Volume 5, Issue 5, MAY 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Review on Various Code Clone Detection Techniques

Manpreet Kaur, Madan Lal
Dept. of Computer Engineering
Punjabi University
Patiala, India

Abstract— Software programmers copy and paste a piece of code to the software with or without any alteration for their convenience, but it cause some maintenance problems. This copied and pasted code is known as clone. The code clone is the main factor which degrades the design and structure of the software because it lowers the quality of the software such as readability, changeability and maintainability. The process to indentify code clone from the software is known as code clone detection. Many code clone detection techniques are available. The various types of clone and detection techniques are describe in this paper.

Index Terms— Software, Code clone, Code clone detection techniques.

I. INTRODUCTION

The software life cycle constitutes of three steps first we have to clearly define the requirement, implement these requirement and then we have to maintain the software and evolve it according to user's requirement. The maintenance phase is very costly and important among all the phase of software life cycle. Code clone is bad smell for maintenance point of view [3][6].

The software engineer, copy a piece of code from one part of software and paste it to other part with or without any changing or alteration to save time and efforts.[1] This process of copy and paste of code is known as code cloning. The code which is copied and pasted is known as code clone. The code clone make maintenance of the software more difficult because bug or error occurs in one module will be occurred in all the other copies of that module [2][6]. There are no records of how many copies of a module present in software, this make hard to fix such bug and maintain.

Code clone is like a duplicate code which is copied from one part of software and pasted to another part of the software with and without any changes or alteration. The reason of code clone can be intentional or unintentional [3]. In code clone there are two types of similarities [2].

1. If the program text matches also known as syntactic similarity
2. If the behavior among them is matches also known as semantic similarity.

The syntax similarity is easily detected as compare to semantic similarity.

Code clone need high maintenance because if change in one clone is done then that change have to be done in all the respective clones. By the previous studies have shown that 7% to 23% of code clone present in source code in the software [4][8].

The process of indentify the code clone in the software is known as code detection. Code detection can be done manually. In manual code detection code clone is indentify by comparing lines one by one, but for the large software or program it is very time consuming and tedious[3]. Various tools are proposed to detect code clone in the software, all tools follow their own algorithm. The code clone detection tools must provide precise and crucial information about clone.

II. TYPE OF CODE CLONE

A. *Type 1*

These types of clone are identical clone that is these types of clone are exactly same to the code from it is copied that is original code. These code clones has same structure as original code. Only comment and white space modification is allowed[2]. These types of clone are very easy to detect. Type 1 clone has syntactic similarity to the original code.

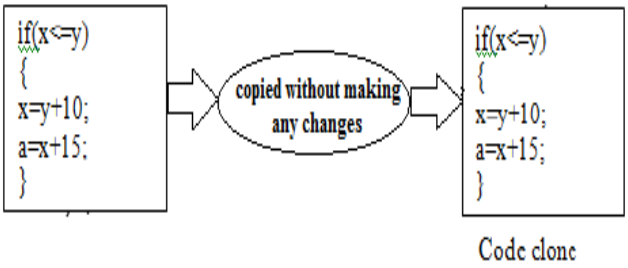


Fig 1: Type 1 clone [4]

B. Type 2

These are also known as rename clone [2][3]. These clones also have same structure as original code and have syntactic similarity to the original code. In these clones change in literal and identifier is allowed. These clones are also easy to detect by code clone detection tools because only literal and identifier is changed.

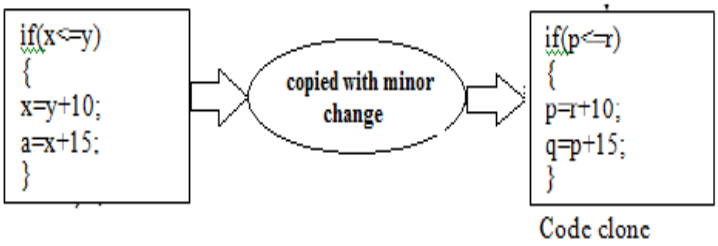


Fig 2: Type 2 clone [4]

C. Type 3

These type of clone copies the code from original code and modified by adding, deleting and changing statement [2]. These are also known as modified clone. In these clones there may be two or more module or original code are concatenate and modified. These clones are harder to detect as compare to type 1 and type 2.

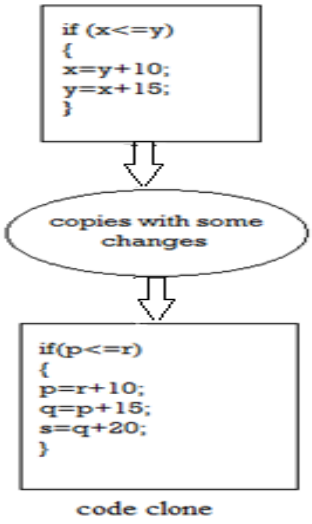


Fig 3: Type 3 clone

D. Type 4

These clones have similar functionality and logic but have different syntax. These clone also known as semantic clone [2]. In these clone only functionality and logic of the code is same i.e. type 4 has semantic similarity to the original code. These clones are most difficult to detect because type 4 clones does not have similar syntax, it only have similar semantic. Example: In these clones, for loop is modified to while loop or while loop is modified to for loop. The logic and functionality is same (i.e. have similar semantics) but syntax is modified.

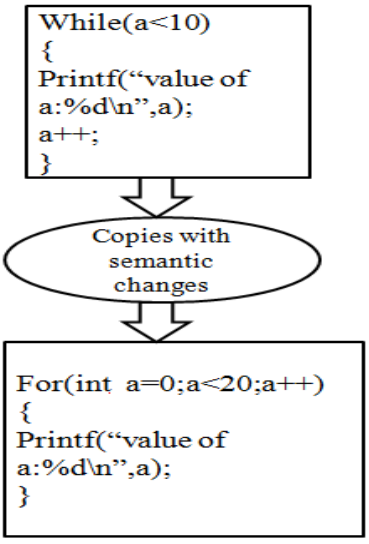


Fig 4: Type 3 clone

III. DIFFERENT TECHNIQUES TO DETECT CLONE

A. Text based technique

This technique is most basic technique. This technique detect code clone by comparing source code line by line in the form of string [4]. In this technique, there is no need to transform the source code to any other form to detect clone. This technique is used to detect only type 1 code clone [2][3]. This technique is not used to detect clone having same structure and logic but different coding.

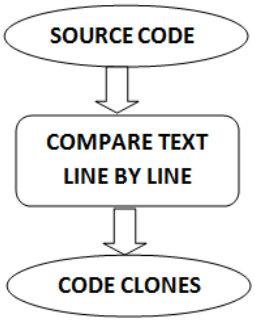


Fig 5: Text based technique

B. Token based technique

This technique detects the code clone by comparing source code line by line in the form of token [4]. In this technique the source code have to convert into tokens with the help of parser or lexer [3]. This technique is better than the text based technique when the comments and spaces are present in the source code but it has low efficiency because while conversion source code to token sequence various false positive may introduction in the code. This technique is used to detect only type 1 and type 2 code clone [2].

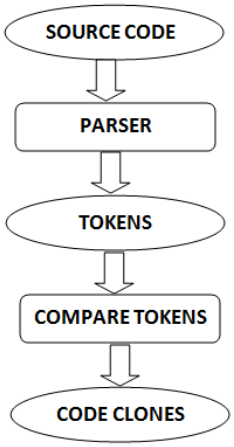


Fig 6: Token based technique

C. Abstract syntax tree(AST) based technique

In this technique the source code is first converted into abstract syntax tree, then the sub tree of the abstract syntax tree is compared to detect code clone [3][4]. This technique detect all syntactic clone i.e. type 3 clone [2]. It is very efficient technique but is very complex to create abstract syntax tree [3].

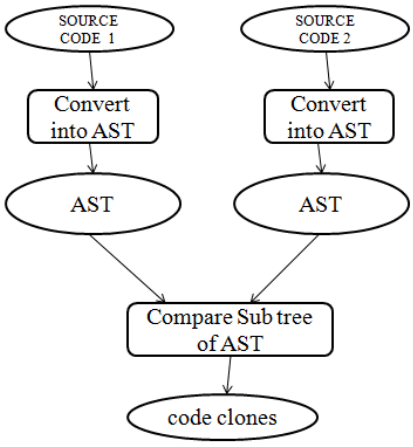
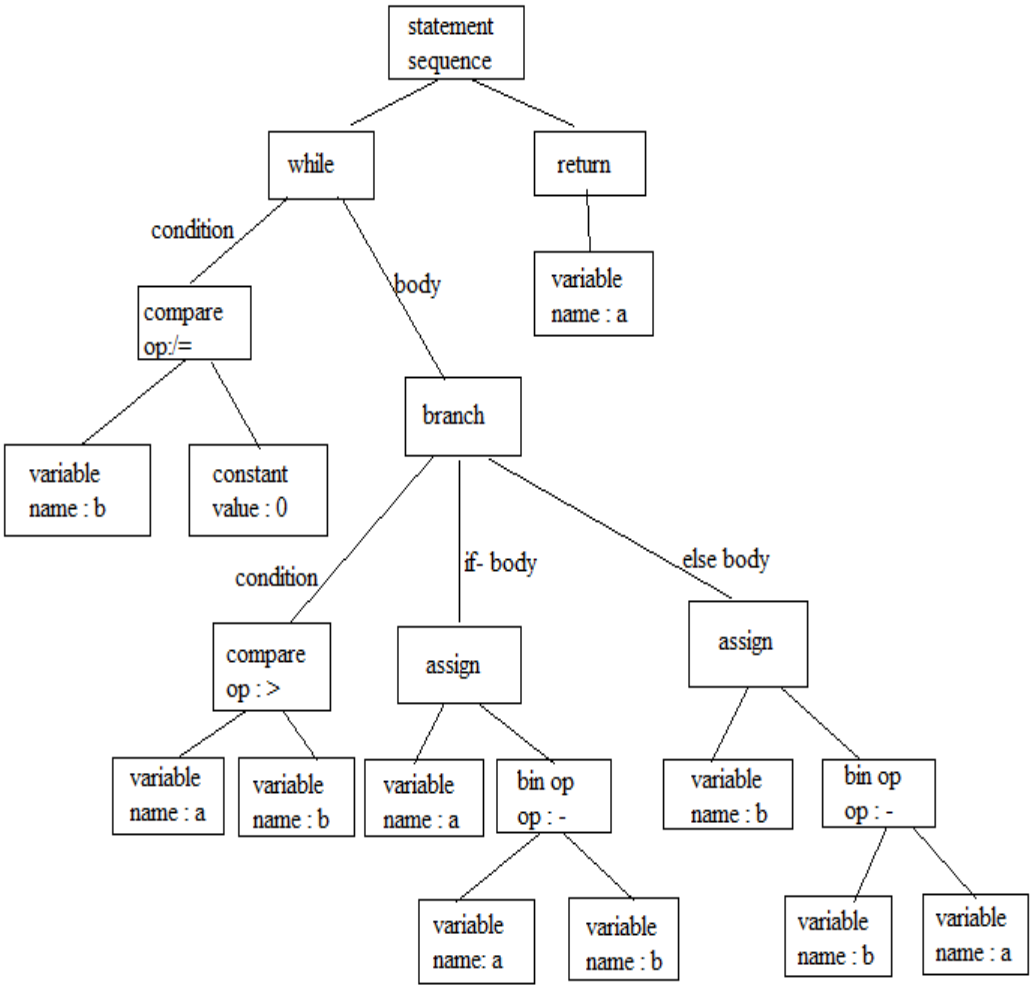


Fig 7: Abstract syntax tree technique


```
while b!=0
If a>b
a:=a-b;
else
b:=b-a;
return a;
```

The abstract syntax tree of above code is shown in figure below.



. Fig 8: Abstract syntax tree[9]

D. Program dependency Graph (PDG) technique:

In this technique the source code must converted into program dependency graph then its sub graph is compared to detect the code clone. This technique is very efficient because it can detect syntactic and semantic clone in the software or program. PDG has nodes which represent program element and edges which represent dependencies. There is two type of dependencies one is control dependency other is data dependency [6][3].PDG has two disadvantages.[6]

1. PDG technique is deficient to detect contiguous code clone is difficult than other clone detecting techniques.
2. PDG is time consuming than other detecting techniques.

```
 1: int fibonacci(int n){
 2:   int value = -1;
 3:   if (n <= 0) {
 4:     System.out.println(
                    "Illegal parameter");
 5:   } else if (n == 1 || n == 2) {
 6:     value = 1;
 7:   } else {
 8:     value = fibonacci(n - 2) +
                    fibonacci(n - 1);
 9:   }
10:   return value;
11: }
```

Fig 9: Source code[6]

PDG of above source code is shown below. In this label attached to the notes represent the line where their element locates in the source code. [6]

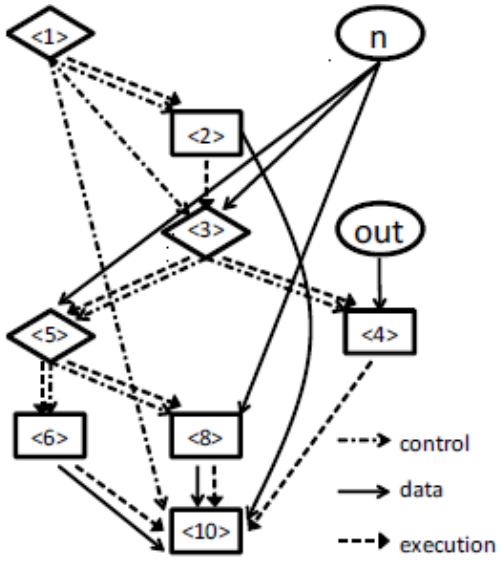


Fig 10: PDG[6]

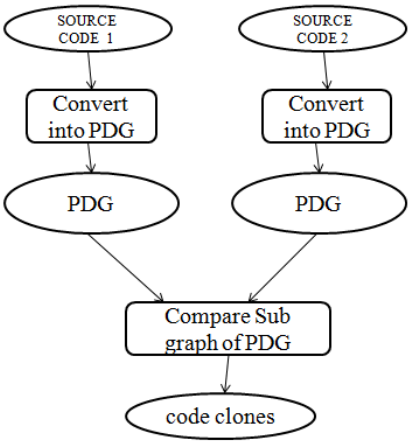


Fig 11: Program dependency graph technique

E. Metric based technique:

This technique is not applied directly to the source code, metric are calculated from the source code then the measured metric are compared to detect code clones [3]

The metric can be calculated on two levels [4]

1. File Level Metrics
   - i. Line : This metric includes total number of physical line while ignoring blank line, this also known as line of code metric
   - ii. Branches: This metric includes total number of branches like if, while, for etc used in the program
   - iii. Calls: This metric include total number of calls to other method or function present in program.
   - iv. Classes: This metric include the total number of classes present in the program.
   - v. Maximum complexity: This metric include the maximum complexity value of the most complex method present in the program.

2. Method Level Metrics
   - i. Complexity: This metric include complexity of the method which is equal to one plus the total number of branch statement present in the method.
   - ii. Statement: this metric include the total number of statement present in the method.
   - iii. Calls: this metric include the total number of calls to other methods or function occur in each methods. This is also known as fan out.

This technique can detect syntactic and semantic clone in the software or program but this technique is not so efficient i.e. not detect all possible clone present in the programs or software [3].

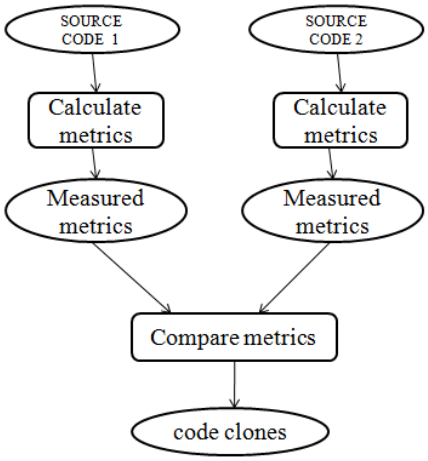


Fig 12: Metric based technique

F. Hybrid technique:

This technique is the combination of any two or more techniques describe above, example if token based technique and metric based techniques are combined it form hybrid techniques.

IV. CONCLUSION

The aim of the code clone detection technique is to detect all possible clone present in the program or software but all the code detection techniques have some limitations like text based and token based techniques can't detect semantic similarities in the software, abstract syntax tree and program dependency graph techniques are very complex to create trees and graphs respectively and metric based technique is not efficient to detect all possible code clones. Hybrid technique is very effective, because by this technique we can enhance the properties of combined technique. Example hybrid technique by combining token based and metric based technique, it eliminates some of limitation of both combined techniques, unlike token based technique it can detect more than type 1 and type 2 clone and unlike metric based technique it can detect more probable clones, i.e. hybrid technique enhance efficiency and correctness of the software or program.

REFERENCES

[1] Rajkumar Tekchandani, Rajesh Kumar Bhatia and Maninder Singh "Semantic Code Clone Detection Using Parse Trees and Grammar Recovery", IEEE, 2013

[2] Amandeep Kaur , Balraj Singh " Study on Metrics Based Approach for Detecting Software Code Clones", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 1,January 2014.

[3] Kanika Raheja, Rajkumar Tekchandani., "An Efficient Code Clone Detection model on java byte code using hybrid approach", IEEE, SEPT 2013.

[4] Geetika*, Rajkumar Tekchandani, " Detection of Potential Clones from Software using Metrics", IJARCSSE, Volume 4, Issue 4, April 2014.

[5] Deepak sethi, Manisha sehrawat and Bharat Bhushan Naib, "Detection of code clones using datasets" IJARCSSE, Volume 2, Issue 7, july 2012.

[6] Yoshiki Higo, Yasushi Ueda, Minoru Nishino, Shinji Kusumoto, "Incremental Code Clone Detection: A PDG-based Approach", IEEE, 2011.

[7] Mai Iwamoto, Shunsuke Oshima, Takuo Nakashima, "Token-based Code Clone Detection Technique in a Student's Programming Exercise", IEEE, 2012.

[8] Tahira Khatoon, Priyansha Singh, Shikha Shukla" Abstract Syntax Tree Based Clone Detection for Java Projects" IOSR Journal of Engineering,Volume 2,Issue12,Dec 2012

[9] http://www.wikipedia.org/wiki/Abstract_syntax_tree

