Detection of Software Cloning by using Visual Detection Technique with Result Analysis

Harish Patidar, Amit Mishra, Shiv K. Sahu

Abstract: Code duplication or copying a code fragment and then reuse by pasting with or without any modifications is a well known code smell in software maintenance. Many record show that about 5% to 20% of a software systems can contain duplicated code, which is basically the results of copying existing code fragments and using then by pasting with or without minor modifications. Researchers think clones lead to additional changes during maintenance phase, in later stage increase the overall maintenance effort because of to find modified cloned code in same or another code file within time period with higher accuracy for any kind of modification in it. This project use visual detection technique to find the clone code. Visual detection technique uses near-miss clones detection method to find clones in a program file with higher accuracy and give results better than current Clone detection techniques.

Keywords: Code clone, Web server application, Visual detection, Token and flag.

I. INTRODUCTION

In computer a program code is used as Duplicate code in another program by copy and paste [3] [5] [17], this duplicate code is called clone code. The working of clone code is similar as originally code. For example if we want to make a clone of a person, we create clone body, voice, face, hair. Clone is similar to person in body, voice, face and hair but habit and mind power is not match means clone is functionally same but logically different. Mainly four types of cloning: Chanchal Roy [12], S. Bellon, R. Koschke, G. Antoniol, J. Krinke and E. Merlo [10]. Type-1.(a) Difference whitespace. (b) Changing in Comments.

Type-2.(a) Renaming of identifiers. (b) Renaming of operator.

Type-3.(a) Modification code. (b) Add new lines. (C) Delete some lines.

Type-4.(a) Reordering Statements. (b) Control displacements**.**

- Code character-to-character similar.
- Code character-to-character similar with comments and white.
- Codes are token-to-token same.
- Codes are functionally same.
- Codes are logically same.
- Identifier and variable are same

A. How duplicates are created

Copy and paste is the best way to create the clone code. In object oriented programming code reuse property is apply, Level of piracy is increase by this property. Functionality is the way to create cloning. Some codes are functionally identical but to change in declaration of variable [14] [16]. Mainly personal used software is implemented to duplicated code, level of piracy is increase day by day because code re-usability is do in maximum software. Reusable code is error free and reduces programming time [12]. Find the several reusable codes, calculate the actual programming effort. Two duplicated codes are functionally same but nature and characteristics are difference. Behavior (Object oriented nature) different. Two different addresses, to compared at the one platform and find the level of cloning.

B. Block Diagram

Firstly take source and target programs, normalized the both codes and remove white space, comments. Then filtration is second steps for removing regular language key and header files, then compare, find out type-I clone. For Near miss [19] clone firstly take sample program and call web application program from web server. Tokens are generated of each Code, and then tokens are converted into flags. For matching similarity apply visual detection method and finding code clone. Tokenization**:** In the token-based approaches [3] [11], each line of the source code is converted into tokens with the help of lexical rule of the programming language. First each line of source files is converted into tokens by a laxer and the tokens of all source lines are then concatenated into a single token sequence.

II. PROBLEM IDENTIFICATION

Many clone detection method to find simple cloning, variation in code is not find-out properly. At present time most of clone detection technique to detect only copy past code. Base paper [5], technique is only find Type-1 and Type-2 code clone. Accuracy level range is 30-40%. Traditional methods [8] do not solve the problem of server side clone detection and not to provide application of web. In object-oriented application [16] [19], code reuse property is used inheritance, polymorphism, encapsulation, and those features to growing up the cloning. Logical change in code is not detected by traditional method. Most of the method not to support multi language program code detection. If the base code is error full then it's all clone code is contain bug. In old approach long code size clone is not measure at accurately. It is also find the misuse of reusability property in software field. New programming effort is reduce and quality of code programming is down.

III. OBJECTIVE

In this paper, automatic clone detecting technique is used, which is based on visual detection technique to provide maximum number of copy-past. Firstly present technique for structural clones, by lexical analysis. It is to find type-1, type-2 & type-3 clone detection. Data stored

Revised Version Manuscript Received on June 18, 2016.

Harish Patidar, M.Tech Scholar, Department of Information Technology, Technocrats Institute of Technology, Bhopal (M.P). India.
Amit Mishra, Assistant Professor, Department of Information Technology, Technocrats Institute of Technology, Bhopal (M.P). India.
Dr. Shiv K. Sahu, Associate Professor & Head, Department of Information Technology, Technocrats Institute of Technology, Bhopal (M.P). India.

into . Detection technique is based on C++ and GUI. Data evaluated the performance by analyzing structural clones found in software systems. On detecting code clones of code fragments, it saves comprehension time and space. It is believe that this technique is scalable and useful. On detecting code clones, the quality of code is improved. This tool detects a significant amount of code clones. Identity and subsequent uniform of simple clones is helpful in software management. Its main goal is identify clones and quantify the amount of similarity present.

IV. METHODOLOGY

In this Approach Visual detection method (Token based and Normalization based techniques) used to find out quantity of copy-past. Data stored into SQL/MYSQL. Detection technique is based on C++ and GCC compiler. Clone detection method to find code clone between two program, one is sample program other is server related web application. In this way it can find [4] the maximum similarity between two programs. Comparison between the two programs is show the level of originality. Two programs which are belong to.

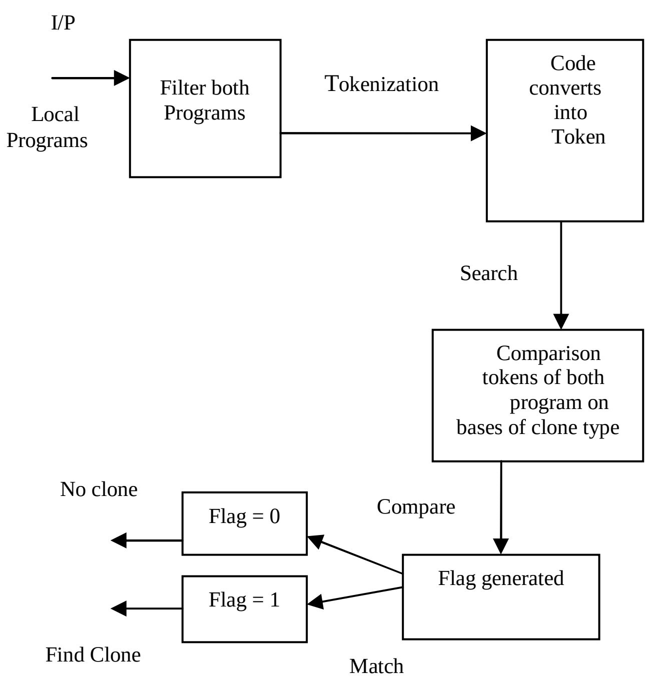


Fig.1. Block diagram of Near miss clone detection

Example: Input Source Code [14],
If (p>=q) q=20;
Generated Token is:
<IF> <LPAREN> <ID (P)> <GEQ> <ID (Q)> <RPAREN> <ID (Q)> <BECOMES> <INT (20)> <SCOLON>

The token sequence is then transformed into flag. In programming language a SEARCH FLAG is used for detection or matching. Time period is given to each flag. [13] Number of flag is equal to number of flag, means each search process is generated flag each time.

In this paper improve working of visual detection approach using the web based application and searching pointer is work as flag.

B. Flow chart

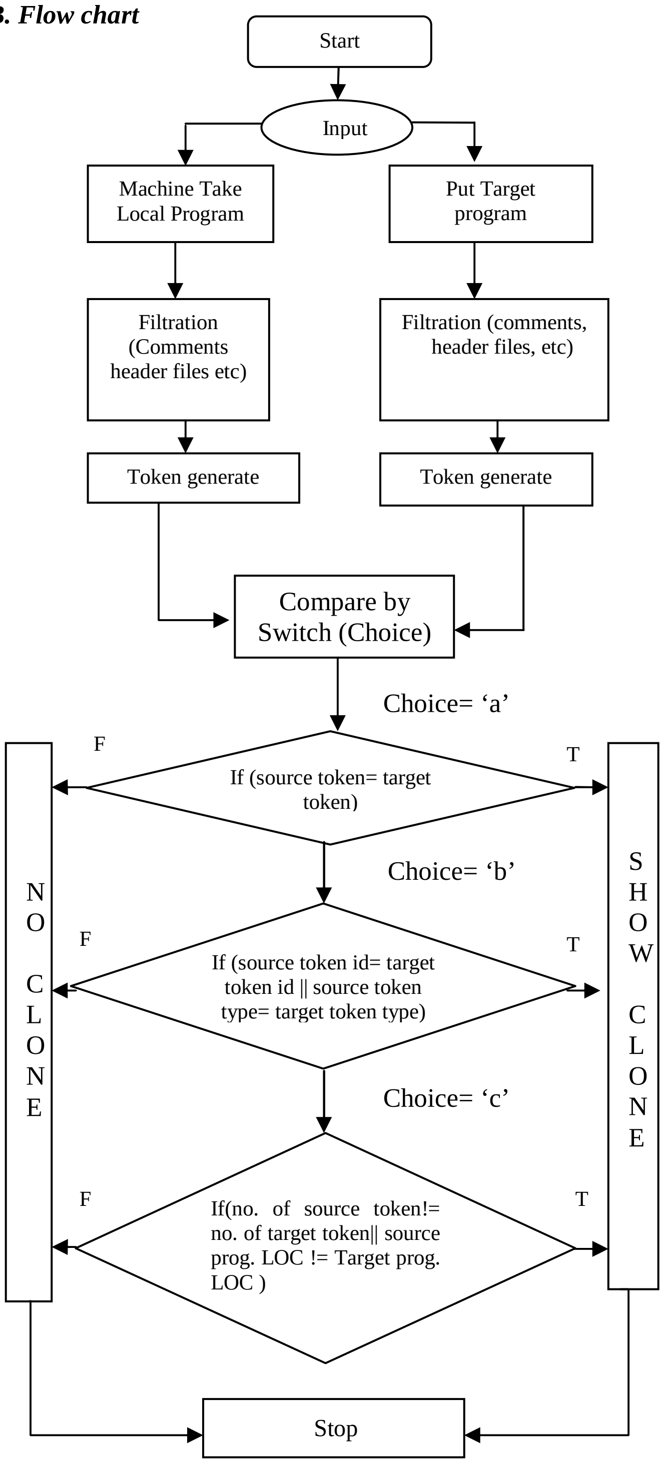

Start
Input
Machine Take Local Program
Put Target program
Filtration (Comments header files etc)
Filtration (comments, header files, etc)
Token generate
Token generate
Compare by Switch (Choice)
Choice= 'a'
If (source token= target token)
Choice= 'b'
If (source token id= target token id || source token type= target token type)
Choice= 'c'
If(no. of source token!= no. of target token|| source prog. LOC != Target prog. LOC)
F
T
NO CLONE
SHOW CLONE
Stop

Fig.2. Flow chart of near miss clone

Flag-based Techniques: This is search based technique [5], token as input to symbol that is used for matching similarity, after search value of flag is zero that means no cloning, if value of flag is one then clone code is find out. Traverse & Select methods are used to find the appropriate item set. Then flag is used as input for visual detection, to find the code clone. Flag search period is depend on flag variable which is contain limited time depend on user.

C. Proposed algorithm

1) Algorithm for Near miss clone detection

I/P: Source Code & Target code.

```
O/P: Report (Clone absence / Copy of Clone code in
     Target program)
Step 1: Take sample programs as a input. // Source &
     Target code
Step 2: Filtration of both codes
     // Removing the comment, white space, header
     files, comma, brackets etc.
Step 3: Generation of Tokens for both programs.
Step 4: Compare tokens of both program by switch
     statement.
     Choice= 'a'
     If (source token=target token)
     Exact match.  // Type-1 Clone
     Break;
     Choice= 'b'
     if (source token id= target token id || source token
     type= target token type)
   Search renamed identifiers, keywords, and literals
                    // Type-2 Clone
     Choice= 'c'
     If (no. of source token!= no. of target token|| source
     prog. LOC != Target prog. LOC)
     Search modified line, added new lines, and deleted
     lines.      //Type-3 Clone
     iv. Search re-ordering and control replacement.
     //Type-4 Clone
Step 5: Report Generation.
   If
        Flag = 1;
```

V. CONCLUSION

Proposed and evaluated a new approach to web application clone search using normalization, filtration and visual detection. This approach is also useful for server side program to finding cloning. It is like open system software to apply different web application. It is solve the problem of long code, which is used as copy-past and find-out easily. It is automatic filter based detection method. Traditional method is useful for limited type of program but this method is use for all server side programs.

This approach may find type-1, type-2 and Type-3 clones. Result may generate in chart wizard and comparison formats.

VI. RESULT ANALYSIS

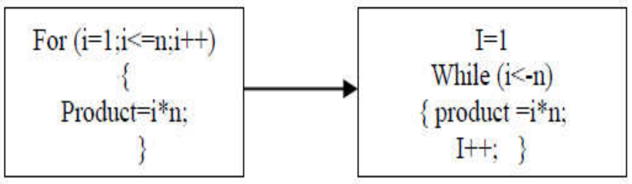
Proposed system on the window-XP or window-7 & 8 and source code in C++ with advance 32 bit GCC 4.8.1 compiler. Type-3 clone code based on modification of line, addition of new line and deletion of lines.Type-4 clone code based on reordering and control replacement. Here some specific symbols are used four type clone. Type-1 is use –a, type-2 for –b, type3 for –c and type-4 use – d. With the help of cmd command firstly point the drive where program file is stored and then name of program folder, name of .exe file.

Example:

```
F:\cd clonecode/clonecode.exe/source.cpp
```

Here <clonecode> is program folder, <clonecode.exe> is .exe file,

<-d> is symbol for type-4 clone and <source.cpp> <target.cpp> is C++ program files.



```
For (i=1;i<=n;i++)
{
Product=i*n;
}
```

```
I=1
While (i<-n)
{ product =i*n;
I++;  }
```

Figure 6.1. Compare source and target program, detect type-4 clone code

In type-4 clone is based on reordering and control based, control based changes in coding is fully change the structure of program. But logical concept of coding is same. As a token-based approach, Boreas matches the variables, rather than matching sequences or structures. Using this idea, the similarity of two code segments is decided by the proportion of variables that could be matched based on their characteristics.

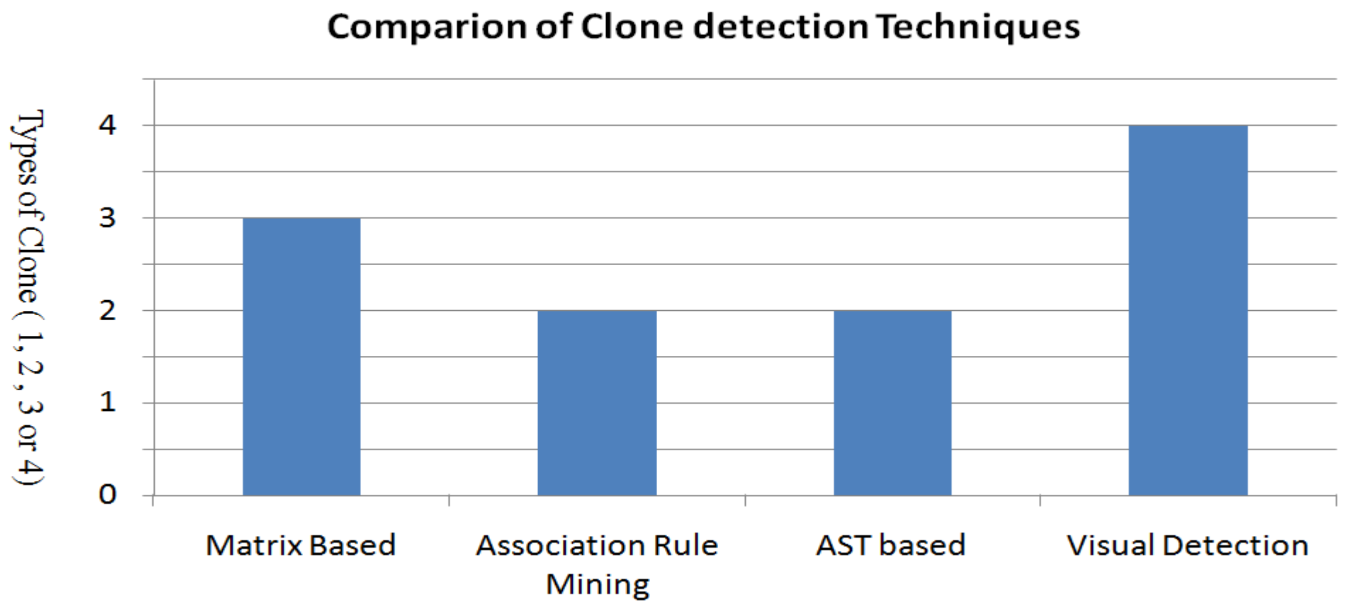
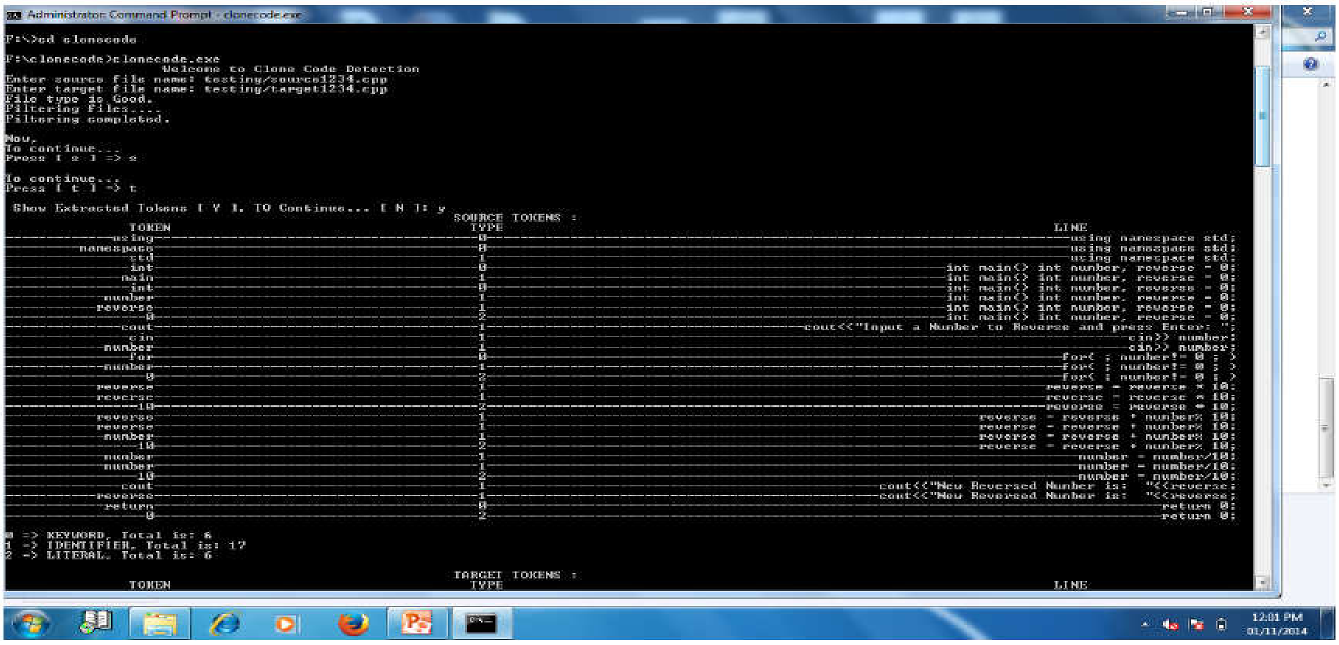
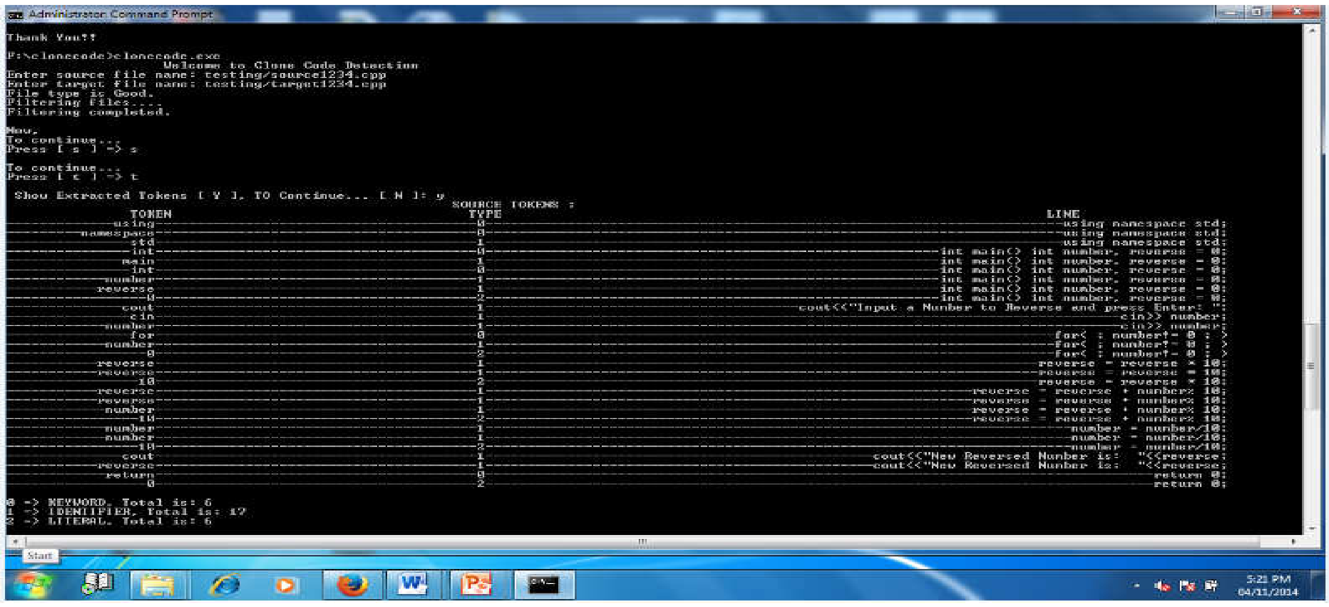


Fig.3. Result Analysis



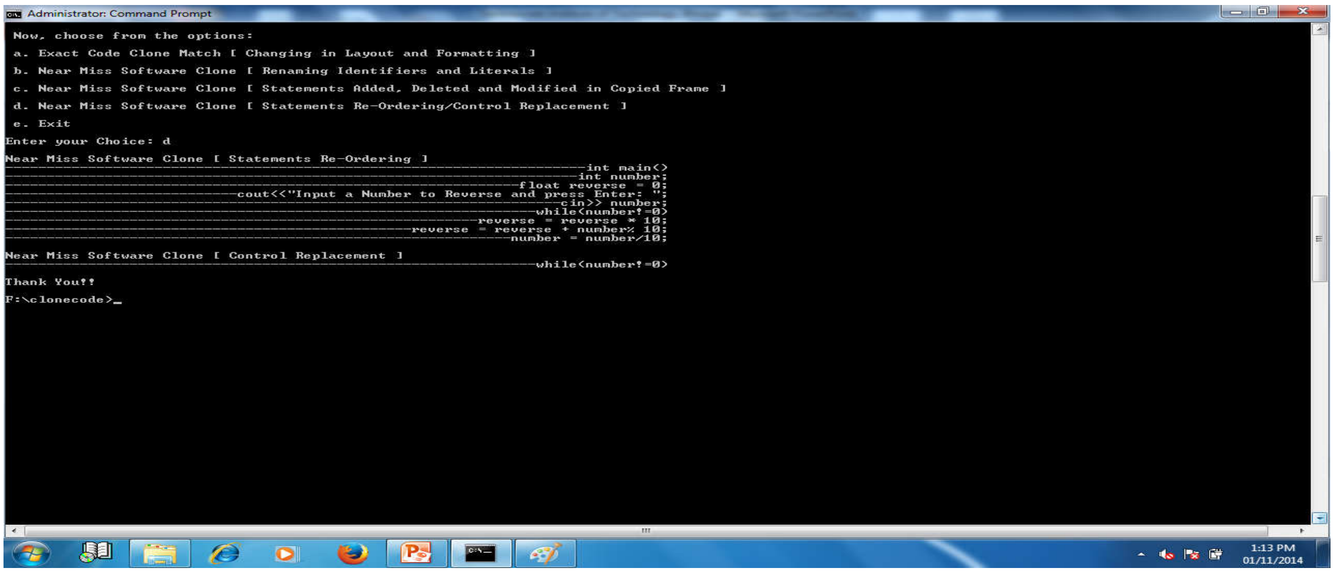
```
F:\>cd clonecode

F:\clonecode>clonecode.exe
                   Welcome to Clone Code Detection
Enter source file name: testing/source1234.cpp
Enter target file name: testing/target1234.cpp
File type is Good.
Filtering files....
Filtering completed.

Now,
To continue...
Press [ s ] => s

To continue...
Press [ t ] -> t

 Show Extracted Tokens [ Y ], TO Continue... [ N ]: y
                     SOURCE TOKENS :
       TOKEN           TYPE                                                     LINE
-------using--------0-----------------------------------------using namespace std;
---namespace--------0-----------------------------------------using namespace std;
---------std--------1-----------------------------------------using namespace std;
---------int--------0--------------------------int main() int number, reverse = 0;
--------main--------1--------------------------int main() int number, reverse = 0;
---------int--------0--------------------------int main() int number, reverse = 0;
------number--------1--------------------------int main() int number, reverse = 0;
-----reverse--------1--------------------------int main() int number, reverse = 0;
-----------0--------2--------------------------int main() int number, reverse = 0;
--------cout--------1---------cout<<"Input a Number to Reverse and press Enter: ";
---------cin--------1------------------------------------------------cin>> number;
------number--------1------------------------------------------------cin>> number;
---------for--------0----------------------------------------for( ; number!= 0 ; )
------number--------1----------------------------------------for( ; number!= 0 ; )
-----------0--------2----------------------------------------for( ; number!= 0 ; )
-----reverse--------1--------------------------------------reverse = reverse * 10;
-----reverse--------1--------------------------------------reverse = reverse * 10;
----------10--------2--------------------------------------reverse = reverse * 10;
-----reverse--------1------------------------------reverse = reverse + number% 10;
-----reverse--------1------------------------------reverse = reverse + number% 10;
------number--------1------------------------------reverse = reverse + number% 10;
----------10--------2------------------------------reverse = reverse + number% 10;
------number--------1------------------------------------------number = number/10;
------number--------1------------------------------------------number = number/10;
----------10--------2------------------------------------------number = number/10;
--------cout--------1-------------------cout<<"New Reversed Number is:  "<<reverse;
-----reverse--------1-------------------cout<<"New Reversed Number is:  "<<reverse;
------return--------0----------------------------------------------------return 0;
-----------0--------2----------------------------------------------------return 0;

0 => KEYWORD, Total is: 6
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

                     TARGET TOKENS :
       TOKEN           TYPE                                                     LINE
```

Fig.4: Snap shot of Input & Filtration



```
Thank You!!

F:\clonecode>clonecode.exe
                   Welcome to Clone Code Detection
Enter source file name: testing/source1234.cpp
Enter target file name: testing/target1234.cpp
File type is Good.
Filtering files....
Filtering completed.

Now,
To continue...
Press [ s ] -> s

To continue...
Press [ t ] -> t

 Show Extracted Tokens [ Y ], TO Continue... [ N ]: y
                     SOURCE TOKENS :
       TOKEN           TYPE                                                     LINE
-------using--------0-----------------------------------------using namespace std;
---namespace--------0-----------------------------------------using namespace std;
---------std--------1-----------------------------------------using namespace std;
---------int--------0--------------------------int main() int number, reverse = 0;
--------main--------1--------------------------int main() int number, reverse = 0;
---------int--------0--------------------------int main() int number, reverse = 0;
------number--------1--------------------------int main() int number, reverse = 0;
-----reverse--------1--------------------------int main() int number, reverse = 0;
-----------0--------2--------------------------int main() int number, reverse = 0;
--------cout--------1---------cout<<"Input a Number to Reverse and press Enter: ";
---------cin--------1------------------------------------------------cin>> number;
------number--------1------------------------------------------------cin>> number;
---------for--------0----------------------------------------for( ; number!= 0 ; )
------number--------1----------------------------------------for( ; number!= 0 ; )
-----------0--------2----------------------------------------for( ; number!= 0 ; )
-----reverse--------1--------------------------------------reverse = reverse * 10;
-----reverse--------1--------------------------------------reverse = reverse * 10;
----------10--------2--------------------------------------reverse = reverse * 10;
-----reverse--------1------------------------------reverse = reverse + number% 10;
-----reverse--------1------------------------------reverse = reverse + number% 10;
------number--------1------------------------------reverse = reverse + number% 10;
----------10--------2------------------------------reverse = reverse + number% 10;
------number--------1------------------------------------------number = number/10;
------number--------1------------------------------------------number = number/10;
----------10--------2------------------------------------------number = number/10;
--------cout--------1-------------------cout<<"New Reversed Number is:  "<<reverse;
-----reverse--------1-------------------cout<<"New Reversed Number is:  "<<reverse;
------return--------0----------------------------------------------------return 0;
-----------0--------2----------------------------------------------------return 0;

0 -> KEYWORD, Total is: 6
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6
```

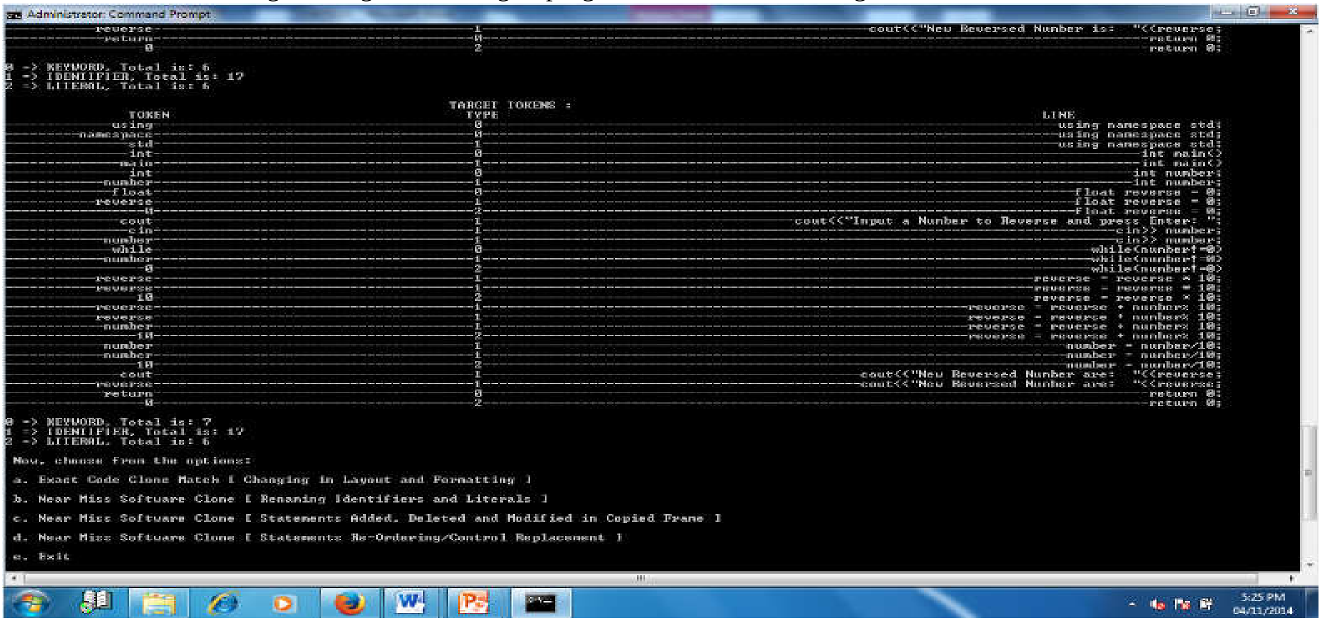
Fig.5: Snap shot of target & Filtration



```
 Now, choose from the options:

 a. Exact Code Clone Match [ Changing in Layout and Formatting ]

 b. Near Miss Software Clone [ Renaming Identifiers and Literals ]

 c. Near Miss Software Clone [ Statements Added, Deleted and Modified in Copied Frame ]

 d. Near Miss Software Clone [ Statements Re-Ordering/Control Replacement ]

 e. Exit

Enter your Choice: d

Near Miss Software Clone [ Statements Re-Ordering ]
----------------------------------------------------------------------int main()
---------------------------------------------------------------------int number;
--------------------------------------------------------------float reverse = 0;
----------------------------cout<<"Input a Number to Reverse and press Enter: ";
-------------------------------------------------------------------cin>> number;
----------------------------------------------------------------while(number!=0)
---------------------------------------------------------reverse = reverse * 10;
-------------------------------------------------reverse = reverse + number% 10;
-------------------------------------------------------------number = number/10;

Near Miss Software Clone [ Control Replacement ]
----------------------------------------------------------------while(number!=0)

Thank You!!

F:\clonecode>_
```

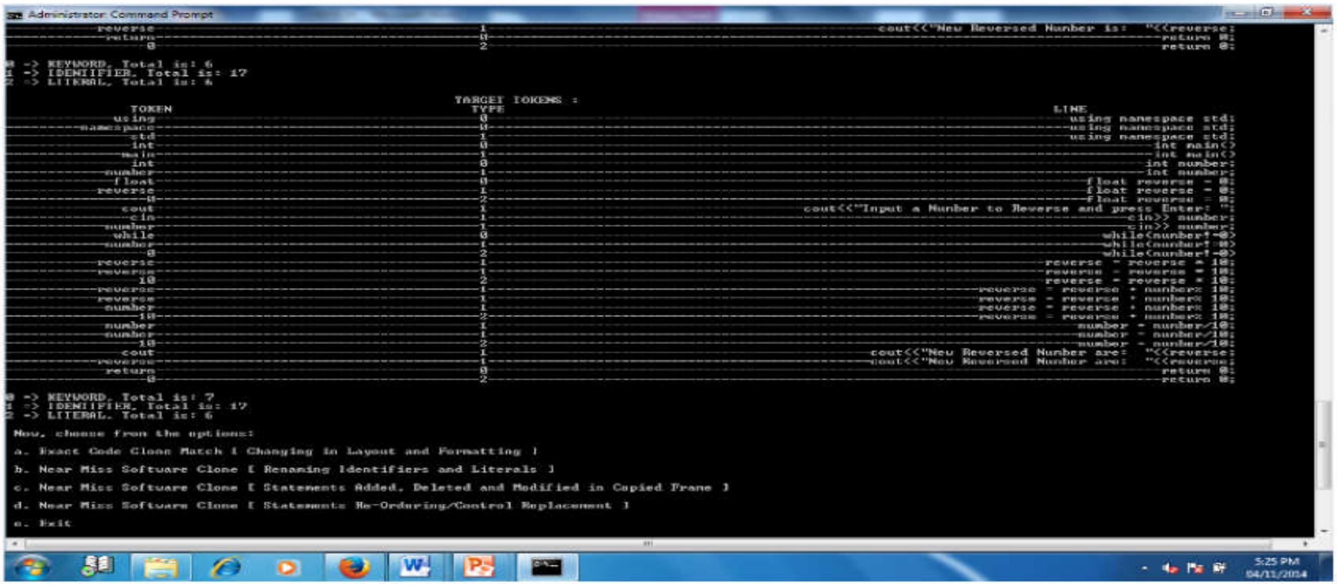
Figure 6. Snap shot of output & clone codes

In reorder cloning function or declaration of variables order is change with respect to source program file. In target Program order of statement is change but logical meaning of program statement is unchanged.



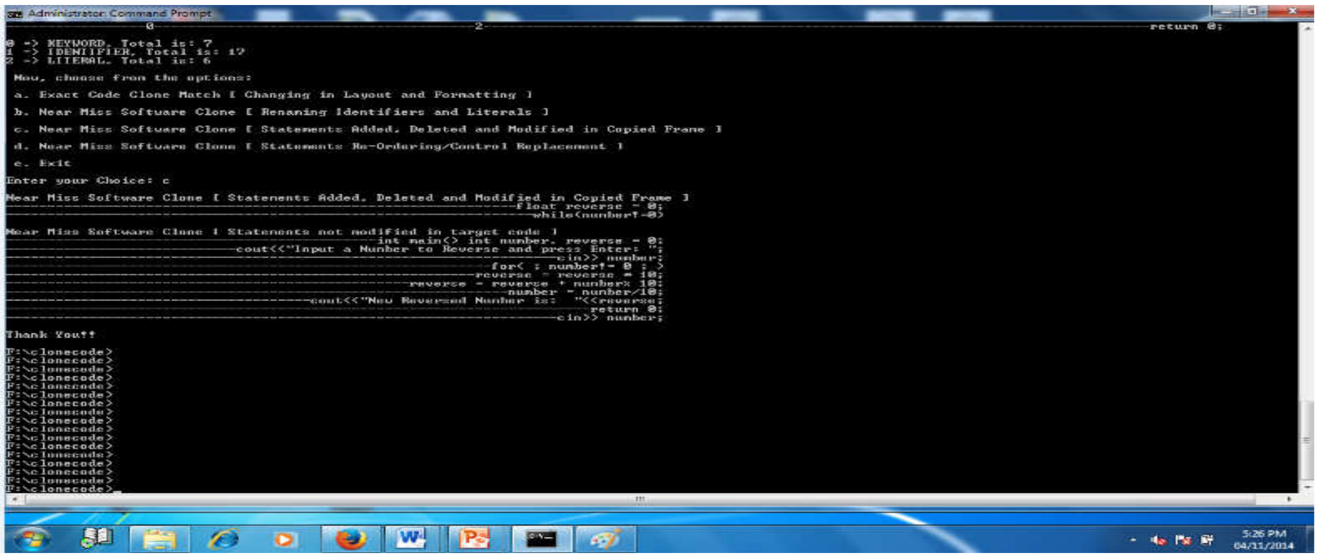
```
          reverse--------------------1------------cout<<"New Reversed Number is:  "<<reverse;
           return--------------------0-----------------------------------------------return 0;
                0--------------------2-----------------------------------------------return 0;

0 -> KEYWORD, Total is: 6
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

                            TARGET TOKENS :
            TOKEN               TYPE                                         LINE
            using-----------------0---------------------------------using namespace std;
        namespace-----------------0---------------------------------using namespace std;
              std-----------------1---------------------------------using namespace std;
              int-----------------0-------------------------------------------int main()
             main-----------------1-------------------------------------------int main()
              int-----------------0------------------------------------------int number;
           number-----------------1------------------------------------------int number;
            float-----------------0-------------------------------------float reverse = 0;
          reverse-----------------1-------------------------------------float reverse = 0;
                0-----------------2-------------------------------------float reverse = 0;
             cout-----------------1-----------cout<<"Input a Number to Reverse and press Enter: ";
              cin-----------------1----------------------------------------cin>> number;
           number-----------------1----------------------------------------cin>> number;
            while-----------------0--------------------------------------while(number!=0)
           number-----------------1--------------------------------------while(number!=0)
                0-----------------2--------------------------------------while(number!=0)
          reverse-----------------1-------------------------------reverse = reverse * 10;
          reverse-----------------1-------------------------------reverse = reverse * 10;
               10-----------------2-------------------------------reverse = reverse * 10;
          reverse-----------------1-----------------------reverse = reverse + number% 10;
          reverse-----------------1-----------------------reverse = reverse + number% 10;
           number-----------------1-----------------------reverse = reverse + number% 10;
               10-----------------2-----------------------reverse = reverse + number% 10;
           number-----------------1----------------------------------number = number/10;
           number-----------------1----------------------------------number = number/10;
               10-----------------2----------------------------------number = number/10;
             cout-----------------1--------------cout<<"New Reversed Number are:  "<<reverse;
          reverse-----------------1--------------cout<<"New Reversed Number are:  "<<reverse;
           return-----------------0--------------------------------------------return 0;
                0-----------------2--------------------------------------------return 0;

0 -> KEYWORD, Total is: 7
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

 Now, choose from the options:

 a. Exact Code Clone Match [ Changing in Layout and Formatting ]

 b. Near Miss Software Clone [ Renaming Identifiers and Literals ]

 c. Near Miss Software Clone [ Statements Added, Deleted and Modified in Copied Frame ]

 d. Near Miss Software Clone [ Statements Re-Ordering/Control Replacement ]

 e. Exit
```

Figure 7. Input file name & source program



```
          reverse--------------------1------------cout<<"New Reversed Number is:  "<<reverse;
           return--------------------0-----------------------------------------------return 0;
                0--------------------2-----------------------------------------------return 0;

0 -> KEYWORD, Total is: 6
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

                            TARGET TOKENS :
            TOKEN               TYPE                                         LINE
            using-----------------0---------------------------------using namespace std;
        namespace-----------------0---------------------------------using namespace std;
              std-----------------1---------------------------------using namespace std;
              int-----------------0-------------------------------------------int main()
             main-----------------1-------------------------------------------int main()
              int-----------------0------------------------------------------int number;
           number-----------------1------------------------------------------int number;
            float-----------------0-------------------------------------float reverse = 0;
          reverse-----------------1-------------------------------------float reverse = 0;
                0-----------------2-------------------------------------float reverse = 0;
             cout-----------------1-----------cout<<"Input a Number to Reverse and press Enter: ";
              cin-----------------1----------------------------------------cin>> number;
           number-----------------1----------------------------------------cin>> number;
            while-----------------0--------------------------------------while(number!=0)
           number-----------------1--------------------------------------while(number!=0)
                0-----------------2--------------------------------------while(number!=0)
          reverse-----------------1-------------------------------reverse = reverse * 10;
          reverse-----------------1-------------------------------reverse = reverse * 10;
               10-----------------2-------------------------------reverse = reverse * 10;
          reverse-----------------1-----------------------reverse = reverse + number% 10;
          reverse-----------------1-----------------------reverse = reverse + number% 10;
           number-----------------1-----------------------reverse = reverse + number% 10;
               10-----------------2-----------------------reverse = reverse + number% 10;
           number-----------------1----------------------------------number = number/10;
           number-----------------1----------------------------------number = number/10;
               10-----------------2----------------------------------number = number/10;
             cout-----------------1--------------cout<<"New Reversed Number are:  "<<reverse;
          reverse-----------------1--------------cout<<"New Reversed Number are:  "<<reverse;
           return-----------------0--------------------------------------------return 0;
                0-----------------2--------------------------------------------return 0;

0 -> KEYWORD, Total is: 7
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

 Now, choose from the options:

 a. Exact Code Clone Match [ Changing in Layout and Formatting ]

 b. Near Miss Software Clone [ Renaming Identifiers and Literals ]

 c. Near Miss Software Clone [ Statements Added, Deleted and Modified in Copied Frame ]

 d. Near Miss Software Clone [ Statements Re-Ordering/Control Replacement ]

 e. Exit
```

Figure 8. Snap shot of target program filtration for type-3clone code



```
                0--------------------2-----------------------------------------------return 0;

0 -> KEYWORD, Total is: 7
1 -> IDENTIFIER, Total is: 17
2 -> LITERAL, Total is: 6

 Now, choose from the options:

 a. Exact Code Clone Match [ Changing in Layout and Formatting ]

 b. Near Miss Software Clone [ Renaming Identifiers and Literals ]

 c. Near Miss Software Clone [ Statements Added, Deleted and Modified in Copied Frame ]

 d. Near Miss Software Clone [ Statements Re-Ordering/Control Replacement ]

 e. Exit

Enter your Choice: c

Near Miss Software Clone [ Statements Added, Deleted and Modified in Copied Frame ]
----------------------------------------------------------------float reverse = 0;
------------------------------------------------------------------while(number!=0)

Near Miss Software Clone [ Statements not modified in target code ]
-----------------------------------------------int main() int number, reverse = 0;
--------------------------------cout<<"Input a Number to Reverse and press Enter: ";
-----------------------------------------------------------------------cin>> number;
----------------------------------------------------------------for( ; number!= 0 ; )
-------------------------------------------------------------reverse = reverse * 10;
------------------------------------------------------reverse = reverse + number% 10;
-----------------------------------------------------------------number = number/10;
---------------------------------------------cout<<"New Reversed Number is:  "<<reverse;
---------------------------------------------------------------------------return 0;
-----------------------------------------------------------------------cin>> number;

Thank You!!

F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
F:\clonecode>
```

Figure 9. Compare source and target program, detect type-3clone codes

This new approach programs, clone search using filtration and visual detection. This approach is also useful for server side program to finding cloning. It is like open system software to apply different web application. It is solve the problem of long code, which is used as copy-past and find-out easily. It is automatic filter based detection method. Traditional method is useful for limited type of program but this method is use for all server side program.

REFERENCES

1. D.Gayathri Devi and Dr. M Punithavalli "DETECTING SOFTWARE CLONES USING ASSOCIATION RULE MINING "Volume 3, Issue 1, Jan. 2013.
2. Shaheen Khatoon and Azhar Mahmood "An Evaluation of Source Code Mining Techniques" Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2011, Page No.234-246.
3. Chanchal Kumar Roy and James R. Cordy ,"ClonesA Survey on Software Clone Detection Research", September 26, 2007, PageNo.1-115.
4. Christopher Forbes, Iman Keivanloo, Juergen Rilling "Doppel-Code: A Clone Visualization Tool for Prioritizing Global and Local Clone", IEEE 36th International Conference on Computer Software and Applications, 2012, PageNo.366-368.
5. Rainer Koschke, "Large-Scale Inter-System Clone Detection Using Suffix Trees", 16th European Conference on Software Maintenance and Reengineering, 2012, pp. 309-318.
6. Yuehua Zhang, Ying Liu, Lingling Zhang and Yong Shi," A Data Mining Based Method Detecting Software Defects in Source Code" in ICSM '98: Proceedings of the International Conference on Software Maintenance, 1998, Page No. 368–377.
7. Alexander Breckel "Error Mining: Bug Detection through Comparison with Large Code Databases," MSR 2012, Zurich, Switzerland, page No.175-178.
8. Salwa K. Abd-El-Hafiz "A Metrics-Based Data Mining Approach for Software Clone Detection", 36 th International Conference on Computer Software and Applications, 2012, PageNo.35-42.
9. YOSHIHITO HIGO AND SHINJI KUSUMOTO, "HOW OFTEN DO UNINTENDED INCONSISTENCIES HAPPEN? DERIVING MODIFICATION PATTERNS AND DETECTING OVERLOOKED CODE FRAGMENTS," 2012 28TH IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE (ICSM)
10. S. BELLON, R. KOSCHKE, G. ANTONIOL, J. KRINKE AND E. MERLO, "COMPARISON AND EVALUATION OF CLONE DETECTION TOOLS," IEEE TSE, VOL. 33, NO. 9, 2007, PP. 577-591.
11. FILIP VAN RYSSELBERGHE, SERGE DEMEYER "EVALUATING CLONE DETECTION TECHNIQUES", 2010, PAGENO.1-12.
12. Chanchal Roy "A Mutation / Injection-based Automatic Framework for Evaluating Code Clone Detection Tools",The 9th CREST Open Workshop.
13. Aaron Bloomfield "Scanning", 2005, ppt 1-32.
14. Wu Zhifei ,Wang Tie, Zhang Qinghua, GaoTingyu, Li Hongfang,"Research on Generating Detector Algorithm in Fault Detection" page 7-8.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.