Online Source Code Reuse: A Case Study of Stackoverflow *vs.* Qualitas OSS

RAW STATISTICS

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 three different settings: default settings, settings from FSE'13 paper, and settings from CloPlag experiment. Full Simian's parameter names can be found from the footnote[1].

Manual investigation of Simian's clone report showed that there were problematic fragments (6 for defaults, 0 for FSE'13, and 11 for CloPlag). These fragments generate false clone containing array initialisation. Hence, they were removed from the result set before analysis.

Table I: Simian clone results

	Default settings threshold=6 iCharC,iCurlB,iIdC,iMod,iStrC				Default settings* threshold=6 iCharC,iCurlB,iIdC,iMod,iStrC,bSqBrck				FSE'13 settings threshold=5 iCurlB,iId,iIdC,iStr,iChar,iSbtNm,bSqBrck			
	Frag.	C_{pairs}	C_{SLOC}	$C_{\%}$	Frag.	C_{pairs}	C_{SLOC}	$C_{\%}$	Frag.	C_{pairs}	C_{SLOC}	$C_{\%}$
Removed	6	–	–	–	0	–	–	–	0	–	–	–
Total	647	24,802	–	–	3	10	–	–	152	7,289,272	–	–
Mean	–	38	7.53	0.27	–	3.33	8	0.29	–	47,956	4.76	0.27
Std Dev.	–	87	3.87	0.22	–	2.62	1.41	0.18	–	433,467	0.93	0.18
Max	–	551	63	0.94	–	7	10	0.54	–	5,278,408	8	0.83
Min	–	1	5	0.01	–	1	7	0.10	–	1	4	0.01
Median	–	3	7	0.22	–	2	7	0.24	–	22.5	4	0.23
Mode	–	1	7	0.25	–	N/A	7	N/A	–	1	4	0.50

Table II: NiCad clone results

	Default settings threshold=6, UPI=0.30 rename=none,abstract=none				FSE'13 settings threshold=5, UPI=0.2 rename=blind,abstract=literal			
	Frag.	C_{pairs}	C_{SLOC}	$C_{\%}$	Frag.	C_{pairs}	C_{SLOC}	$C_{\%}$
Total	983	106,091	–	–	8,100	31,141,599	–	–
Mean	–	108	9.48	0.25	–	3,845	5.52	0.19
Std Dev.		200.49	3.04	0.18	–	4,305	2.49	0.15
Max	–	1816	39	0.80	–	46,469	111	0.88
Min	–	1	7	0.02	–	1	3	0.01
Median	–	14	8	0.19	–	3,095	5	0.14
Mode	–	1	8	0.53	–	1	4	0.33

AGREED CLONED FRAGMENTS

The results from both tools are compared to find clone pairs that they mutually agree. Theses agreed clone pairs will be manually investigated. The results of agreed clone pairs found by different parameter settings of Simian and NiCad are shown in Table III.

Table III: Agreed cloned fragments between Simian and Nicad

	Simian$_{default}$	Simian$_{fse13}$
NiCad$_{default}$	92	1
NiCad$_{fse13}$	155	25

Relationships between the cloned fragments are displayed in Fig. 1.

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

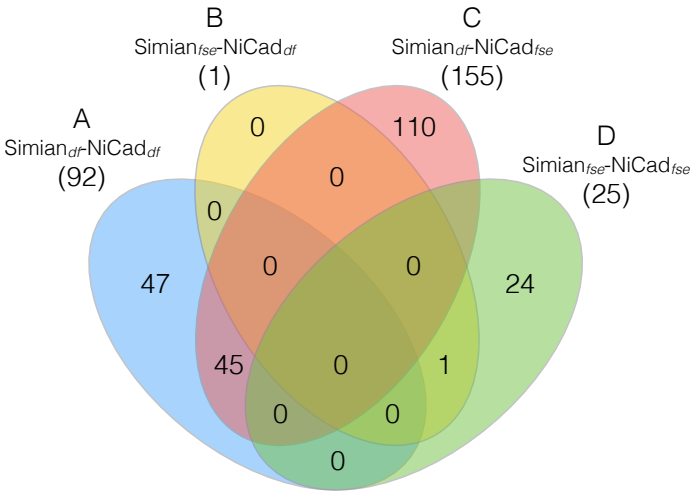

Fig. 1: Agreed clone fragments reported by Simian and NiCad with default and FSE'13 settings

AGREED CLONE PAIRS

The investigation of the clone pairs using Bellon's *good-match(p)* and *ok-match(p)* criteria are listed in Table IV and Table V. We use the threshold p of 0.7 for both *good-match* and *ok-match*. Visualisation of *good-match* common clone pairs between sets of parameter settings can be seen from Figure 2. There are 14 *good-match* pairs. The *ok-match* subsumes the *good-match* resulting in totally 528 common clone pairs. I think I can manually investigate all of the *good-match* ones and 86 samples of the *ok-match* ones.

Table IV: Agreed good-match(0.7) cloned pairs between Simian and Nicad

	Simian$_{default}$	Simian$_{fse13}$
NiCad$_{default}$	9	1
NiCad$_{fse13}$	3	5

Table V: Agreed ok-match(0.7) cloned pairs between Simian and Nicad

	Simian$_{default}$	Simian$_{fse13}$
NiCad$_{default}$	373	1
NiCad$_{fse13}$	118	36

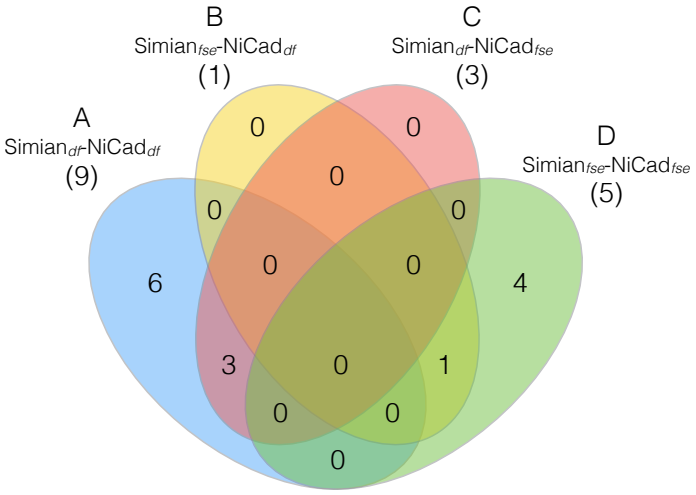

Fig. 2: Agreed clone pairs reported by Simian and NiCad with default and FSE'13 settings using good-match(0.7) criterion

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 3 would match by having `-balanceSquareBrackets+` enabled.

```
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 );
```

Fig. 3: 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]).

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 VI.

Table VI: 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	–