Common Techniques and Tools for the Analysis of Open Source Software in Order to Detect Code Clones: A Study

Hamed Jelodar[1], Javad aramideh[2]
(Corresponding author: Hamed Jelodar)

Department of Computer, Science and Research, Islamic Azad University, Bushehr, Iran[1]
(Email :Jelodarh@gmail.com)

Department of Computer, Islamic Azad University, Sari branch, Iran[2]

(Received May 27, 2014; revised and accepted June 29, 2014)

Abstract

Code clone detection and its elimination are introduced as an influential factor in improving software quality, especially in large software systems. The code that duplicates and is in similar files of source is called code clone. So far, researchers introduced and presented many tools and techniques to analyze and identify clones. But given the importance of this paper, some common techniques and tools used to identify clones should be examined and 4 open-source software systems are analyzed using CloneDr tool. According to the results, many clones were identified and the results show that further improvements are needed for high quality software.

Keywords: code clone, clone detection, clone detection tool, analysis

1 Introduction

Several factors influence the reduction or decrease of software quality. One factor that reduces the quality of software is the existence of code clone is a resource file. It is said that code clone is defined differently and clone means the duplicated codes in a source file. Identifying clones renders gives a substantial assistance to improve the quality of software. Generally, code clones are divided into four types, which referred in the next section. Many different techniques have already been proposed to detect code clones and the most common include Text-based, Token-Based, ASSET-Based, PDG-Based. Also, in order to analyze and identify clones several tools have been introduced to implement and the most popular tools are CCFinder, CloneDr. In this paper, some common techniques and tools to identify clones are examined and finally (4) open source software systems are analyzed. In the second part we will continue with a review of earlier works and in the third and fourth parts the clone is introduced and some common techniques and tools are examined to identify and analyze the clones. The fifth section analyzes four software systems using the tool "CloneDr". Finally, in the sixth Section the conclusions are discussed.

2 Previous Works

Code clones in software systems lead to lower quality of software. Since it is not possible to easily identify the clone, researchers have developed various tools and techniques to detect clones. This section will point out some of the works associated with the topic. Kamiya and colleagues [1].presented a method for the detection of code clones using token-based technique. Burd and colleagues [2] tried to measure and evaluate clone detection tools and found that each have strengths and weaknesses and suggested a combination of tools to perform analysis. Also, some researchers evaluated open source software systems such as Saha and colleagues who evaluated 17 open source software systems with languages C, Java, and C + + [3]. Calefato and colleagues offered a semi-automated method to identify clones in web applications and the results show that the proposed approach effectively and efficiently to detect clones in web applications [4].

3 Clone Code

One of the major problems in software system is code clones and it is more common in large systems. Code clones are duplicated codes which are useless in software source code [5,6].Code clone is classified into many different types and several techniques have been introduced to identify them and several tools were implemented to isolate clones. Code clone detection applications benefits can include: improving software processes, detecting duplicate code, detecting plagiarism and copyright infringement pointed out [7].

Code clones are generally divided into four categories which include 1. Type one. 2. Type two. 3. Type three. 4. Type four and some techniques are introduced to identify any of these types.

Type 1: Pieces of code that are identical except for whitespace, layout and comments are different.

Type 2: Parts are the same except in writing and identifiers, literals, types, whitespace, layout and comments are different.

Type 3: More changes will be copied components, for example, to change, add or delete the comments, in addition to the name change, literals, types, whitespace.

Type 4: Two or more pieces of code that do the same calculation, but with different methods are carried out.

4 Clone Detection Techniques and Tools

In this section we will introduce some of the techniques and tools to identify clones.

4.1 Common Techniques

There are several techniques for Code clone detection that can be very effective in detecting code clones, introducing four common techniques to detect clones [8].

In Figure 1, some of the most popular tools for identifying clones with respect to the existing methods are shown.

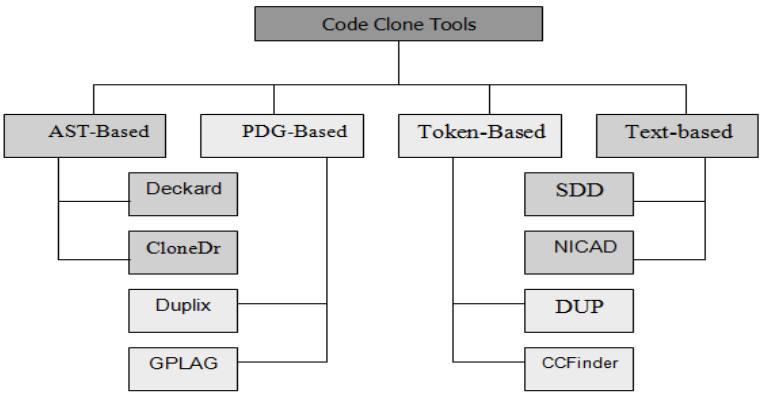

Figure 1. Some popular tools for identifying clones

1) Text-based: In this way, the intended source code is considered as a sequence of lines or strings and two pieces of code to find a sequence of text strings that are similar are compared. When two or more pieces of code had the highest degree of similarity, it will be returned as a clone or clones of a class (as a result). Typically, in this method whitespace and comments are not considered[9].

2) Token-Based: In this method, lexical analysis of source code is discussed, and then the token is to be used as a basis for clone detection [10]. This token can be identifier and literal. This method can detect various types of clones [11,8].

3) AST-Based: AST-Based compares the same sub-trees. Tree-based methods provide a tree graph of a summary of source code and then search the clone detection algorithm under a tree similar to this "cognitive summary graph" (AST) [12,13].

4) PDG-Based: This method uses the semantic approach in order to identify clones uses. PDG includes control flow and data flow information of a program[14].

4.2 Available Tools to identify clones

Different tools have been designed for clone detection with different types. While a few of them have been commercialized, many of them are currently used for research purposes to help the software development and maintenance processes. As shown in Figure 1, each of the instruments uses techniques to detect clones. These tools will help us to analyze software systems with multiple languages (such as C #, JAVA) and find the code clones. Some of the most popular of these tools include CCFinder, SimCad, CloneDR, JPlag.

1) Clone detection by CCFinder: CCFinder is one of the tools to detect duplicated codes that are called code clones and are used for the analysis of large-scale systems. With this software, the files source with different languages can be analyzed and these languages include: Java, C / C + +, COBOL, VB, C # [15].

2) Clone detection by CloneDr: CloneDr is a tool that is used to detect code clones. With this tool, you can analyze thousands or millions of lines of code or files. This tool can be used to run the Windows operating system. This tool supports various languages including: Java, C #, C + +, COBOL, JavaScript, PHP [6]. In this paper, using this software we analyze multiple open source software systems, which version of C # is chosen for analysis.

3) Clone detection by Jplag: Jplag is a system that can detect duplicated code in a code source. The tool has a powerful graphical interface to present the results. Currently this tool supports Java, C #, C, C + +, Scheme and natural language text [16].

4) Clone detection by SimCad: This tool uses hashing technique to detect code clones and runs through the Eclipse software. SimCad has been used successfully in various fields of research such as Web Mining, text retrieval, etc. [17].

5. EXPERIMENTS AND RESULTS

In this section we evaluated and analyzed multiple open source software systems to detect code clones. To do this, we have analyzed 4 Open Source Software Systems with the3 programming language of C #.

Table 1 shows the information systems that have been selected for analysis And Table 2 is the meanings of the used terms and table 3 is the overall results obtained from the analysis of the software.

Table 1. Systems Information

S.No	Sysytem	Language	Total Size	Version
1	OpenCL.Net	C#	2.86 M	0.6.3
2	OpenPop.NET	C#	2.31 M	2.0.5
3	id3lib	C#	9.82 M	0.6
4	CANopen	C#	9.86 M	0.85

Table 2. Meanings of used terms

Meanings of the terms	
Meanings	Term
FC	File Count
TSLOC	Source Lines of Code
TCS	Total ColneSet

Table 3. Overall results obtained from the analysis of software

N.System	FC	SLOC	TCS
OpenCL.Net	50	10835	91
OpenPop.NET	71	14398	146
id3lib	254	38986	92
CANopen	83	14398	60

As mentioned in the previous section, there are several techniques to detect cloned code and each of these tools and techniques are introduced that can be used to detect cloned code. To perform this test, we have the tools CloneDr disposal. Figure 2 is an example of the code clone detection system software is OpenCLNet, according to Figure 5 to 8 lines of code clone "Mem.cs" is detected. The analysis was carried out; OpenPop file size of 71 and 140 clones that code has been detected. Id3lib with 254 files as well as 92 clones were identified code. 50 files as well as OpenCL code, we found 91 clones. The CANopen File 83 of 63 clones was found.

Clone Instance	Line Count	Source Line	Source File
1	8	132	.../OpenCLNet/Mem.cs
2	8	142	.../OpenCLNet/Mem.cs
3	8	152	.../OpenCLNet/Mem.cs
4	8	162	.../OpenCLNet/Mem.cs
5	8	172	.../OpenCLNet/Mem.cs
Code Clone			
1	public virtual void Write(CommandQueue cq, long dstOffset, [[#variable75fca00]][] srcData, int srcStartIndex, int count){		
2	IntPtr p = cq.EnqueueMapBuffer(this, true, MapFlags.WRITE, dstOffset, [[#variable75f9460]]);		
3	[[#variable75fca00]]* pBlock = ([[#variable75fca00]]*)p.ToPointer();		
4	for (long i = 0; i < count; i++)		
5	pBlock[i] = srcData[i + srcStartIndex];		
6	cq.EnqueueUnmapMemObject(this, p);		
7	cq.Finish();		
8	}		

Figure 2. An example of an identified clone

Initial parameters of the instruments used for this analysis is shown in Figure 3 shows. The full results are shown in Figure 4 are obtained for the software (id3lib) are shown. Figure 5 also analyzed the results for the four open-source software, including the amount of code clones are detected and analyzed to show the number of files.

Detection Parameters	
Value	**Value**
Similarity Threshold	95%
Maximum parameter count	6
Minimum Mass (Lines)	6.0
Characters per node	16
Starting height	2

Figure 3. Initial parameters for analysis

Clone Detection Statistics	
Statistic	**Value**
File Count	245
Total Source Lines of Code (SLOC)	38986
Estimated SLOC before preprocessing	39195
Expanded SLOC after preprocessing	38906
Total CloneSets	92
Exact-match CloneSets	57
Near-miss CloneSets	35
Number of cloned SLOC	37163
SLOC in clones %	94.8%
Estimated removable SLOC	23350
Possible SLOC reduction %	59.6%
Possible SLOC reduction in expanded file %	60.0%

Figure 4. Results for id3lib

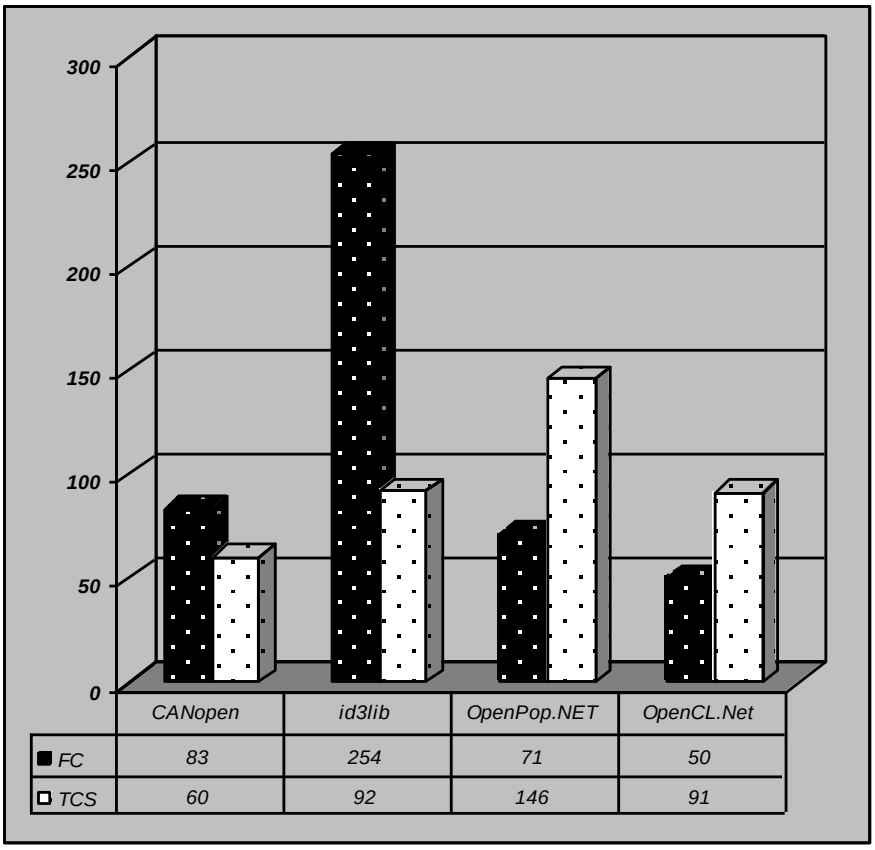

Figure 5. The overall results of the analysis of four open-source software

6 Conclusion

This paper deals with the code clone detection methods and some of the techniques and tools used to identify clones of code has been introduced and the case study analysis was a few open-source software systems to detect cloned code. The results of this study show that despite the immense complexity of the code clone detection in software systems, if there are common techniques used to clone overshadowed the quality of software code that can be attempted in order to identify clones. It is also hoped that the developers of software systems factors (such as quality programming) that creates a special attention should be cloned.

References

[1] Kamiya, Toshihiro, Shinji Kusumoto, and Katsuro Inoue. "CCFinder: a multilinguistic token-based code clone detection system for large scale source code." Software Engineering, IEEE Transactions on 28.7 (2002).

[2] Burd, Elizabeth, and John Bailey. "Evaluating clone detection tools for use during preventative maintenance." Source Code Analysis and Manipulation, 2002. Proceedings. Second IEEE International Workshop on. IEEE, 2002.

[3] Saha, Ripon K., et al. "Evaluating code clone genealogies at release level: An empirical study." Source Code Analysis and Manipulation (SCAM), *2010 10th IEEE Working Conference on. IEEE*, 2010.

[4] Calefato, Fabio, Filippo Lanubile, and Teresa *Mallardo. "Function clone detection in web applications: A semiautomated approach." Journal* of Web Engineering 3 (2004).

[5] Krishnan, Giri Panamoottil, and Nikolaos Tsantalis. Unification and refactoring of clones. *CSMR-18/WCRE-21 Software Evolution Week*, 2014.

[6] Ueda, Yasushi, et al. "Gemini: Maintenance support environment based on code clone analysis." Software Metrics, 2002. *Proceedings. Eighth IEEE Symposium on*. IEEE, 2002.

[7] Morshed, Md, Md Rahman, and Salah Uddin Ahmed. "A Literature Review of Code Clone Analysis to Improve Software Maintenance Process." arXiv preprint arXiv:1205.5615 (2012).

[8] Schulze, Sandro, and Daniel Meyer. "On the robustness of clone detection to code obfuscation." Software Clones (IWSC), *2013 7th International Workshop on. IEEE*, 2013.

[9] Brenda S. Baker. A Program for Identifying Duplicated Code. In Proceedings of Computing Science and Statistics: *24th Symposium on the Interface*, Vol. 24:4957, March 1992.

[10] Bruntink, Magiel. "Aspect mining using clone class metrics." 1st Workshop on Aspect Reverse Engineering. 2004.

[11] Agrawal, Akshat, and Sumit Kumar Yadav. "Technique for Searching of Similar Code Segments." (2013).

[12] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwe. An evaluation of clone detection techniques for identifying cross-cutting concerns. In Proc. *International Conference on Software Maintenance*, 2004.

[13] Roy, Chanchal Kumar, and James R. Cordy. A survey on software clone detection research. Technical Report 541, Queen's University at Kingston, 2007.

[14] Yuan, Yang, and Yao Guo. "CMCD: Count matrix based code clone detection." *Software Engineering Conference (APSEC), 2011 18th Asia Pacific. IEEE,* 2011.

[15] http://www.semdesigns.com/Products/Clone/

[16] https://svn.ipd.kit.edu/trac/jplag/

[17] http://homepage.usask.ca/~mdu535/tools.html

Hamed Jelodar is received the B.Sc. degrees in computer software engineering from Islamic Azad University, Iran in 2012. He is a Master's student in Computer engineering and his research interests include Wireless Networks, Software Quality, Clone Detection, and Web Mining. Also, He has presented 5 papers in International Journal, 1 paper in International Conference and 4 papers in National Conference.

Javad Aramideh received his Master in Computer Engineering from Islamic Azad University, Sari, Iran in 2014 and his research interests include Wireless sensor Networks, ad-hoc network, Software Testing, Clone Detection, and Web Mining Also, He has presented 3 papers in International Journal, 1 paper in International Conference and 7 papers in National Conference.