Volume 2, Issue 5, May 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparison of Clone Detection Tools: CONQAT and SolidSDD

Prabhjot Kaur*, Harpreet Kaur, Rupinder Kaur,
department of computer engineering
UCOE,Punjabi university
Patiala
sidhu.965@gmail.com

Abstract: Copying and pasting the code fragment leads to the term so called code clone. This becomes a major issue in large software systems. Clones can be roughly classified into four types-type1 clones, type 2 clones, type3 clones, and type 4 clones. In this paper we are going to compare two clone detection tools i.e. CONQAT and solidSDD on the basis of various clone metrics. solidSDD is an application for detecting and managing duplicate code. It can be used to analyse large projects and detect code that has been cloned (e.g., via cut-n-paste operations) during development. Identifying such code fragments can be very useful for facilitating development and maintenance, or for reducing the memory footprint of an application.

***Keywords*— Clone, CONQAT, SolidSDD, SolidSX, Clone metrics,**

I. INTRODUCTION

Clones are identical or near identical segments of source code. Code clones are usually intentionally created through copying another piece of code. However, in certain cases [3] clones appears unintentionally due to code segments using the same APIs. Code cloning is a common phenomenon in the development of large software systems. It is reported that 5-50% of large software systems are clones [2, 4, 1].Large scale software systems are expensive to build and are even more expensive to maintain. Sometimes, developers take easier way of implementation by copying some fragments of the existing programs and use that code in their work. This kind of work is called code cloning.

A.Code cloning:

A clone is a segment of code that has been created through duplication of another piece of code. Clones share similar code structures. However, since the size and the degree of similarities among code segments vary, code cloning is a fairly subjective concept. It depends on the context or human judgment whether it is a code clone or not. When referring to clone relations, we use two terms: clone pairs and clone classes. A clone pair is a pair of code segments which are identical or similar to each other. A clone class is the maximum set of code segments in which any two of the code segments forms a clone pair. disk or other media, or over a network. For example, Acronis True Image and Symantec Ghost are popular commercial products to clone PCs.

B. Code Clone types

There are basically two categories of similarities between two code fragments. Two code fragments can be

similar based on the similarity of their program text or they can be similar in their functionalities without being textually similar. The first category of clones are often the result of copying a code fragment and then pasting to another location. In this section, we consider clone types based on the kind of similarity two code fragments can have:

1) Textual Similarity: Based on the textual similarity we distinguish the following types of clones [6, 5]:

Type I: Identical code fragments except for variations in whitespace (may be also variations in layout) and comments.

Type II: Structurally/syntactically identical fragments except for variations in identifiers, literals, types, layout and comments.

Type III: Copied fragments with further modifications. Statements can be changed added or removed in addition to variations identifiers, literals, types, layout and comments.

2) Functional Similarity: If the functionalities of the two code fragments are identical or similar i.e., they have similar pre and post conditions, we call them semantic clones [7, 8, 9, 10] and referred as Type IV clones.

Type IV: Two or more code fragments that perform the same computation but implemented through different syntactic variants. with libraries as well [11].

Thus, the uses of similar APIs or libraries may introduce clones.

C. Code clone terminologies

1) Clone Pair: A pair of code portion/fragments is called clone pair if there exists a clone relation between them.

2) Clone class: A clone class is the maximal set of code fragments in which any two of the code fragments holds a clone relation i.e. form a clone pair. A clone class is therefore, the union of the clone pairs which have code portions in common. Clone classes are also called clone communities [12].

3) Clone Class Family: The group of all clone classes that have the same domain is called clone class family [13]. Such a clone class family is also termed as super clone. In their context, multiple clone classes between the same source entities are aggregated into one large super clone.

Fragment 1 …	Fragment 2 …	Fragment 3
For(int i=1;i<n;i++){ Sum=sum+i; } a	For(int i=1;i<n;i++){ Sum=sum+i; } a	…
If(sum<0){ Sum=n-sum } b	If(sum<0){ Sum=n-sum }。 b	If(result<0){ result=m-result } a
	While(sum<n){ Sum=n/sum } c	While(result<m){ result=m/result } c
	…	…

Fig 1. Clone pair and Clone classes

II. CLONE DETECTION TOOL: CONQAT

In CONQAT the clone detection process is divided into different phases: Input & pre-processing, detection, post-processing, output. User can build her own configuration using CONQAT's graphical configuration tool. Input and output directories have been modeled as parameters. This figure 2 is taken as snapshot from configuration file of clone detection in java. In input phase the code is read and required program representation is created. In detection phase similar code segments are identified. Then the results are post-processed and presented to users in various formats. The detection phase mainly consists of a single processor performing the core detection algorithm, which might be parameterized using additional constraints. The detection result is then filtered and the redundancy free source statements (RFSS) are calculated, before in the output phase clone coverage reports and statistics are generated as HTML pages and an XML clone report is written as input for further tools in the tool chain. Several settings, such as the minimal length of detected clones are given as parameters to the processors. The detection process of CONQAT searches for similar subsequences in the program unit sequence created by the input phase. Currently, two different algorithms are available. Both work on an optimized suffix-tree created from the program unit sequence

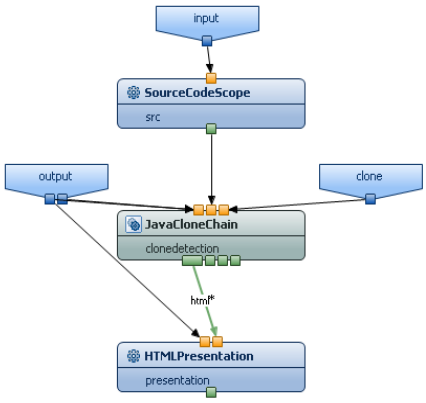

Fig 2. Clone detection configuration

. Ungapped clone detection searches for identical sub-sequences, thus detecting code fragments that only differ (depending on normalization settings) in whitespace, identifier names, comments or literal values. Detection is simply performed by suffix tree traversal and runs in linear time and space. The novel Gapped clone detection additionally detects clones with statement modifications, additions or removals. The performance of these algorithms depends on the analyzed systems and overall detection configuration. Both are capable of analysing systems of several MLOC which is sufficient for most industrial systems. CONQAT is both configurable and extendible so it is considered ideal platform for research and experiments. It provides a dedicated compare-view that shows cloned code fragments side-by-side, highlighting inconsistencies, as depicted in Figure 3. Numerous filters are available to manage large clone reports without having to re-execute detection. Deissenboeck et al. [17] have extended the CONQAT tool suite to detect exact subsystem clones in simulink models. They detect clone pairs using a breadth-first search of the model graph and an inspection of the nodes' neighbourhoods. Clone pairs are then clustered into clone classes by representing the pairs in a graph and identifying sets of connected pairs as clusters

Fig 3. Clone view of CONQAT

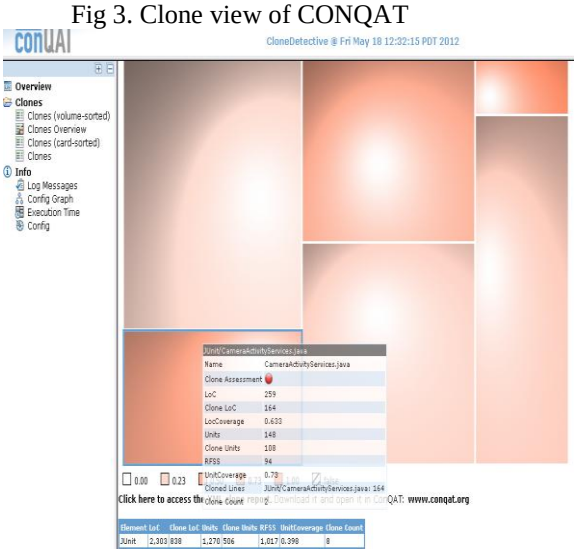

Fig 4. Clone detection result overview

III. TOOLSET:SolidSDD AND CONQAT

SolidSX and SolidSDD are provided as self-contained installers, freely available for research [14] on Windows XP and later editions. The SolidSX tool supports the analysis of software structure, dependencies, and metrics. It gives basic metrics like code and comment size, complexity, fan-in, fan-out, and symbol source code location. Clone detection uses the same algorithm as CCfinder [15], configurable by clone length (in statements), identifier renaming (allowed or not), size of gaps (inserted or deleted code fragments in a clone), and whitespace and comment filtering. Nodes and edges have metrics, e.g. percentage of cloned code, number of distinct clones, and whether a clone Includes identifier renaming or not. Figure 5 shows the text view of SolidSDD. The table shows all files with percentage of cloned code, number of clones, and presence of identifier renaming. Sorting this table allows e.g. finding files with the most clones or highest cloned code percentage.

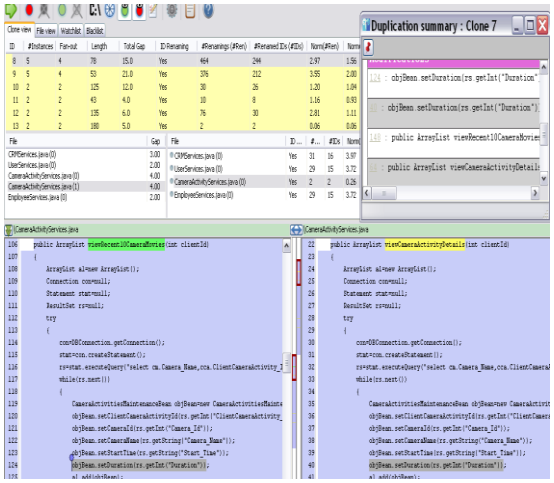

Fig 5. Clone view of SolidSDD

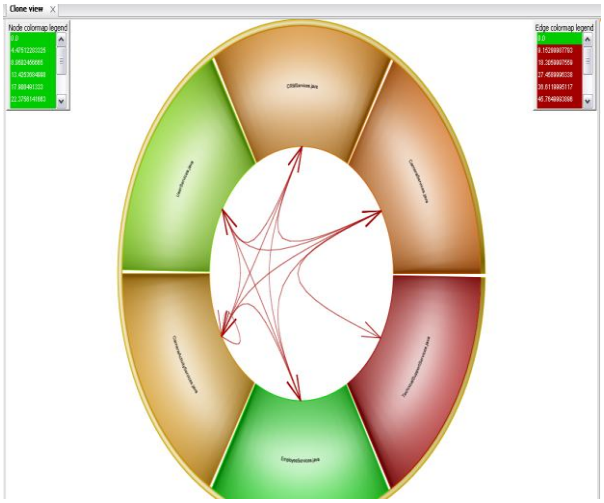

Fig 6. Graphical view of solidSX

IV. CLONE METRICS FOR COMPARING TOOLS

These are some clone set metrics of clone class on the basis of which we compare the tools i.e. CONQAT and SolidSDD. The clone set metrics are:

A. Population of clone class (POP) [16]: Population of a Clone Class; POP is the number of elements of a given clone class C. A clone class with a large POP means that similar code portions appear in many places. Higher POP(S) values mean that code clones in a clone set appear more frequently in the system. Contrary to this, lower POP(S) values mean that code clones in a clone set appear in fewer places is the system.

B. RNR(S) [16]: RNR(S) is the ratio of non-repeated token sequences of code clones in clone set S. Higher RNR(S) values mean that each code clone in a clone set S consists of more non-repeated token sequences. Contrary to this, lower RNR(S) value means that each code clone in a clone set s consists of more repeated token sequences. In most cases, repeated code sequences are involved in language-dependent

code clones. *LOSwhole(ci)* is the Length Of the whole token Sequence of code clone *ci*. *LOSnon-repeated*(*ci*) is the Length Of non repeated token Sequence of code clone *ci*,

$$RNR(S) = \frac{\sum_{i=1}^{n} LOS_{non-repeated}(c_i)}{\sum_{i=1}^{n} LOS_{whole}(c_i)} \times 100$$

C. LEN [16] : LEN(S) is the average length of token sequences of code clones in clone set S. Higher LEN(S) values mean that each code clone in a clone set S consists of more token sequences. Contrary to this, lower LEN(S) values mean that each code clone in a clone set S consists of less token sequences and the size of code fragments are smaller.

*D. Execution Time***:** The time it takes the tool to execute the code and find clones.

E. #clones: The number of clones a tool finds in a file.

F. Gaps: If there are any insertions or deletions in a code then it is called as gaps. Code example is as follows:

```
If (b < = c) {
        d = e + c; //comment 1
        e = e + 1;
else
        d = e- b; //comment 2
```

code this segment after adding one statement could be as follows:

```
if (b < = c) {
d = e + c; //comment 1
        f = 2;
        e = e + 1;
else
        d = e – b; //comment 2
```

Tools / Features	solidSDD	CONQAT
Languages suppoted	C, C++, java, C#	ABAP, ADA, C++, C, java, COBOL
Domain	Clone detection	Clone detection
Requirements	No requirements	Java1.6, graphviz2, Microsoft.net2.0
Submission method		
Source data	Programming language, files	files
Result output	Source code, graphical view	Repots, clone compare view
Metrics produced	Clone metrics	Clone metrics, line metrics

Table1. Features of tools

This table shows the features offered by both the tools. Both the tools have different features with some fatures in common.

V.EXPERIMENTAL RESULTS

Metrics / Tools	Clones	Gaps	RNR	POP	LEN	Extn time
SolidSDD	9	2	5.2	386	80	1s
CONQAT	2	0	7	398	82	20s

Table 2. Comparing clone set metrics

Table 2, represents clone metrics comparison of both the tools in which solidSDD shows more clones as compare to CONQAT with less non repeated tokens value and taking less execution time.

VI.CONCLUSION

In this paper, we have compared two clone detection tools i.e. CONQAT and solidSDD on the basis of various clone set metrics. As solidSDD take less time than CONQAT to execute the code and find the clones, it gives more clones and even shows gaps also i.e. any modification in the code taking same length of the clone as of CONQAT. So we conclude that solidSDD is better than CONQAT. It doesnot require any additional environment to work as CONQAT requires.

REFERENCES

[1] Bruno Lague, Daniel Proulx, Jean Mayrand, Ettore Merlo, John P. Hudepohl. Assessing the Benefits of Incorporating Function Clone Detection in a Development Process. In Proceedings of the International Conference on Software Maintenance (ICSM1997), pages 314–321, 1997.

[2] B.S. Baker. On finding duplication and near duplication in large software system. In Proceedings of Second IEEE Working Conference on Reverse Eng. (WCRE 1995), July 1995.

[3] Raihan Al-Ekram and Cory Kapser and Richard Holt and Michael Godfrey. Cloning by Accident: An Empirical Study of Source Code Cloning Across Software Systems. In Proceedings of the 2005 Intl. Symposium on Empirical Software Engineering (ISESE- 05), Noosa Heads, Australia, 2005.

[4] S. Ducasse and M. Rieger and S. Demeyer. A language independent approach for detecting duplicated code. In Proceedings of IEEE International Conference on Software Maintenance (ICSM 1999), August 1999.

[5] Stefan Bellon. Detection of Software Clones Tool Comparison Experiment. Tool Comparison Experiment presented at the 1st IEEE International Workshop on Source Code Analysis and Manipulation, Montreal, Canada, October 2002.

[6] Stefan Bellon.Vergleich von techniken zur erkennung duplizierten quellcodes. Diploma Thesis, No. 1998, University of Stuttgart (Germany), Institute for Software Technology, September 2002.

[7] Raghavan Komondoor and Susan Horwitz. Effective, Automatic Procedure Extraction. In Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03), pp. 33-42, Portland, Oregon, USA, May 2003.

[8] Jens Krinke. Identifying Similar Code with Program Dependence Graphs. In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE'01), pp. 301-309, Stuttgart, Germany, October 2001.
[9] Gilad Mishne and Maarten de Rijke. Source Code Retrieval Using Conceptual Similarity. In Proceeding of the 2004 Conference on Computer Assisted Information Retrieval (RIAO'04), pp. 539-554, Avignon (Vaucluse), France, April 2004.
[10] Neil Davey, Paul Barson, Simon Field, Ray J Frank. The Development of a Software Clone Detector. International Journal of Applied Software Technology, Vol. 1(3/4):219- 236, 1995.
[11] Eytan Adar and Miryung Kim. SoftGUESS: Visualization and Exploration of Code Clones in Context. In the proceedings of the 29th International Conference on Software Engineering (ICSE'07), Tool Demo, pp.762-766, Minneapolis, MN, USA, May 2007
[12] Jean Mayrand, Claude Leblanc, EttoreMerlo.Experiment on the AutomaticDetection of Function Clones iSoftware System Using Metrics. In Proceedings ofSannella, M. J. 1994 Constraint Satisfaction and Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
[13] Ira Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant Anna. Clone Detection Using Abstract Syntax Trees. In Proceedings of
[14] the 14th International Conference on Software Maintenance (ICSM'98).
[15] SolidSource BV, SolidSX, SolidSDD, SolidSTA, and SolidFX tool distributions, www.solidsourceit.com, 2010.
[16] T. Kamiya, S. Kusumoto, K. Inoue, CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large-Scale Source Code, IEEE TSE 28 (7) (2002) 654–670.
[17] F. Deissenboeck, B. Hummel,E.Jurgens,B.Schatz, S.Wagner, J. Girard, and S. Teuchert, "Clone detection in automotive model-based development," in ICSE, 2009, pp. 603–612.

