ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

A Review on Code Clone Detection and Implementation

Shashank Prabhakar[1], Sonam Gupta[2]

Student, Department of CSE, Ajay Kumar Garg Engineering College, Ghaziabad, India

Assistant Professor, Department of CSE, Ajay Kumar Garg Engineering College, Ghaziabad, India

ABSTRACT: Code cloning has always been an area of research for quite a time now. The copying of the code and pasting it over a set of code-with or without the modification is known as code cloning. Code detection technique are different for different type of code clones in order to find a code fragment. In many large-scale software industries; the developers are mostly perturbed by the code duplication in their source code.Such that, the clone duplication or clone detection technique reassured the identification of the code fragments in the source code. The prime concern of the code detection is to identify the clone and replace it with a single function or with some code fragments that will eventually result in the same working behavior of the removed code clone. As a result of that, in the last decade, the topic of detecting code duplication led to various tools that ease the work to detect the duplicated blocks of code. In brief, the text based technique can detect only the Type 1 clone. The Token based technique detects both the Type1 and Type II clone. In this paper, the different methods for code clone detection, different tools and technique are covered and further, the analysis of code cloning is also discussed.

KEYWORDS: Code Analysis, Code Clones, Code Clone Detection

I. INTRODUCTION

Research showed that a large-scale applications often contain code clones. The existence of code clones or unnecessary duplication of code fragments create much instability and ambiguity. These instabilities leads to the routine maintenance tasks, which is often complex and takes a lot of time to detect even a single fragment of code clone. In addition, the code clones can induce the root of bugs and errors and preventing the changes of being propagated.Thus, in order to prevent and to avoid these instabilities, we have to figure out the potential root clones. [1]

There are various techniques available for code clone detection, they are: string based, token based, tree based, semantic based [1,2,3]. Out of these token based and abstract syntax tree based approaches are more popular. Since, every technique has its own advantages and disadvantages.

The approach here is to find the sub-clones and sequence clone for the detection of the clones [4].And it has been described in two steps; firstly Abstract syntax tree (AST) generation and then suffix tree generation for the clone detection. And, AST is basically used to detect all the statement clones. Further, the block level, the clone detection is achieved by making a suffix tree, after which the tokens are being generated according to the developed suffix tree. This method does not require any threshold value by the user. Andthis is the illustration of every form of block level, such as, Type-1 and Type-2 [5, 6] clones. Though, the viewpoint used in this paper is solely independent. And the tool that is developed is based on Java.

II. RELATED WORKS

To begin this, let's get familiar with different clone detection terminology.

A. Code Fragment

A code fragment (CF) is the sequence of code of lines (with or without comments). It can be of type, e.g. Function definition, begin-end block, or sequence of statements. The CFs are identified by its file name and begin-end are identified by the numbers which are in the original code and is denoted in a triple: (CF.FileName, CF.BeginLine, CF.EndLine) [7, 8].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

B. Code Clone

To define code clone, let's consider;A code fragment CF2 is a clone of another code fragment, CF1. The similarity can be defined as the similarity function, i.e.f(CF1) = f(CF2), where f is the similarity function. Two fragments are similar to each other if they usually form a clone pair (CF1; CF2), and when many of the fragments of code are similar, they form a clone class or a clone group [7, 9].

C. Clone Types

The clone types have basically two types of similarity between them - theFragments which can be based on the similarity of the program text, or they can be based on their functionality. The first kind is the code fragment that has been copied and pasted on some other location. The various types of clones which are based on both the textual (Types 1 to 3) [3] and functional (Type 4) similarities have been described:

Type-1: Identical code fragments; except a little variations in the whitespaces, layouts and comments.

Type-2: Syntactically identical fragments; except the variations in the identifiers, literals, whitespaces, layouts and comments.

Type-3: Copied fragments with a variation in code such as changed, added or removed statements. Also the addition of variations in the identifiers, literals, types, whitespace, layout and comments.

Type-4: Two or more code fragments that performs the same computation butare only implemented by some different syntactic variants that are available [8,9].

III. TECHNIQUES TO DETECT CLONE

There are four types of code clone detection techniques:

1) Textual approach: Textual approaches (or text-based technique) has no transformation on the source code before the comparison is done on the both side.And in most cases; the original source code is used as it is in the clone detection process i.e. with or without modification. Examples: SDD, NICAD, Simian1, etc. [10]

2) Lexical approach: Lexical approaches (or token-based technique) alwaysstarsby converting the source code into a sequence of lexical, which are called as "tokens", by using compiler-style lexical analysis. The sequence then scans for unnecessary duplication of the subsequence of the token with respect to the original code which are returned back as clones[17]. Lexical approaches are generally more resilient over small changes in the code. They can be defined as formatting, spacing, and renaming that differs than the textual techniques. Examples: Dup, CCFinder, CPMiner etc. [11, 12]

3) Syntactic approaches: Syntactic approaches uses a parser to convert a source programs into the parse trees or abstract syntax trees, which can be processed using tree matching or structural metrics to find the clones. Examples: CloneDr, Deckard, CloneDigger etc.

4) Semantic approaches: Semantic approach has been proposed using the static program; as it provides the in-depth information than the syntactic similarity. In some approach, the given program is represented in the form of program dependency graph (PDG) or in Expressions and statements; while all the edges represents- control and data dependencies. Examples: Duplix, GPLAG [13].

IV. CLONE DETECTION PROCESS

A clone detector should try to find the abstract of code with high similarity in the source text. But the problem is that we don't not know which code fragments will be repeated. Therefore, the detector should compare every code fragments with other code fragments [14]. In order to do that, this comparison of code fragments with one another is an expensive and time consuming task from a computational point of view, therefore, several steps are taken to avoid these[17]. Even after identifying the potentially the cloned fragments, various tools and analysis may be required to identify the real clones. In this section, we have provided an overall summary of the basic steps in the clone detection process [15,16].

1) Textual Similarity: In the textual based similarity,code fragment are exactly similar to each other. Now, based on the program text; there are three types of clones:

Type I Clones: In Type I clone, two code fragments are completely identical to each other. However, there can be a little variations in the; whitespace, layout and comments. Let's consider the following example:

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

```
classCheckEvenOdd
{
public static void main(String args[])
{
intnum;
System.out.println("Enter an Integer number:");
//The input provided by user is stored in num
Scanner input = new Scanner(System.in);
num = input.nextInt();
/* If number is divisible by 2 then it's an even number
else odd number*/
if ( num % 2 == 0 )
System.out.println("Entered number is even");
else
System.out.println("Entered number is odd");
}
}
```

(Original Code)

```
classCheckEvenOdd
{
public static void main(String args[])
{
intnum;
System.out.println("Enter an Integer number:");

//The input provided by user is stored in num
Scanner input = new Scanner(System.in);
num = input.nextInt();

/* If number is divisible by 2 then it's an even number else odd number*/
if ( num % 2 == 0 )
System.out.println("Entered number is even");
else
System.out.println("Entered number is odd");
}
}
```

(Code Clone)

Two code fragmentscopied the code fragment and the original code and the fragment code are same. When we remove the white space and comments from the code, it doesn't change the overall functioning of the program.

Type II Clones: In Type II clone, the code segment which is taken or copied from the original code is exactly the same as the original. However, there can be any possible variations in the literals, variables, constants, class, types, layouts and comments, whatever the programmer wishes. But, the syntactic structure of both the code segments are nearly the same. Let us consider the following example:

```
classOddOrEven
{
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

```
public static void main(String args[])
{
int x;
System.out.println("Enter an integer to check if it is odd or even ");
Scanner in = new Scanner(System.in);
x = in.nextInt();

if ( x % 2 == 0 )
System.out.println("You entered an even number.");
else
System.out.println("You entered an odd number.");
}
}
```

(Original Code)

```
classEvenOdd
{
public static void main(String args[])
{
int c;
System.out.println("Input an integer");
Scanner in = new Scanner(System.in);
c = in.nextInt();

if ( (c/2)*2 == c )
System.out.println("Even");
else
System.out.println("Odd");
}
}
```

(Code Clone)

We saw that the two code segments are syntactically similar to each other, but there are a lot of variations in the values, variable names and value assignments

V. CONCLUSION

Our Main focus is on Type-1 code clones only. Therefore, the paper's primary discussion was only for the Type-1 and Type-2 code clones and their detection, accordingly.

REFERENCES

1. H. A. Basit, S. J. Puglisi, W. F. Smyth, A. Turpin, and S. Jarzabek. Efficient token based clone detection with flexible tokenization. In The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers, pages 513–516, New York, NY, USA, 2007. ACM.
2. I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In Proceedings of the International Conference on Software Maintenance (ICSM), pages 368–379,Washington, DC, USA, 1998. IEEE Computer Society.

3.Ms.Sonam Gupta and Dr. P.C Gupta, "Clones: A Survey", pages 152-156, IJCST Vol. 3, ISSue 3, July – Sept 2012.

4. Andrea De Lucia, Rita Francese, Giuseppe Scanniello, GenoveffaTortora, "Understanding Cloned Patterns in Web Applications," Proceedings of the 13th International Workshop on Program Comprehension (IWPC'05), IEEE.

5. KanikaRaheja, RajkumarTekchandani, "An Emerging Approach towards Code Clone Detection: Metric Based Approach on Byte Code," IJARCSSE, Vol.3, May 2013.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

6. S. Bellon, R. Koschke, G. Antoniol, J. Krinke and E. Merlo, Comparison and Evaluation of Clone Detection Tools, Transactions on Software Engineering, 33(9):577591 (2007).
7. PrajilaPrem, "A Review on Code Clone Analysis and Code Clone Detection," IJEIT, Vol.2, Issue 12, June 2013.
8. Chanchal Kumar Roy and James R Cordy, "A Survey on Software Clone Detection Research", Computer and Information Science, Vol. 115, No. 541, pp. 115, 2007
9. Robert Tairas, "Clone detection and refactoring", Proceeding of OOPSLA '06 Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, pp. 780-781, New York, USA, 2006
10. Chanchal K. Roy, James R. Cordya and Rainer Koschkeb, "Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach", Journal Science of Computer Programming, Vol. 74, No.7, pp. 470-495, May 2009
11. Mohammed Abdul Bari, Dr.ShahanawajAhamad, "Code Cloning: The Analysis, Detection and Removal", International Journal of Computer Applications (0975 – 8887) Vol. 20, No.7, April 2011.
12.Katsuro Inoue, "Code Clone Analysis and Its Application", Software Engineering Lab, Osaka University.
13.Rainer Koschke, "Survey of Research on Software Clones", Dagstuhl Seminar Proceedings.
14. Mohammed Abdul Bari, Dr.ShahanawajAhamad, "Code Cloning: The Analysis, Detection and Removal", International Journal of Computer Applications (0975 –8887) Vol. 20, No.7, April 2011.
15.D. Gayathri Devi, Dr.M.Punithavalli, Comparison and evaluation on matrices based approach for detecting code clone.
Indian Journal of Computer Science and Engineering (IJCSE)
16. J. Krinke, "Identifying Similar Code with Program Dependence Graphs", in Proceedings of the 8th Working Conference on Reverse Engineering, (2001).
17. Gupta, Sonam, and P. C Gupta. "Literature survey of clone detection techniques." International Journal of Computer Applications 99.3 (2014): 41-44.