International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887*
*Volume 6 Issue III, March 2018- Available at www.ijraset.com*

# Code Clone Detection using Generic Approach

Dr.Amita Goel[1], Vishal Vats[2]
[1]*Associate Professor in Department of Information Technology, Maharaja Agrasen Institute of Technology)*
[2]*Department of Information Technology Maharaja Agrasen Institute of Technology Sector-22, Rohini, New Delhi-110086, India*

***Abstract: Of late, the process of development of software has become very tiresome and time consuming. In order to make the development fast and easy, developers tend to use a pre-existing code with or without altering few lines of the code. This reuse of code with or without some modification is termed as code clone. Cloning of code has become common problem in development of software which makes software maintenance really difficult and expensive. The presence of code clones in a source code leads to inconsistency. In order to stay ahead of these problems, it is very important to detect the presence of code clones in software. Code cloning reduces the time and effort of the software developer but it also decreases the quality of the software and increases maintainability. Keeping in mind the importance of code clone detection, various techniques has been proposed for detection of code clones such as detection based on textual approach, detection based on tokens, detection based on abstract syntax tree, detection based on program dependency graph and detection based on metrics. In this paper, I purpose detection based on generic approach.***

***Keywords: SoftwareMaintainance, Software Development, Code Clone.***

## I. INTRODUCTION

In software development, it has been seen that it is common to use a pre-existing code with or without some modification in few lines of code to save time and effort. This reuse of pre-existing code with or without modification is termed as Code Clone. Code clone has no single or generic definition, each researcher had its own definition.[8] Data from previous work suggests that a considerable fraction (between 7% to 23%) of the code in a typical software system has been cloned [9] i.e. developer copying and pasting the code with or without making the changes in the code. Code clones occur in a source code when a developer uses a existing code in a new way by copying it and using it with or without any modification in functionality. There can be various reasons for presence of clone in a code. The main reason why developer tries to copy-paste code is that it saves a lot of time in development and otherwise it is really hectic to start a code from the scratch. Other reason could be like similar coding style. The developer might see some superficial advantages of using existing code but it becomes a disadvantage when it comes to software maintainance and overall cost. As the code is copied without alternations , there is a chance of increase in bugs in the software, because errors present in one module can increase the errors in another when they are linked with modules, increasing bugs in a software
system[1][10]. Code cloning is found to be a more somber and serious problem in industrial software systems . In presence of clones, the normal operation of the system may not be affected, but if maintenance teams do not take measures to counter the problem, further development may become extremely expensive. Clones are supposed to have a negative impact on advancement and evolution[6] **.**That's where there is need for identification of code clones. Identification of clones is a process of identifying similar part of code in a source code. There has been various code clone detection techniques for different types of code clones. In this paper, I have mentioned about various types of code clones( Type I, Type II, Type III, Type IV) and various techniques for their detection such as detection based on textual approach, detection based on tokens, detection based on abstract syntax tree, detection based on program dependency graph and detection based on metrics.

## II. BASICS OF CLONE

Before diving into the various code clone detection techniques, one should be well acquainted with few terminologies:

### *A. Code Fragments (CF)*

A code fragment refers to any sequence of code lines (with or without comments). It can be any granularity, such as, function definition, begin-end block, or sequence of statements [8][7].

### *B. Code Clone*

A code fragment let's say CF1 is a clone of another code fragment let's say CF0 if they are similar syntactically or semantically. By similar it means f(CF0) = f(CF1) where f is a similarity function.

### C. *Clone Pair*

Two code fragments which are similar to each other form a clone pair (CF0; CF1). When multiple fragments are similar to each other, they form a clone class or clone group [7].

### D. *Clone Types*

Mainly there are two kinds of similarities between code fragments. First is similarity based on program text i.e. Type I, Type II, Type III and Second is similarity based on functionality i.e. Type IV.

1) *TYPE 1:* are similar code fragments except for variations in whitespaces, layout and comments. They are known as exact clones.
2) *TYPE II:* are syntactically identical fragments except for variations in identifiers, literals, types, whitespaces, layout and comments. They are known as renamed clones.
3) *TYPE III:* are copied code fragments with some modifications such as addition or deletion of few statements, comments and whitespaces. They are known as near miss clones.
4) *Type IV:* are two or more code fragments that are semantically similar but are syntactically different. They are known as semantic clones.

## III. CODE CLONE DETECTION TECHNIQUES

### A. *There are different code clone detection Techniques Which detect Various types of Clones presEnt in a Source Code[8]:*

1) Textual Approach
2) Token Based Approach
3) Abstract Tree Based Approach
4) Program Dependency Graph Approach
5) Metrics Based Approach

### B. *Textual Approach*

In textual approach or text-based approach, line by line comparison is done between two code fragments in order to find textual similarity between both the code fragments. This technique does not require any filtration or normalization process and can be directly applied on the code. Therefore, this approach can detect exact clones with little to no variation in layout or comment [5]. This technique is basically all about string based matching of two code fragments and detecting whether those code fragments are similar or clone to each other. This technique loses its credibility when there is any change in variable name or syntactical change in the code fragment. So this approach is not highly efficient and can only detect Type I clones. This approach has a complexity of O(n) where n is lines of code. It is easy to implement.

### C. *Token Based Approach*

In token based approach lexical analyzer is used. This technique uses a more sophisticated transformation algorithm by constructing token sequence from the source code using a lexer. The sequence is then scanned for duplicated subsequence of tokens and the corresponding original code is returned as clones. Lexical approaches are generally more effective over minor code changes such as formatting, spacing, and renaming than textual techniques [9][ 2]. This approach can be used to detect Type I and Type II clones. One can also detect Type III clones using this approach but that would require concatenation of Type I and Type II clones [9]. It has a complexity of O(n) where n is the number of token sequences.

### D. *Abstract Syntax Tree Based Approach*

This approach uses parser to convert the source code into abstract syntax tree. This approach creates sub trees rather than constructing tokens for the source code and then pattern matching is applied on them in order to find similar sub trees which are considered as code clones. This approach follows even more sophisticated algorithm where variable names and constants can be abstracted in tree creation [5]. This approach is capable of finding out the clones in which there has been some addition or deletion of statements, that is, Type III clones or near miss clones. This technique requires advanced level of algorithm which in turns increases its complexity but still it is better than text based or token based techniques[11]. It has a complexity of O (n) where n is number of nodes of AST. Implementation of the ASTB is very tedious as conversion of source code into tree is difficult.

### *E. Program Dependency Graph Approach*

Program Dependency graph approach converts the source code into a Program Dependency Graph (PDG). A PDG is directed graph representing dependencies between program elements. There are two types of dependencies in a PDG, namely, control dependency and data dependency [3]. Because of presence of these dependencies PDG carries semantic information. Once PDG is obtained, isometric sub graph matching algorithm is used to find similar sub graphs and the similar sub graphs are taken as code clones [1]. This approach goes one step further in obtaining a source code representation with high abstraction than other approaches because it considers the semantic information of the source [4]. This technique is capable of detecting Semantic and syntactic clones in a source. It has complexity of O ($n^3$) where n is node of PDG.

### *F. Metrics Based Approach*

Metrics based approach is an advanced technique which is used to find all four types of code clones. This technique does not compare code directly instead source code is first converted into AST or PDG for calculation of metrics like effective lines of code, number of classes, number of methods, number of different loops, Number of variables and many others[2]. The calculation of metrics is done by tools like Columbus. The code fragments having similar metrics values are declared as code clones. This technique is capable of detecting Type I, Type II, Type III and Type IV code clones.

## IV. PROPOSED APPROACH

Generic approach is a general approach for finding multiple types of clones present in a Java file. This approach is simple as well as effective.

Steps for the proposed tool are described below.

*A. Step 1:* Open two files which has to be checked as source files
*B. Step 2:* Check similarity between two files using word based comparison. If both files are same then notify user.
*C. Step 2:* Modulate the files into smaller sub-programs based on functions in the program
*D. Step 3:* The functions from each file are converted to mycode by variable separation and replacing operands.
*E. Step 4***:** The functions from one file are compared with functions of other file using string matching algorithm i.e. Knuth Morris Pratt.
*F. Step 5:* Percentage of presence of clone is shown.
*G. Step 6:* User can delete the clone from the file and can save it as a new file.

The proposed method mainly aims on checking the number of functions in the files and comparing them using string matching algorithm. The result of the project can be done manually by selecting two input files. An open dialog tool is used for selecting the input files. Any two similar java file can be compared. After including the two files, click on the button for finding potential clones.

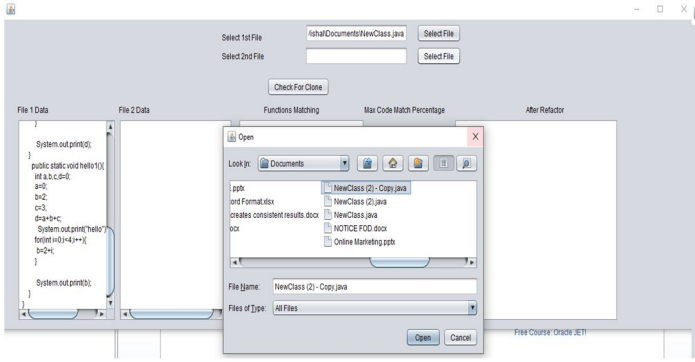


Fig 1: Selecting input files

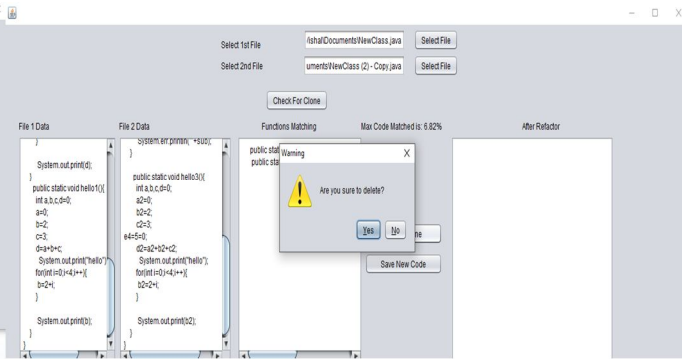


Fig 2: Deleting the clone from file

## V. CONCLUSION AND FUTURE SCOPE

The proposed approach is generic approach which is able to identify Type1, Type2 and Type 3 clones. This is developed in NetBeans using Java. It provides a user friendly graphic user interface as shown in Fig1. This tool is based on the concept of extracting functions from the files and comparing them using string matching. In same of same files, the user is notified about the similarity. This tool can be further enhanced in terms of efficiency and complexity. Code Clone Detection and its removal is an active research area and work has been carried out on the same.

## VI. ACKNOWLEDGEMENT

I would like to thank Dr. Amita Goel for her immense help, support, useful discussions and valuable recommendations throughout the development of this paper.

## REFERENCES

[1] Kiran preet, Sushil Garg, “Detection and measuring similarity in code clone using ripley’s function Approach.”, IJAST,Volume 2, 2014
[2] Sushma, Jai Bhagwan,"A Novel Metrics Based Technique for Code Clone Detection”, Guru Jambheshwar University of Science & Technology,IJECS, Volume 05 Issue 9 Sep., 2016
[3] Yoshiki Higo, Yasushi Ueda, Minoru Nishino, Shinji Kusumoto,"Incremental Code Clone Detection: A PDG-based Approach",Graduate School of Information Science and Technology, Osaka University
[4] Sunayna, Kamna Solanki, Sandeep Dalal, Sudhir,"Comprehensive Study of Software Clone Detection"M.D. University, IOSR Journal of Computer Engineering (IOSR-JCE),Vol 18, Issue 4, Ver. II (Jul.Aug. 2016)
[5] Surbhi Sonika, Rajkumar Tekchandani,"Hybrid Approach for Code Clone Detection"Thapar University, Volume 2 Issue 5, Jul-Aug 2014
[6] Shahid Ahmad Wani, Shilpa Dang,"A Comparative Study of Clone Detection Tools",International Journal of Advance Research in Computer Science and Management Studies,Vol.3, Issue 1, Jan 15
[7] Prajila Prem,"A Review on Code Clone Analysis and Code Clone Detection",International Journal of Engineering and Innovative Technology (IJEIT)Volume 2, Issue 12, June 2013
[8] Dr.Amita Goel, Vishal Vats,“Comparison and Evaluation of Code Clone Detection Techniques “ Maharaja Agrasen Institute of Technology,IJRASER, Volume 5 Issue XII (December 2017)
[9] Harpreet Kaur, Rupinder kaur,"A Review: Clone Detection in Web Application Using Clone Metrics",Yadavindra College of Engineering (YCOE), Volume 2 Issue 4, Jul-Aug 2014
[10] Chanchal K. Roy, James R. Cordy , Rainer Koschke, “Evaluation of code clone detection tools: A qualitative approach”, School of Computing, Queen's University, Canada and University of Bremen, Germany, March 2009
[11] Tahira Khatoon, Priyansha Singh, Shikha Shukla”Abstract Syntax Tree Based Clone Detection for Java Projects” IOSR Journal of Engineering,Volume 2,Issue12,Dec 2012