January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

Comparative Review of Code Clone Detection Tools and Techniques

Neha Saini*, Sukhdip Singh**
*Assistant Professor, Computer Science Department , Government College Chhachhrauli, Yamunanagar, Haryana.
** Associate Professor, Computer Science & Engineering Department , DeenbandhuChhotu Ram University of Science & Technology, Murthal , Sonipat, Haryana.

Article Info
Volume 82
Page Number: 4989 - 4997
Publication Issue:
January-February 2020

Article History
Article Received*: 18 May 2019*
Revised*: 14 July 2019*
Accepted*: 22 December 2019*
Publication*: 25 January 2020*

Abstract:

Code reuse is a typical movement in programming improvement and is one of the primary reasonsbehind code clones and programming advancement. A codeclone is a part of the source code that is identicalor highly similarto another part (clone) as with reference to structure and semantics. There have been various correlation and assessment concentrates to relate those which gave noteworthy commitments to the clone detectionresearch. These also exposed how challenging it is to compare different tools for certain reasons. There is no examination that comprehends which device or system works better in various sorts of code clones. In this exploration venture, an exhaustive examination is given on the exhibitions of as of now accessible clone detectiontools and techniques.

Keywords: *Code Fragment,Code Clone, Clone Detection*

1. Introduction

Code cloning or the act of copying code sections and making minor, non–practical adjustments, is a notable issue for developing programming frameworks prompting duplicated code fragments or code clones [1]. Obviously, the typical working of the framework isn't influenced, yet without countermeasures by the upkeep group, further advancement may turn out to be restrictively costly. Luckily, the issue has been contemplated seriously and a few methods to both distinguish and remove duplicated code have been proposed in the writing [2].

Code cloning is observed to be a progressively difficult issue in mechanical programming frameworks. In nearness ofclones, the typical usefulness of the framework may not be influenced, yet without countermeasures by the upkeep group, further advancement may turn outto be restrictively costly. Code clones may antagonistically influence the product frameworks' quality, particularly their viability and understandability [3]. For instance, if a bug is found ina codefragment, all of its similar cloned sections ought to be distinguished to fix the bug being referred to. In addition, to an extremecloning builds the framework size and regularly demonstrates structure issues, for example, missing legacy or missing procedural deliberation. In spite of the fact that the expense of keeping up clones over a framework's lifetime has not been evaluated at this point, it is at any rate concurred that the budgetary effect on support is high [4]. Grubb gauges the expenses of changes

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

did after conveyance at 40% - 70% of the all-out expenses during a framework's lifetime. Existing exploration demonstrates that a lot of code of a product framework is cloned code also, this sum may shift contingent upon the space and source of the product framework [5].

2. Software Clones

2.1. Clone Types

Code can be cloned in a few different ways. There are four primary sorts of likeness between code parts. Parts can be comparative dependent on the similitude of their program content, or they can be comparable dependent on their usefulness (autonomous of their content). The primary sort of clone is frequently the consequence of replicating a code part and sticking into another area. In the accompanying the kinds of clones are given dependent on both the printed (Types 1 to 3) and practical (Type 4) similitudes [6]:

- **Type-1:** Identical code parts aside from varieties in whitespace design and remarks. Fig 1.1(b) demonstrates this kind of code clone.

- **Type-2:**Grammatically indistinguishable sections aside from varieties in identifiers, literals, types, whitespace, design and remarks. Fig 1.1(c) demonstrates this sort of Replicated sections with further adjustments, for example, changed, included or evacuated code clone.

- **Type-3:**proclamations, notwithstanding varieties in identifiers, literals, types, whitespace, design and remarks. Fig 1.1(d) demonstrates this kind of code clone.

- **Type-4:** Two or more code actualized by various syntactic variations. Fig 1.1(e) demonstrates this kind of code clone.

```
Void abc(int n) {
Float sum = 0.0; //C1
Float proud = 1.0;
For (int i=1; i<=n; i++)
    {sum=sum + i;
     Prod = prod * I;
Foo(sum, prod); } }
```
(a)

```
Void abc(int n) {
Float sum = 0.0; //C1
Float proud = 1.0;
For (int i=1; i<=n; i++)
    {sum=sum + i;
     Prod = prod * I;
Foo(sum, prod); } }
```
(b)

```
Void abc(int n) {
Float sum = 0.0; //C1
Float proud = 1.0;
For (int j=1; j<=n; j++)
    {s=s + j;
     P = p * j;
Foo(s, p); } }
```
(c)

```
Void abc(int n) {
Float sum = 0.0; //C1
Float proud = 1.0;
For (int i=1; i<=n; i++)
    {sum=sum + i;
     Prod = prod * I;
Foo(sum, prod); } }
```
(d)

```
Void abc(int n) {
Float sum = 0.0; //C1
Float proud = 1.0;
int i=0;
While (i<=n)
    {sum=sum + i;
     Prod = prod * I;
Foo(sum, prod);
i++;} }
```
(e)

Figure 1.1: Clone types

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

2.2. Clone Terminology

All clones are recognizing as Clone Classes and ClonePairs. Clone classes and clone sets tell about the likeness between different code clone sections. On the off chance that they have some comparable successions in the code, clone-relation exists betweenthe code parts. For instance character strings, strings without void area, changed token arrangements and groupings of token kind so on [4].

***A. Code Fragment**:*

A Code Fragment (CF) is any sequence of code lines (with or without comments). Clone isdetected utilizing correlation between the sections in a source code. It tends to be of any sorts ofcode, for instance work definition, start end square, or arrangement of articulations. A CF is recognized by its record name and start end line numbers in the first code base. Let CF1 and CF2 are two code pieces. CF2 is a clone of CF1 on the off chance that they are comparable by some given meaning of likeness, that is, f(CF1) = f(CF2) where f is the closeness work. A similitude capacity can be characterized in different ways, for example, precise match between parts, coordinate pieces in the wake of evacuating the remarks or normalizing identifiers. Two parts that are like each other structure a clone pair (CF1, CF2), and when numerous sections are comparable, those structure a clone class or clone group.

***B. Clone Pair*:**

If there is any clone connection exists in the pair ofcode parts then it is known as a clone pair or clone pair is a couple of code part having some likeness between them.

***C. Clone Set*:** A set of all the identical or similar fragments [6].

***D. Clone Class*:**A lot of all the clone matches in which the current clone sets having a few clone Based on functionalities and program content, two.

***E. Code Clone Types*:** On the basis of functionalities andprogram text, two code parts are said to be comparable. The main sorts of clone are chiefly the aftereffect of reorder exercises. In the accompanying sort of clones Type I, Type II and Type III clones depend on the literary closeness and Type IV clones depend on the practical closeness [7].

2.3. Clone Detection Techniques

This segment characterizes the methods for code clonedetection.

i. Textual Approach (Text Based technique):

This approach has no source code change before the correlation being drawn on the two sides. In assortment of cases, the first source code is used as it is introduced during the time spent clone recognition. For instance, NICAD, SDD, Simian 1, etc [5].

ii. Lexical Approach (Token Based technique)

This method at first believers the source code in the lexical arrangement, known as tokens by using compiler style lexical examination. The grouping later sweeps the not required duplication of token succession by methods for unique code that is come about as clones. These kinds of methodologies are typically stronger for little varieties in the code [6]. It is characterized as separating, arranging and renaming which is distinctive as contrast with literary systems. For instance CC Finder, Dup, CPMiner, etc [7].

iii. Syntactic Approach

This methodology uses a parser for changing over a source program in unique language structure trees or parse trees that can be prepared by utilizing basic measurements or tree coordinating for finding the clones. For instance: Deckard, Clone Dr andCloneDigger, etc [8].

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

iv. Semantic Approach

This approach has been developed by utilizing the static program because it gives the in-depth data as compare tothe syntactic similarity. In different approaches, theprovided approach is given in the form of PDG (Programdependency graph) or in the form of statements orexpressions but the edges shows the data or controldependencies. For example, GPLAG, Duplix and so on [7].

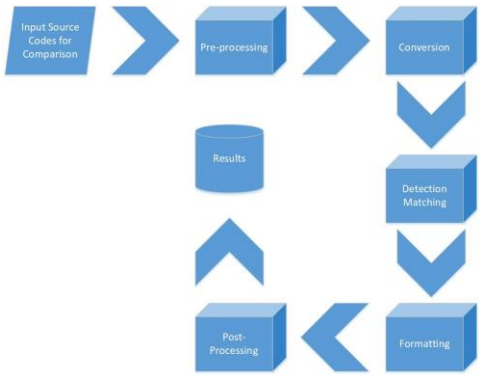

Figure 3: Process of clone detection

Figure 3is characterizing the general strides during the time spentfor codeclone detection [6].

Table 1: Clone Code Classification and techniques.

	Text based	**Token Based**	**AST Based**	**PDG Based**
Category	Textual	Textual	Semantic	Semantic
Supported Clone	Type 1	Type 1, 2	Type 1, 2, 3	Type 1, 2, 3
Complexity	O(n)	O(n)	O(n)	$O(n^3)$
n Meaning	Lines of code	Number of token	Node of AST	Node of PDG

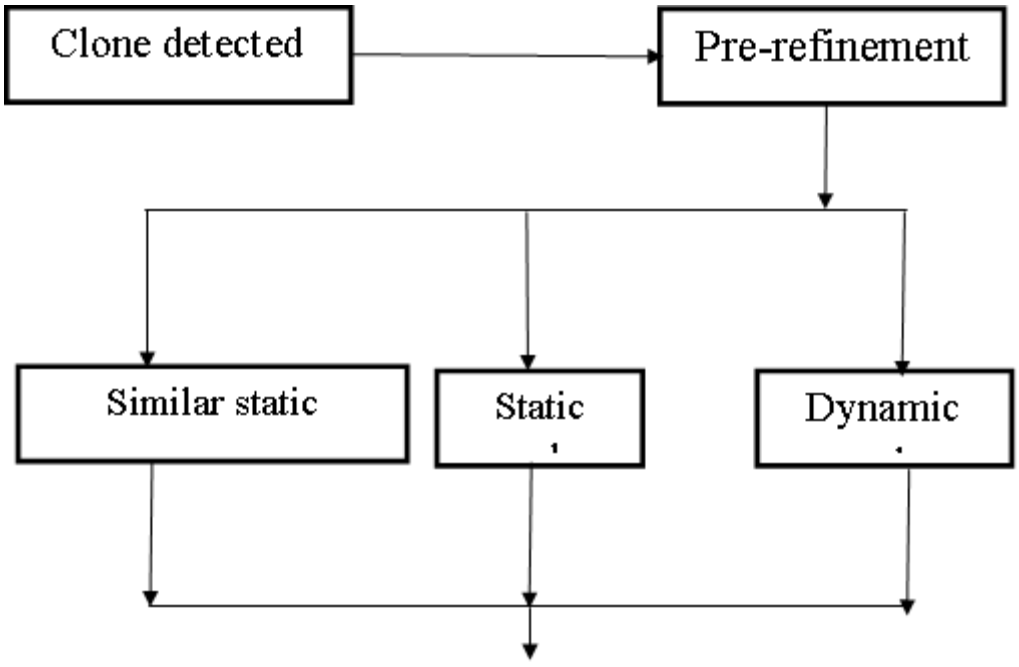

Figure 4: Process of Clone removal

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

Figure 4 is defining the code clone evacuation process. After the discovery of clone, the reasonable view is seen just. On the off chance that it is same as the clone which is being distinguished, than same strategy will be suggested. On the off chance that the code is static, since it is language free than a novel technique will be connected and if the code is dynamic than the execution is finished with the assistance of instruments [8].

2.4. Code Analysis

Source code investigation is the mechanized testing of source code to investigate a PC program or application before it is appropriated or sold. The source code is the most changeless type of a program, despite the fact that the program may later be adjusted, improved or updated. Source code examination can be either static or dynamic [10].

- In static investigation, troubleshooting is finished by analyzing the code without really executing the program. This can uncover mistakes at a beginning time in program improvement, regularly wiping out the requirement for various amendments later.

- After static examination has been done, unique investigation is performed with an end goal to reveal progressively unpretentious imperfections or vulnerabilities. Dynamic examination comprises of ongoing project testing.Clone is detected through static analysis of a source code.

3. Overview of the Implemented Tool

A relative report among code clone recognition devices and procedures are proposed in this examination venture. The grids of the correlation are the clone types. For this examination, the accompanying apparatuses and systems are considered [9] namely Johnson, SDD, CCFinder, CPD, CPMiner,CloneDigger and CloneDr. CP-Miner and CPD is free device, SDD and clone Dr are found as overshadowing module and clones is a visual studio clone recognition highlight. The other three strategies will be executed as an instrument. The outline of the executed apparatus is shown in Figure 5 [11].

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

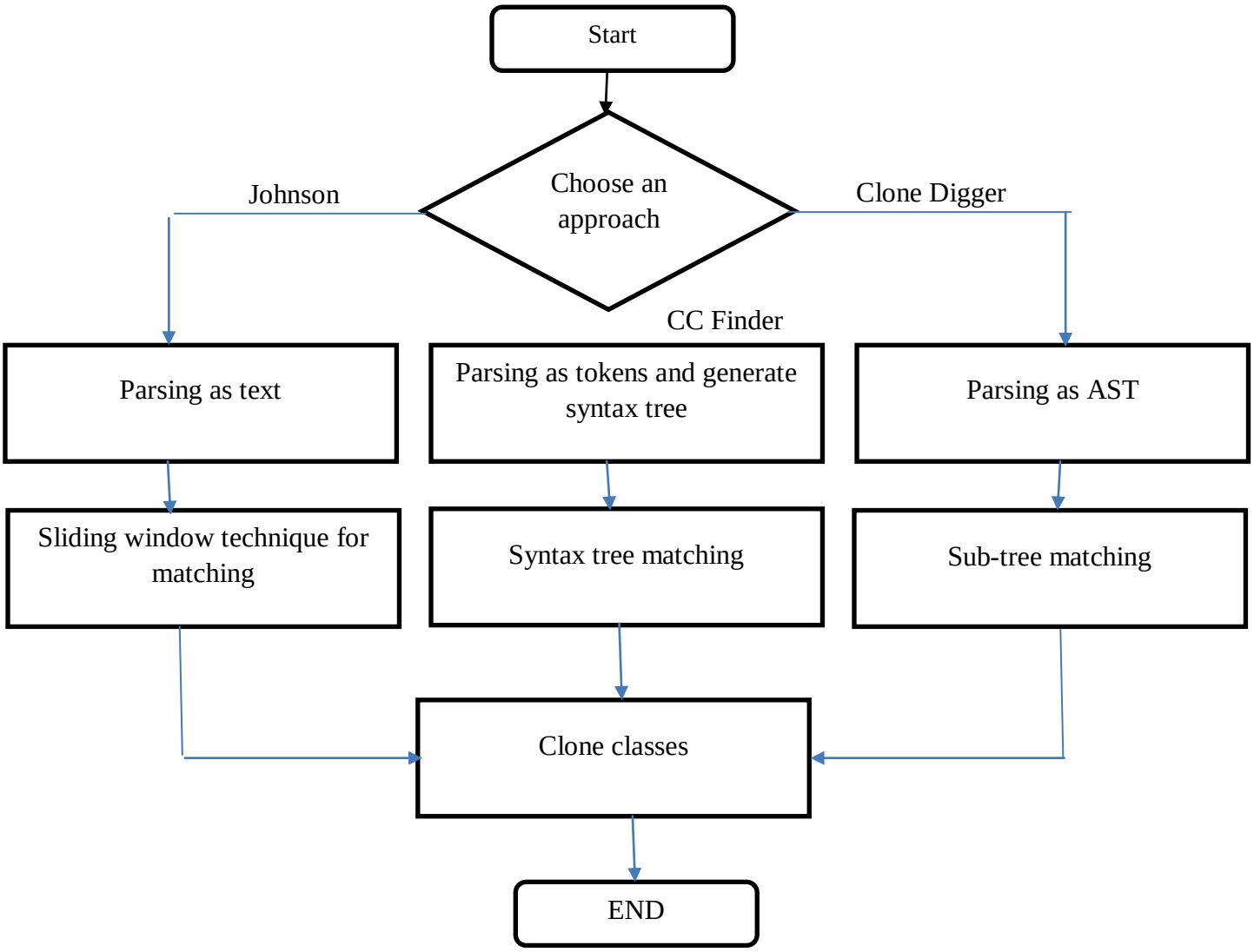

Figure 5: Overview of the implemented tool.

Client will give a java venture as information a java undertaking and give a decision which approach s/he is happy to utilize. At that point the devices create clone classes utilizing the picked methodology [5]. Johnson is a content based methodology of distinguishing clones. It parses the entire source code as content and matches the codeparts utilizing sliding window procedure. CC Finder is a token-based methodology which initially standardizes the identifiers and parses the stndardized source as tokens. At that point create postfix tree utilizing the token successions and play out a tree coordinating calculation to identify the clones. Then again CloneDigger is a tree-based clone identification approach. At first it produces Abstract Syntax Tree (AST) by parsing the source code. At that point coordinates the sub-trees to recognize the clones [7].

3.1.Description of implemented tool

The device contains three methodologies. Those are Johnson, CCFinderandClone Digger

A short depiction of the methodologies is given in the accompanying sub-areas.

3.1.1. Johnson

It is the soonest approach of clone identification. It is a content based clone identification system where the source is considered as content and

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

broke down by the manner in which archives are examined. It doesn't utilize any adjustment in the source code while coordinating. The sliding window system utilized in the coordinating period of this methodology. The methodology can be condensed as [8].

1) For each record being considered, apply a content to content change to dispose of characters not to be considered for coordinating. For this examination, this is a personality change (yield equivalents input). Be that as it may, different kinds of surmised coordinating can be obliged by disposing of various pieces of the information [4].

2) Create a lot of substrings that spread the source (i.e., each character of content shows up in at any rate one substring) [12].

3) Recognize which of the substrings coordinate (i.e., have a similar grouping of characters).

4) Change this database of crude matches into a structure that all the more briefly communicates a similar data [11].

5) Perform task-explicit information decrease.

6) Abridge abnormal state matches.

Steps (2) and (3) are data gathering stages, (4) is a data protectingchange, (5) an accumulation and rearrangements stage, and (6) the introduction of results in a valuable structure. Stage (1) gives more prominent affectability to specific sorts of info [9].

3.1.2. CCFinder

CCFinder is token-based code clone recognition device. CCFinder utilizes an addition tree calculation with both reality complexities O(mn), where m is the greatest length of included clones and n is the absolute length of the source record. On the off chance that it would be expected that m does not rely upon n and it is limited by some fixed length, the existence complexities will basically be O(n) [7].

The optimizations employed inCCFinder to handle large source files are as follows:

Alignment of Token Sequence:

Source code has its innate granularity, for example, character, token, explanation, or square. Code segments of a code clone should start at their limit. For instance, a code divide, which starts at the center of an announcement X and closures the center of an announcement Y, is less valuable than a code partition which starts toward the start of Y. As a basic sifting for this reason, it permits just explicit tokens toward the start of clones as driving tokens. Watchwords that start explanations are driving tokens [4]. In C and C++ source records, those watchwords are '#', '{', catchphrases for choice proclamations (else, if, switch, and so on.), cycle articulations (do, for, and keeping in mind that), bounce or organized special case taking care of explanations (break, get, return, and so forth), and assertions (class, enum, typedef, and so on). Likewise, tokens following watchwords that end explanations (';', ')'} or marks (':') are additionally driving tokens [9]. The quantity of hubs in the addition tree was diminished to 33% by this sifting. This method may somewhat decrease the affectability of clone identification, however for all intents and purposes it is imperative to make the procedure versatile.

Repeated Code Removal:

Repetition of a short code portion tends to generate manyclone pairs. For example, consider the following code [10]:

```
switch (c) {
case '0' : value = 0; break;
case '1' : value = 1; break;
case '2' : value = 2; break;
case '3' : value = 3; break;
case '4' : value = 4; break;

}
```

Now, consider that the following code section is

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

also included in the target source files:

```
case 'a':
flag = 2;
break;
```

For this situation, five code parts make a clone class, { a2-a2,a3-a3, ..., a6-a6, b1-b3 }, where each pair of the code segments makes a clone pair, and the quantity of maximal clone sets are 6C2 = 15, altogether [11]. To maintain a strategic distance from this blast of clone combines, a heuristic methodology is presented. After structure a postfix tree, if a reiteration of a2 is distinguished at a3, the succeeding redundancy segment (a3-a6) isn't deliberately embedded into the tree, with the goal that a piece of the clone sets isn't being accounted for. Be that as it may, the clone pair (a2-a2, b1-b3) is still extricated, which offers adequate data [9]. The rehashed code expulsion process likewise anticipates location of self-clones, e.g., (a2-a5, a3-a6), or redundancy of "steady" revelations [10].

Concatenation of Tokens:

Just before processing the match in the token succession, adjoining tokens, with the exception of punctuator catchphrases, are connected. This procedure decreases the length of a token grouping in return for an expansion in variety of the tokens [11].

Division of Large Archive of Source Files:

In the event that the all-out size of source documents surpasses the memory space for a solitary postfix tree, the apparatus consequently utilizes a 'separate and overcome' approach. The info source documents are isolated into a few sections [9]. For every mix of the parts, a sub-postfix tree is worked to concentrate clone sets. The all out accumulation of clone sets will at last be the yield. Give m a chance to be the quantity of subsets of source documents, and after that the quantity of sets of the lumps (i.e., the quantity of built subsuffix trees) is mC2. In this way, the time multifaceted nature moves toward becoming O($m2$) [10].

4. Conclusion

Code cloning is a procedure of reusing the code all things considered or with a few changes. Code clone acknowledgment is a forte of perceiving the substance equivalence between the ventures or Web Pages. An undertaking is made to arrangement a methodology called "SDCode Clone Detection" for both static and dynamic Web Pages. A device was built up that actualizes three methodologies, that are picked for assessment however usage isn't accessible, to play out an examination on comparison among code clonedetection techniques.The tools of code clonemust be incorporated in standard IDE for having across the board reception. This paper predominantly centers on depicting the recognition systems with the techniques for code clone technique. The procedure of expulsion is additionally talked about. The code clone discovery assumes a crucial job in the exploration of programming advancement wherein the properties of comparable code element are watched for number of variants. It is being inferred that the clones are the noteworthy viewpoints for programming development. In the event that the framework must be developed than its clones should make consistent variations.

References

[1]. **Gundeep Kaur and Er. Sumit Sharma [2018],** The Survey of the Code Clone Detection Techniques and Process with Types (I,II, III and IV), International Journal on Future Revolution in Computer Science & Communication Engineering ISSN: 2454-4248Volume: 4 Issue: 2, IJFRCSCE | February 2018.

[2].**Ekta Manhas and Er.Samriti Rana [2017],** Code Clone Detection: A Review And Comparative Analysis, Web Site:

January - February 2020
ISSN: 0193 - 4120 Page No. 4989 - 4997

www.ijettcs.org Email: editor@ijettcs.org Volume 6, Issue 5, September- October 2017.

[3].**DHAKA, BANGLADESH [2016]**,Clone Type Based Comparison among Code Clone Detection Tools and Techniques.

[4].**Hahid Ahmad Wani and Shilpa Dang [2015]**, A Comparative Study ofClone Detection Tools, International Journal of Advance Research inComputer Science and Management Studies, Volume 3, Issue 1, January 2015.

[5]. **Filip Van Rysselberghe and Serge Demeyer [2015]**, Evaluating Clone Detection Techniques,http://www.informatik.uni-stuttgart.de/ifi/ps/clones/.

[6]. **Kuldeep Kaur and Dr. Raman Maini [2015]**, A Comprehensive Review of Code CloneDetectionTechniques, ISSN 2278 – 2540, Volume IV, Issue XII, December 2015.www.ijltemas.in

[7].**Jeffrey Svajlenko and Chanchal K. Roy [2014],** Evaluating Modern Clone Detection Tools, jeff.svajlenko, chanchal.roy.

[8].**H. Murakami, Y. Higo and S. Kusumoto [2014],** "A dataset of clone referenceswith gaps," in MSR, 2014, pp. 412–415

[9].**D. Rattan, R. Bhatia and M. Singh [2013],** "Software clone detection: Asystematic review," Information and Software Technology, vol. 55, no. 7,pp. 1165 – 1199, 2013

[10].**Saeed Shafieian and Ying Zou [2012]**, Comparison of Clone Detection Techniques Technical Report 2012-593.

[11]. **Chanchal K. Roy, James R. Cordy and Rainer Koschke [2009],** Comparison and Evaluation ofCode Clone Detection Techniquesand Tools: A Qualitative Approach, February 24, 2009.

[12]. **Stefan Bellon and Rainer Koschke [2007]**, Comparison and Evaluation of Clone Detection Tools, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 9, SEPTEMBER 2007.

[13].**Elizabeth Burd and John Bailey [2002]**, Evaluating Clone Detection Tools for Use during Preventative Maintenance, The Research Institute in Software Evolution University of DurhamSouth RoadDurham0-7695-1793-5/02 $17.00 © 2002.