A Comprehensive Review of Code Clone Detection Techniques

Kuldeep Kaur[1], Dr. Raman Maini[2]

Department of Computer Engineering, Punjabi University, Patiala-147002(India)

***Abstract*-code cloning is a current area of research in software systems .To copying the existing code and paste it with or without modification is known as code cloning. Code clone detection techniques concerned with finding of the code fragment that produce the same result. The issue of finding the duplicated code led to different tools that can detect the copied code fragments. In this paper comparative analysis of various code clone detection technique have been done. It has been observed that text based technique can detect only Type 1 clone. Token based technique detect Type1, Type II clone. Tree based approach detect Type1, Type II, Type III clone but it is very difficult to create a syntax tree its complexity is very high.**

***Keywords:*-code clones, clone detection, comparison, precision, recall.**

I. INTRODUCTION

Code clone is a current area of research. To copy the code and reused the code by doing some modifications or without doing some modification in the exiting code are common activities in software development. The pasted code are called clone of the original and the process is called software cloning. In the software system copied code fragments and code clones are considered as bad smell of the software. It is observed that code clone has bad effect on the maintenance of the software system. To remove the clones from the software systems is quite beneficial. These clones are syntactically or semantically similar. It is very difficult to identify which code is copied code or which code is original. Several studies show that it is difficult to maintain software system which contains the code clones as compared to others which does not contain the clone. Cloning may increases the bug probability if some bug is found in the source code and that code is reused by copying and pasting then that bug is also found in that pasted code fragment. for fixing the bug all these code fragment should be detected[1]. Code clones are basically of four types , where the first three Type I, Type II, Type III are textual and last one Type IV is functional.

A. Reasons of Code Duplication:-

There are various reasons for code duplication. Reuse of code, logic and design is the main reason of code duplication. Sometimes there is a need to merge two similar system having similar fuctionalities to develop a new one which result duplication of code even both the system are developed by different teams. There is frequent update of the software Developers are asked to reuse the existing code because of high risk in developing the new code. One of the major cause of code duplication is the time limit assigned to developers. To complete a project some time limit are assigned to developers. Developers find the easy solutions of the problem due to time limit. They find the similar code related to their project .they just copy and paste the existing code.

B. Drawback of Code Duplication:-

Code clones have bad impact on the maintainability, reusability and quality of the software. If there is any code segment present in the software which having a bug and the code segment is copied and pasted anywhere in the system then the bug is remains in all the pasted code segment which is difficult to maintain. When duplicated code used in the system it may lead to bad design which increase the cost of the system. If in the software system there is duplicated code, to understand the system additional time needed. It becomes difficult to upgrade the system or even to change the existing one.

II. CLONE TERMINOLOGIES

All clones are identify in the form of Clone Classes and Clone Pairs. clone classes and clone pairs tells about the similarity between various code clone fragments. If they have some similar sequences in the code, clone-relation exists between the code fragments. For example character strings, strings without white space, transformed token sequences and sequences of token type so on.

A. Code Fragment: Code fragment is some sequence of code lines having different types of similarity between various code fragments in its source code. These similar code fragments may have comments or without comments. For example: sequence of statements, begin-end block, etc.

B. Clone Pair: If there is any clone relation exist in the pair of code fragments then it is called a clone pair or clone pair is a pair of code fragment having some similarity between them.

C. Clone Set: A set of all the identical or similar fragments.

D. Clone Class: A set of all the clone pairs in which the existing clone pairs having some clone relationship between them is known as clone class.

E. Code Clone Types: On the basis of functionalities and program text, two code fragments are said to be similar. The first type of clone are mainly the result of copy and paste activities. In the following type of clones Type I , Type II and Type III clones are based on the textual similarity and Type IV clones are based on the functional similarity.

1) Textual Similarity: in the textual based similarity or program based similarity the code fragment are similar to each other based on the program text. Based on the program text there are three types of clones.

Type I Clones: In Type I clone, two code fragments are identical to each other. However, there might be some variations in white space, comments and layouts. Let us consider the following example.

```
Int  n;                                          Int n;
Cout <<"enter the number";                       Cout<<"ENTERTHE NUMBER";
Cin >> n;                                        Cin>>n;
If (n%2==0) { //comment1                         If (n%2==0){//comment1'
Cout <<"the number is even";}                    Cout<<"THE NUMBER IS EVEN";}
Else{                                            Else{
Cout <<"the number is odd";                      Cout<<"THE NUMBER IS ODD";
}                                                }

 (original fragment)                              (copy clone)
```

Two code fragments copied fragment and original code fragment are same, when we remove the white space and comments from the code.

Type II Clones: In Type II clone the code segment which is copied from the original code is same as original code . however their may be some possible variations in the literals , variables, constants, class, types, layout and comments. The syntactic structures of both the code segments are same. Let us consider the following example.

```
If (salary >= 5000) {                            if  (sal >= 5000)
Bonus = 0.5*salary; // comment1                   {//comment1'
}                                                 bon=0.5*sal;
else                                                }
Bonus = 0.2*salary; // comment2                   else
                                                 bon=0.2*sal;
 (original fragment)                            (copy clone)
```

We see that the two code segments are syntactic similar to each other but there are lot of variations in the shape, variable names and value assignments.

Type III Clones: In Type III clones, By adding or changing some statements the copied code fragment is further modified. Let us consider the following example:

```
Total= phy+chem.+math;                    Total= phy+chem.+math;
Per = total/3;                            Per = total/3;
If (per>= 70) {                           if (per>= 70){
Cout<<"first division"; // comment1       //comment1'
    ;}                                    cout<<"first division";
else                                      cout<"remarks excellent"; // this statement is added
cout<<"second division"; // comment2'     ;}
                                          else
                                          cout<<"second division"; //comment2
   (original fragment)                      (copy clone)
```

We see that in the copied fragments one statement is added .

2) Functional Similarity: If there is any two code fragments which are similar in their functionality they are known as semantic clones . Type IV clones are semantic clones.

Type IV Clones: In this code fragments are semantic similar. In such type of clones, it is not necessary that the code fragment are copied from the original code. Two code fragments may be developed by the different teams but they perform same computation. code fragments are similar in their functionality because different teams Implement the same logic. Let us consider the following code fragment 1 and code fragment 2 where the swapping of two variables done .

```
Fragment 1:                    fragment2:
int  a=5, b=10 , temp ;        int a=5, b= 10;
temp = a;                      a =  a + b;
a  = b;                        b =  a-  b;
b  = temp;                     a  =  a- b;
```

Both the code fragments are similar from the semantic point of view. In fragment 1 the swapping is done using three variables and in code fragment 2 swapping is done using two variables.

Precision and Recall: clone detector identified the characteristics of the candidate key which are discussed in terms of recall and precision. If the code clones are exactly identified then the value of precision is high and if the value of precision is low it indicates that code clones are not actual code clones. Recall tells that actual clones which are present in the source code are found or not .if the value of recall is high then in the source code most of the clones have been found and if the value of recall is low then in the source code most of the clones is not found[2].

III. CLONE DETECTION TOOLS AND TECHNIQUES

In the literature several types of clone detection tools techniques are presented . for the research purposes most of

the techniques are used, while a few of them are also used for commercial pupose.

A. Text-based Techniques: In the text based technique the source code fragment are assumed as sequence of line. After removing the various comments, whitespace by applying the various transformations the code fragment are compared with each other. Once the two code fragment are found to similar to each other to some extent they are known as clone pair or clone pairs form the clone class. Sometimes in the clone detection process the source code is directly used. Text based technique is efficient technique but it can detect only Type I clones. Text based approach can not detect the structural type of clone having the same logic but different coding. In the text based approach following transformations are applied on source code.

1. Comments Removal: In the code fragment ignore all the comments.
2. White space Removal: In the code fragment removes all the tabs and blank spaces.
3. Normalization: On the source code some normalization are applied.

Though text based approach can detect only type 1 clone . This technique cannot detect the structural type of clones having same logic but different coding [1].

Tool: Baker's Dup represent the source code as sequence of lines and detects the clones in the code fragment line-by-line. Baker's uses a line based string matching algorithm or lexer on the individual lines. First Dup tool removes comments and white space from the source code and then it replaces various identifiers , variables and types with a special parameter so that if the name of the two variable is different clone can be identified. [1]. Baker ' Dup tool can not detect the clones if the source code is written in different style.

B. Token-based Techniques: In the token-based technique, first sequence of tokens is generated from the source code. For converting the source code into tokens it requires a lexer. Lexer convert the source code into tokens then the various transformation are performed by adding, changing or deleting some tokens. For finding the duplicated code or duplicated subsequence of token the sequence is scanned. and the code portions representing the duplicated code returned as clones. Token based technique can detect Type I, Type II clone .

Tool: CCFinderx is one of the tool of the token-based techniques. CCFinderx find the clones both with in the files or from various files from programs and find the location of the clones in the program. First, tokens are generated from the source code and then the single token sequence are formed by concatenating all the tokens. various transformations are applied on the token sequences based on the transformation rules . After applying various transformations various identifiers are replaced with a special token. To find the clones from the token sequence some tree based sub-string matching algorithm is used and the similar sub strings pairs are called as clone pairs/clone classes [1].

C. Tree-based Techniques: In the tree-based approach from the source code a parse tree or an abstract syntax tree is obtained. This technique creates sub trees rather than creating tokens from each statements. The code then said to be code clone if the sub trees match. With the help of parser of a language similar sub trees are searched in the tree using tree matching algorithm or structural metrics then the code of similar sub trees are returned as clone pairs. Abstract syntax tree have the complete information about the code. The result obtained from this technique is quite efficient but to create a abstract syntax tree is difficult for a large software and the scalability is also not good.

Tool: CloneDR is one of the tool of the abstract syntax tree based clone techniques. Compiler is used to generate AST or abstract syntax tree and the complier compares the sub trees based on some hash function, the sub trees which are similar are returned as clones.

D. PDG-based Techniques: Program Dependency Graph (PDG) technique is more efficient then tree based technique. Program dependency graph show data flow and control flow information. First the program dependency graph is obtained from the source code then to find the similar sub graphs or clones several type of sub graph matching algorithm are applied and returned as clones. This technique can detect both semantic and syntactic clones but in case of large software to obtain the program dependency graph is very difficult.

Tool: One of the important program dependency graph based clone detection approach is that of Komondoor and Horwitz's PDG-DUP which identify isomorphic program dependency sub graphs using program slicing.

E. Metrics-based Techniques: In Metrics based Technique first different types of metrics of the code like number of lines and number of functions are calculated and compare these metrics to find the clones. Metrics based technique does not compare code directly. To find the code clones several type of software metrics are used by clone detection techniques. Most of the time, for calculating the various type of metrics the source code is converted into abstract syntax tree or program data graph. Metrics are calculated from the name , layout, control flow and expression of the functions.

Tool: One of the important tool of metrics-based techniques is **Covet/CLA** to detect the clones using metrics. Mayrand et al. calculate various type of metrics for each function unit of a program like number of CFG edges, lines of source code, number of function calls etc. Code fragments which have similar metrics values are known as code clones. **Covet/CLA** does not detect the Partly similar codes.

Comparison of the clone detection techniques w.r.t different properties:

Properties	Text based	Token based	Tree based	PDG based	Metrics based
Transformation	Removes whitespace and comments	Tokens is generated from the source code	AST is generated from the source code	PDG is generated from the source code	To find metrics values AST is generate from the source code
Representation	normalized source code	In the form of tokens	Represent in the form of abstract syntax tree	Set of program dependency graph	Set of metrics values
Comparison based	tokens of line	Token	Node of tree	Node of program dependency graph	Metrics value
Computational complexity	Depends on algorithm	Linear	Quadratic	Quadratic	Linear
Refactoring opportunities	Good for exact matches	Some Post processing needed	It is good for refactoring because Find syntactic clones	Good for refactoring	Manual inspection is required
Language in dependency	Easily adaptable	It needs a lexer but there is no syntactic knowledge required	Parser is required	syntactic knowledge of edge and PDG is required	Parser is required

In Text based approach only type 1 clone is detected. It cannot detect the clones which having structural similarity but having different coding or have same logic. Token based approach is more efficient then text based approach it can detect type 1 clone as well as type II clone. In tree based approach a parsed tree is generated from the source program. The result which is obtained from the tree based approach is efficient but to create a syntax tree is very difficult and scalability of this is also not good. Program dependency approach contains the data flow and control flow information of a program. The semantic information of a program also contain in the program dependency graph. It detect type 1, type II and type III clones.

IV. CONCLUSION

Code clone is a big problem. A copy and paste activity which is done by programmer is the main reason of code cloning. It looks like a simple and effective method, these copy and paste activities are not documented. Which create a bad effect on the software quality and duplication also increase the bug probability and maintenance problem. In this paper, a comprehensive review of various techniques has been done by emphasis on the types of clones. It has been observed that no technique is found good on the basis of precision, recall, robustness and scalability. Text based approach is efficient technique. It gives overview of the duplicated code. Text based approach can detect only Type I clone it cannot detect the clones having the same logic or having structural similarity. Token based technique is more efficient then text based approach technique. Token based technique detect Type I as well as Type II clone it cannot detect Type III clone. For detecting Type III clone abstract syntax tree approach is used . The result obtained from this quite efficient but it is difficult to crate syntax tree . for detecting Type IV clone program dependency graph approach is used.

REFERENCES

[1]. Chanchal Kumar Roy and James R. Cordy, "*A Survey on Software Clone Detection Research*", Technical Report No. 2007-541, School of Computing Queen's University at Kingston Ontario, Canada, September 26, 2007.

[2]. Prajila Prem," *A Review on Code Clone Analysis and Code Clone Detection*." International Journal of Engineering and Innovative Technology (2277-3754) Volume 2, No.12, June 2013.

[3]. Harpreet Kaur and Rupinder Kaur," *Clone Detection in Web Application Using Clone Metrics*." International Journal of Advanced Research in Computer Science and Software Engineering (2277-128x) volume 4, No.7, july 2014.

[4]. Kanika Raheja and Rajkumar Tekchandani,"An *Emerging Approach towards Code Clone Detection :Metric Based Approach on Byte Code*". International Journal of Advanced Research in Computer Science and Software Engineering (2277-128x) volume 3, No.5,May 2013.

[5]. Manisha Gaholt and Deepak Sethi,"*Comparative Analysis of Tree-based and Text- based Technique for Code Clone Detection."* International Journal of Advanced Research in Computer Engineering and Technology(2320-6802) volume 2,No.2,Fb 2014.

[6]. G. Anil kumar, Dr. C.R.K.Reddy, Dr. A. Govardhan, A. Ratna Raju4,*"Code duplication in Software Systems: A Survey*." International Journal of Software Engineering Research & Practices Vol.2, Issue 1, Jan, 2012.

[7]. Sonam Gupta and P. C Gupta,"*Literature Survey of Clone Detection Techniques.*" International Journal of Computer Applications (0975 – 8887) Volume 99– No.3, August 2014.

[8]. Dhavleesh rattan, Rajesh Bhatia, Maninder Singh, " *Software Clone Detection: Systematic Review,"*Information And Software Technology, ELSERVIER, PP 1165-1199,2013.

[9]. Mrs. Kavitha Esther Rajakumari , Dr. T. Jebarajan ," *A Novel Approach to Effective Detction and Analysis of Code Clones* ," IEEE, 2013.

[10]. S. Bellon, R. Koschke, G. Antoniol, J. Krinke and E.Merlo, "*Comparison and Evaluation of Clone Detection Tools*", Transactions on software Engineering , 33(9):577-591(2007).

[11]. Koschke, R., "*Survey of Research on Software Clones* " In Dagstuhl Seminar 06301, pp.24,2006.

[12]. C. Kapser, M.W Godfrey,"*Suppprting the analysis of Clones in Software Systems : research articles*", Journal of Software Maintenance and Evolution 18 (2) (2006) 61-82.