International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-7)

Research Article

July 2015

Code Clone Detection Using Function Based Similarities and Metrics

Manpreet Kaur, Madan Lal
Department of Computer Engineering, Punjabi University,
Patiala, Punjab, India

Abstract—

Code cloning is a process of coping and pasting of code fragment with or without minor alteration like renaming, addition and deletion to the code fragments in the software. The copied and pasted code fragments are known as code clones. Code cloning reduces the time and effort of the software developer but it also decreases the quality of the software like readability, changeability and increases maintainability. So, code clone has to be detected to reduce the cost of maintenance to some extent. In this paper, a new hybrid technique is purposed to detect code clone from the source codes by segmenting the code into number of functions. The proposed technique can detect type 1, type 2 and type 3 clones efficiently.

Keywords— Software, Code clone, hybrid technique, potential clone.

I. INTRODUCTION

During the software development process, the software engineer copy one piece of fragment and paste with or without alteration like renaming, addition or deletion etc to the fragment to reduce time and efforts. The process of coping and pasting of code fragment is known as code cloning and the copied pasted code is known as code clone. The reusing code is common practice in modern software developing, but it has some drawback [14]. It raises the maintenance cost [16] while decreasing quality like changeability and updating of the software system [1]. It also increase the chances of the bugs in the software system, because bug present in one fragment can be copied and pasted various time may increase the bugs in the software system [11] [8].

According to literature survey, about 7% to 23% of code in the software system is copied and pasted code i.e. cloned code, [4] [2] and around 66% of cloned code is modified clone code i.e. code copied and pasted with some alteration like renaming, addition or deletion of the statement to the code[10]. The reason of existence of code clone is software developer himself [10]. The software developer adopts the code cloning process intentionally or unintentionally. Sometime the code is intentionally introduced by software developer by reusing the code fragment with or without alteration, while code can be unintentionally injected by the software developer because of comply of some well known solution or best practices [8].The code clone detection techniques should provide precise and effective information about clones because chances of detecting large number of clones in the large software system are very high [6].

There are two types of similarities exist in the code clones one is syntactic similarities and other is semantic similarities. If the text of the clone code matches then it syntactic similarity and if the function or implementation of the code clone matches then it is semantic similarity. Code clone are of four types described below.

Type 1 – These are identical clones with syntactic similarities, in this type of clone only variation is allowed in whitespaces and comments.

Type 2 – These are renaming clones with syntactic similarities, in this type of clone only variation is allowed in literal, identifier, type, whitespaces and comments.

Type 3 – These are modified clones with syntactic similarities, in this type of clone variation is allowed to rename literal, identifier and addition or deletion of statement to code.

Type 4 – These are semantic clone with semantic similarities, these clones are semantically same but syntactically different i.e. computation is same but implement by different syntactic variants.

II. CLONE DETECTION APPROACHES

Text Based Approach- This approach is most basic approach. There is no need to transform the source code before applying the comparison [18]. In this approach, code clone is detected by comparing code line by line in the form of string [4].This approach is easy to implement but detect only type 1 clone from the source code

Token Based Approach-This approach is applied only on transformed code i.e. before applying this approach the source code must transform into tokens by using parser or lexer [15]. In this approach, comparison takes place line by line in the form of tokens [4]. This approach can detect type 1 and type 2 clones only [15].

Abstract Syntax Tree Based Approach- This approach is also applied on transformed source code, in this approach first source code is transformed into abstract syntax tree through parsing [1] [8]. In this approach comparison takes place by comparing sub tree of abstract syntax tree. This approach can detect type 1, type 2 and type 3 clones only. This approach is difficult to implement as compare to above approaches because convert source code into abstract syntax tree is difficult task.

Program Dependency Graph(PDG) Based Approach-- This approach is also applied on transformed source code, in this approach first source code is transformed into PDG. PDG is a directed graph which represents the dependencies between program elements. In PDG nodes represents program elements and PDG edges represents dependencies. There are two types of dependencies exist, control dependencies and data dependencies [6]. In this approach comparison takes place by comparing sub graph of PDG. This approach can detect all type of code clone, but it very difficult to implement because conversion of source code into PDG is very complicated task.

Metric Based Approach- There is no need to transform source code into any other form, it is directly applied on the source code [6]. In this approach, metric is calculated from the source code like number of lines, number of function calls, number of branches, number of comments etc and the metrics of two source code is compared to detect code clones [10]. It is easy to implement but it cannot find all probable code clones from the source code.

Hybrid approach is the combination of two or more approaches or techniques. This approach can enhance the advantages of code clone detection techniques while decreasing its disadvantages[10].

III. PROPOSED WORK AND IMPLEMENTATION

The proposed work is based on hybrid approach. Steps for the proposed tool are described below.

Step 1: Load two files containing source programs.

Step 2: Divide programs into number of functions.

Step 3: Calculate metrics of the both files like number of lines, number of function calls, number of for loops, number of while loops etc.

Step 4: Compare metrics to detect potential code clones.

Step 5: Perform template conversion on potential code clones.

Step 6: Perform text based comparison. If the number of matched lines exceeds the threshold value then it is actual code clones.

Proposed technique focuses on to segment the source code into number of function or methods. Metrics are calculated like lines of code, number of for loops number of while loops and number of function calls in a function. The metrics of the two source code is compared to detect potential code clone in the source code. Then parser is applied to the potential clone for template conversion. This parser works same way as the other parser present in the compiler only difference is it does not tokenize the code, it just convert all constant values to word constant and for loop to iterative loop, etc.

Example:

```
For(i=o;i>10;i++)
{
Int a=0;
Int b=0;
b=a+10;
}
```

When parser is applied on above example then output will be as below.

Loop(var = num; var>num: var++)

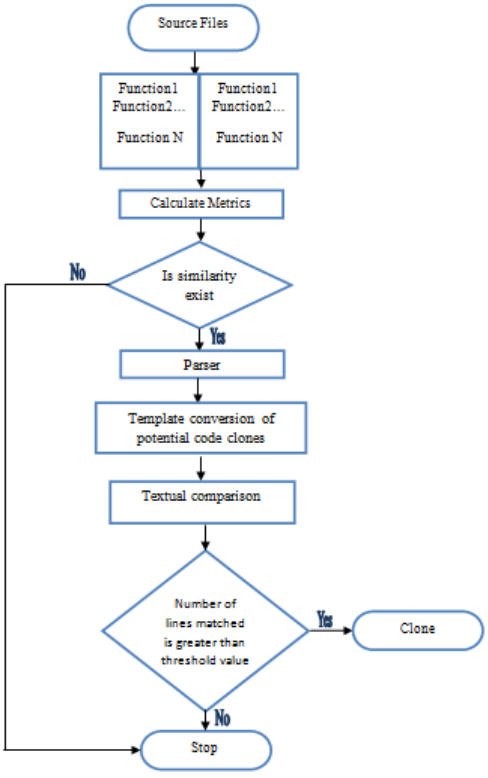

Fig 1 Flow diagram


```
{
Datatype var = num;
Datatype var = num;
Var=var+num;
}
```

After the template conversion, textual comparison is done. If the number of line matched crosses the threshold value or percentage value then the code said to be a code clone. By this technique type 1 type 2 and type 3 code clones can be detected. The flow diagram of the proposed technique is shown below.

The working of proposed tool start with manual selection of two input source files. To choose files, OpenFileDialog tool is used. Only .Cpp files are used as input. Input files are selected with the help of button tools. Button tools handle the file chooser event and display the absolute path of .Cpp file on text box and the content of the source code to the another textbox. After browsing the second button, code detection process is initiated and the detected clone is displayed in the clone text box as shown in Figure 3.

When "browse File 1" or "Browse File 2" is clicked then open dialogue box is appeared to choose .Cpp files to find clones in system as shown in Fig 2. After selecting file, the absolute path is shown in the text box and the content of the source file is displayed on another textbox shown in Fig 3.

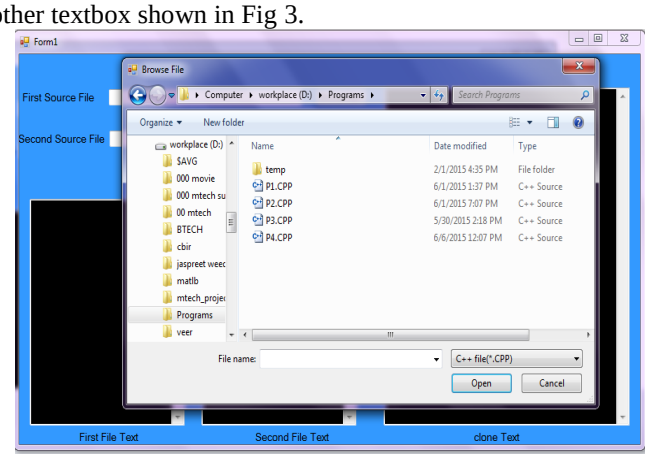

Fig 2 File selection window for first file

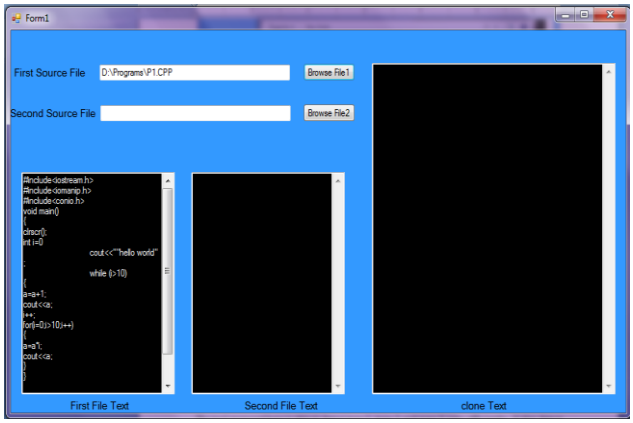

Fig 3 Screen after the selection of first file

After clicking the browse file 2, clone detection processes initiated and the detected clone is displayed in to the clone textbox, shown in Fig 4

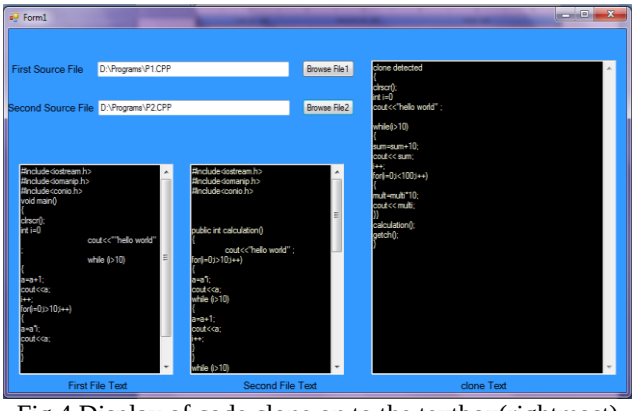

Fig 4 Display of code clone on to the textbox(rightmost)

IV. RESULT AND DISCUSSION

The method proposed here is able to detect type 1, type 2 and type 3 very efficiently. To develop proposed tool, Visual studio 2010, version 10 and .Net framework version 4.0 is used. To provide interactive interface for user, window application form is used as shown in Figure 4. Here potential code clones is detected by metric approach by calculating metric like number of lines, number of function calls, number of loops etc. After the detection of potential clone text based comparison is performed, if the number of matched line exceeds the threshold value then its actual code clone.

V. CONCLUSIONS AND FUTURE SCOPE

The proposed method is a hybrid approach which combines metrics based technique and text based technique. In this technique first metric is applied to detect potential code clone present in the code, then template conversion is done, after the template conversion textual comparison is performed. After textual comparison, if the number of matched lines exceeds the threshold value, then it is actual code clone. The proposed tool can detect type 1, type 2 and type 3 very efficiently. This tool can be further enhanced by using clone removal techniques after detecting actual code clones. Efficiency of this technique can be further improved for detection of type 4 clones by using more metrics.

REFERENCES

[1] Rajkumar Tekchandani, Rajesh Kumar Bhatia and Maninder Singh, "Semantic Code Clone Detection Using Parse Trees and Grammar Recovery", pp.41-46, IEEE, 2013.
[2] Amandeep Kaur and 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",Page 16-21, 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", Page 3-12, IEEE, 2011.
[7] Mai Iwamoto, Shunsuke Oshima, Takuo Nakashima, "Token-based Code Clone Detection Technique in a Student's Programming Exercise", Page 650-655,,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] Mikkel Jonsson Thomsen, Fritz Henglein,"Clone Detection using Rolling Hashing, Suffix Trees and Dagnification:A Case Study."IEEE,2012.
[10] Priyanka Batta, Miss Himanshi, " Hybrid Technique for software code clone detection" IJCT, Volume 2 no. 2 April 2012.
[11] Kiran preet, Sushil Garg, "Detection and measuring similarity in code clone using ripley's function Approach.", IJAST,Volume 2,issue 4,dec 2014.
[12] Rubala Sivakumar,Kodhai.E, "Code clone detection in website using approach ", IJCA, Volume 48-No. 13,June 2012.
[13] Balwinder Kumar, Dr. Satwinder Singh, " Code clone detection and Analysis using Software Metrics and Neural Network- A Literature Review ",IJCST, ,Volume 3,issue 2, mar-apr 2015.
[14] prajila Prem, "A Review on code clone analsis and code clone detection ", IJEIT , Volume 2, issue 12, june 2013.
[15] Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue," CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code" IEEE, Volume 28, No. 7, July 2002.
[16] Robin Sharma," Hybrid approach for efficient software clone detection", ESTIJ, ISSN:2250-3498, Volume 3-No. 2, April 2013
[17] Egambraram Kodhai and Selvaduri Kanmani, " Method level code clone detection through LWH(Light Weight Hybrid) approach" JSERD,2014.
[18] Kanika Raheja, Rajkumar Tekchandani," Detection of code clones using datasets",IJARCSSE, Volume 2, Issue 7, July 2012.
[19] Amandeep Kaur, Mandeep Singh Sandhu, "Software code clone detection model using hybrid approach ", ijct, Volume 3 No. 2, 2012.
[20] http://www.wikipedia.org/wiki/Abstract_syntax_tree

