and marked as *accepted* answers. The investigation results are described in Table 13.

Table 13: Investigation of stale code clones from the online clone pairs

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

5.2 Issue 2: software licensing violation

Software licensing is vital in software industry. Violation of software license can have 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 [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 Stackoverflow 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.

6. INVESTIGATION OF MISSING A/B CLONE PAIRS REPORTED BY SIMIAN$_{DF}$

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

Projects	Old version	New versions	Latest change	Repo.	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	

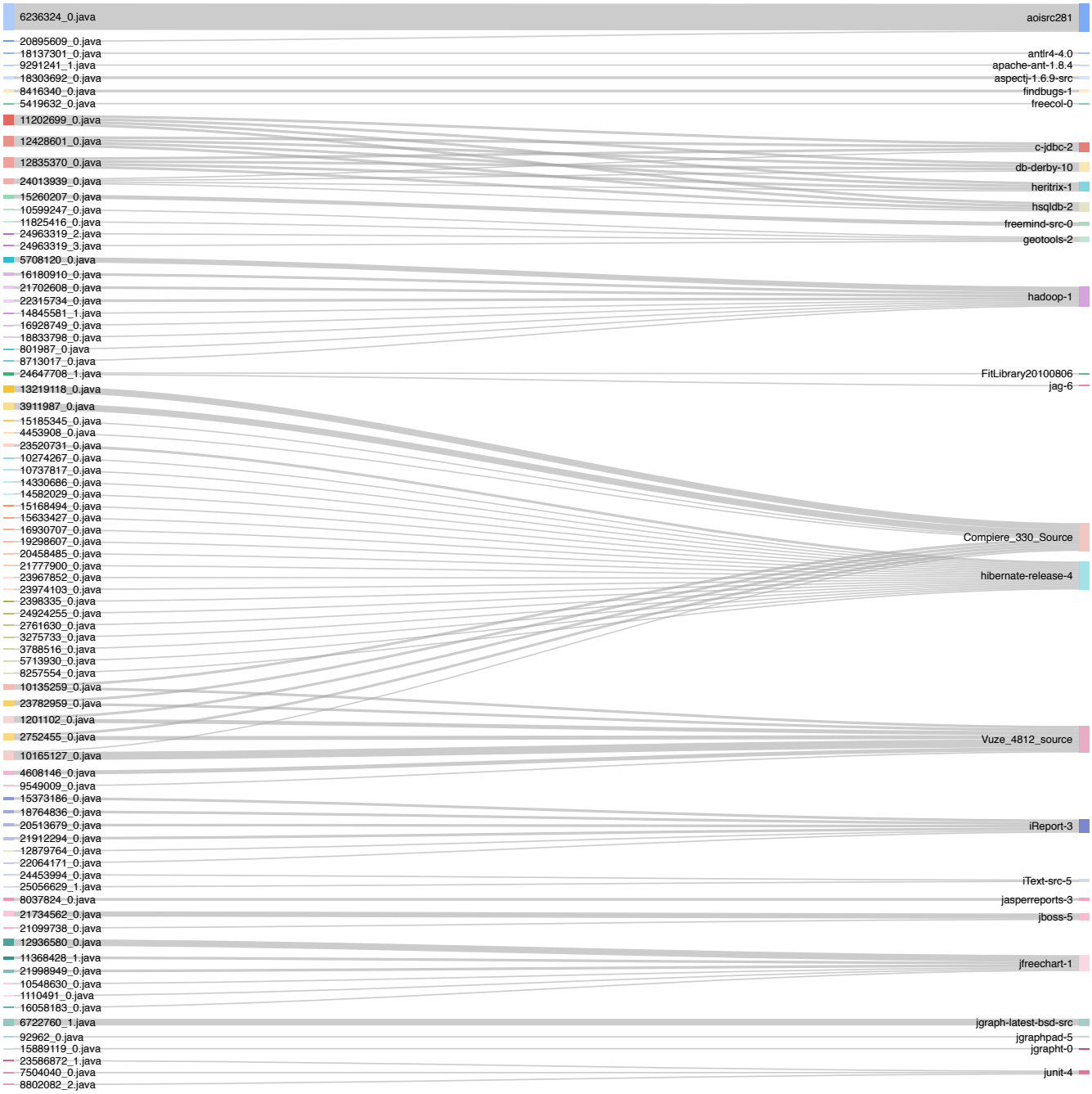

Figure 4: Relationships between 184 true online code clone found between Stackoverflow and Qualitas projects

Table 4: No. of projects in Qualitas Original (Qualitas-O) and New (Qualitas-N) successfully analysed by Simian and NiCad

Qualitas-O			Qualitas-N		
Simian$_{df/EvaClone}$	NiCad$_{df}$	NiCad$_{EvaClone}$	Simian$_{df/EvaClone}$	NiCad$_{df}$	NiCad$_{EvaClone}$
63	57	46	44	40	34
	6 clustering failed	*13 clustering failed*		*4 clustering failed*	*6 clustering failed*
	cayenne	cayenne		jboss (wildfly)	ArgoUML
	checkstyle	checkstyle		hadoop	checkstyle
	db-derby	db-derby		db-derby	db-derby
	geotools	geotools		hibernate	cayenne
	iReport	iReport			jena
	hibernate	ArgoUML			geotools
		castor			
		drjava			
		ganttproject			
		ivatagroupware			
		jasperreports			
		jboss			
		jchempaint			
		4 renaming failed			*4 renaming failed*
		Vuze			Vuze
		aspectj			hadoop
		eXoPortal			jboss (wildfly)
		hibernate			hibernate

Table 5: 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

We investigated the 41 clone pairs previously reported by Simian with default configurations and manually investigated. The 41 pairs were searched for in 4 new results sets: Simian$_{df}$, Simian$_{EvaClone}$, NiCad$_{df}$, NiCad$_{EvaClone}$. The investigation results are shown in Table 15.

The single missing Stackoverflow fragment (19051537_0.java) (denoted by *) is one of the 11 false clones generated by Simian. It is removed from the results of the pretty-printed version because it is an outlier. The rest are missing because of different parameter settings.

7. SIMIAN'S PARAMETERS

We have carefully investigated the effects of the Simian's parameter `-balanceSquareBrackets+`. I found that it works in the expected way of handling a pair of brackets (`[`,`]`) that span over multiple lines. For example, the two code fragments in Figure 5 would match by having `-balanceSquareBrackets+` enabled. However, the `-balanceSquareBrackets+` parameter only works on a small testing environment having toy programs or only small pairs from the full datasets. It does not work with the full complete set of 144,075 Stackoverflow fragments and Qualitas projects. Please find the summary of all the testing scenarios in Table 16.

8. THREATS TO VALIDITY

9. RELATED WORK

- Code clones
  - Definition: Baxter et al. [2]
  - Comparison of clone detectors: [17, 16, 20]
  - NiCad [17, 4]
  - Simian [1]
  - Clone taxonomy [10]
  - Clone evolution [15, 12]
  - Comparing Quality Metrics for Cloned and non cloned Java Methods : A Large Scale Empirical Study [18].
- Agreement-based Clone Detection
  - Bellon's framework [3].
  - EvaClone [22]
  - Hybrid [7]
- Software licensing
  - Code siblings [8], Ninka – Automatic indification of SW license [9], Evolution of SW licensing [5]
- Stackoverflow
  - Code example [13]
  - Search for code in Stackoverflow [6, 11, 14]

10. 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.

Table 6: Classifications of clone creation

Category	Descriptions
A	Code in Stackoverflow is copied from Qualitas (Q → S).
A'	Code in Qualitas is copied from Stackoverflow (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

Table 7: 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

Table 8: Qualitas-*O*: Distribution of classification category A–F according to *good*-match pairs

Category	A	A'	B	C	D	E	F	Total
Simian$_{df}$–NiCad$_{df}$	1	0	1	3	0	4	0	9
Simian$_{df}$–NiCad$_{EvaClone}$	1	0	1	3	1	5	1	12
Simian$_{EvaClone}$–NiCad$_{df}$	0	0	0	0	7	0	0	7
Simian$_{EvaClone}$–NiCad$_{EvaClone}$	0	0	0	0	18	1	1,319	1,338
Total	2	0	2	6	26	10	1,320	1,366
Total (unique)	1	0	1	3	26	6	1,320	1,352

Table 9: Qualitas-*O*: Distribution of classification category A–F according to the *ok*-match pairs

Category	A	A'	B	C	D	E	F	Total
Simian$_{df}$–NiCad$_{df}$	3	0	10	6	433	5	7	464
Simian$_{df}$–NiCad$_{EvaClone}$	8	0	22	4	250	25	32	341
Simian$_{EvaClone}$–NiCad$_{df}$	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 10: Statistics of Simian$_{df}$ and NiCad$_{df}$ clone pairs.

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

Table 11: Classification results of 292 Simian$_{df}$ and 114 NiCad$_{df}$ individual unique 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

```
1   public class MagicSquare {                          public class MagicSquare2 {
2       private int[][] square;                             private int[
3       private boolean[] possible;                                    ][
4       private int totalSqs;                                          ] square;
5       private int sum;                                    private boolean[] possible;
6       private int numsquares;                             private int totalSqs;
7       public static void main ( String[] args ) {         private int sum;
8       MagicSquare m = new MagicSquare ( 3 );              private int numsquares;
9                                                           public static void main ( String[] args ) {
10                                                          MagicSquare m = new MagicSquare ( 3 );
```

Figure 5: Two identical fragments with only differences in locations of the square brackets. All 7 lines are reported by Simian if `-balanceSquareBrackets+` is enabled. If not, the clone pairs is reported as (MagicSquare.java [3,8], MagicSquare2.java [5,10]).

Table 14: 30 stale code clones found by a manual investigation

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

Table 15: Results of matching the original 41 Simian(default) pairs in the pretty-printed result sets

Settings	Found	Not found
Simian_{df}	40	1*
$\text{Simian}_{EvaClone}$	0	41
NiCad_{df}	17	24
$\text{NiCad}_{EvaClone}$	24	17

11. 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.

12. 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.

[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] 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.

[11] 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.

[12] 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. IEEE, 2011.

[13] 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.

[14] 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. 41(3):727–759, dec 2014.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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 16: Simian's `-balanceSquareBrackets+` (`-bsb+`) is observed to have unpredictable behaviours when running against big datasets. CP_2 means the reported clone(s) do not contain lines having dislocated brackets (L_b) (i.e. $CP_2 = CP_1 - L_b$).

Project 1	Project 2	Dislocated brackets?	-bsb+	Clones pair reported
Only run Simian against the pair				
MagicSquare.java	MagicSquare_exact_copy.java	no	0,1	CP_1
MagicSquare.java	MagicSquare2.java	yes	0	CP_2
MagicSquare.java	MagicSquare2.java	yes	1	CP_1
stackoverflow/4298836_0.java	Qualitas/aoisrc281/../ExprModule.java	no	0	CP_3
stackoverflow/4298836_0.java	Qualitas/aoisrc281/../ExprModule.java	no	1	CP_3
stackoverflow/4533682_1.java	Qualitas/cobertura-1/../TouchCollector.java	no	0	CP_4
stackoverflow/4533682_1.java	Qualitas/cobertura-1/../TouchCollector.java	no	1	CP_4
Run Simian against the complete stackoverflow data and the project				
stackoverflow/4298836_0.java	Qualitas/aoisrc281/../ExprModule.java	no	0	CP_3
stackoverflow/4298836_0.java	Qualitas/aoisrc281/../ExprModule.java	no	1	–
stackoverflow/4533682_1.java	Qualitas/cobertura-1/../TouchCollector.java	no	0	CP_4
stackoverflow/4533682_1.java	Qualitas/cobertura-1/../TouchCollector.java	no	1	–