iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** XII **Month of publication:** December 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XII December 2017- Available at www.ijraset.com

Comparison And Evaluation of Code Clone Detection Techniques

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: Nowadays, the process of development of software is very tiresome and time consuming. In order to make the process easy, developer sometimes uses a pre-existing code with or without changing a few part of the code. This reuse of code with or without some modification is termed as code clone. Cloning of code is a 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 get rid of these problems, it is imperative to detect the presence of code clones in software. 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 am presenting comparison and evaluation of different available code clone detection techniques.

Keywords: Code Clone, Software Maintainance

I. INTRODUCTION

In software development it has been seen that it is accustomed to use a pre-existing code with or without some modification in few part of code. 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.[2] Data from previous work suggests that a considerable fraction (between 7% to 23%) of the code in a typical software system has been cloned [1].

Code clones happen 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 try 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. 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[4] .

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. BACKGROUND

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 [2][3].

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. By similar it means f(CF0) = f(CF1) where f is a similarity function.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XII December 2017- Available at www.ijraset.com

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 forms a clone class or clone group[3].

D. Clone Types

Mainly there are two kinds of similarities between code fragments. First, similarity of program text. i.e. Type I, Type II, Type III Second, 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 lines. They are known as near miss clones.
4) *Type Iv:* are two or more code fragments that perform similar functionality 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

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 validity 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.

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 robust over minor code changes such as formatting, spacing, and renaming than textual techniques [1, 2]. This approach can be 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. 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. 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. It has a complexity of O (n) where n is number of nodes of AST.

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 [7]. 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 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 [6]. 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[8]. 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.

Properties	Textual	Token Based	AST Based	PDG Based	Metrics Based
Transformation	Remove white spaces and comments	Tokens from source code	AST from source code	PDG from source code	Find metrics value from AST or PDG generated from source code
Comparison based on	Lines of code	Token	Node of AST	Node of PDG	Metrics value
Detect Clone Type	Type I	Type I,II	Type I,II,II`	Type I,II,III	Type I,II,III.IV
Complexity	O(n) n is lines of code	O(n) n is no. of tokens	O(n) n is nodes of AST	$O(n^3)$ n is nodes of PDG	O(n) n is metrics value
Accuracy	High	Low	High	High	High

Table1: Comparison of code clone detection techniques

IV. COMPARISON OF DIFFERENT CODE CLONE

A. Detection Techniques

There are various code clone detection techniques and therefore it is necessary to summarize the comparison in terms of few properties. Properties used for comparison are listed below:

1) *Transformation:* Transformation refers to using source code and converting it in a form required for code clone detection depending on the technique used.
2) *Comparison based on:* This property tells about the basis of comparison in a technique.
3) *Detect Clone Type:* This property tells about the clone types which a technique is capable of detecting.
4) *Complexity:* Complexity of a technique tells about how fast or slow a technique works.
5) *Accuracy:* Accuracy tells about how accurate a technique can detect the clone types it is capable of detecting.

In Table 1, it is shown that how one technique is different from another on the basis of properties mentioned above.

V. CONCLUSION

Code clone detection is an active research topic and presence of code clones in software is a critical and active problem of software development industry. It is generally happening to reduce to software development time and effort. It helps in decreasing the software development but when it comes to software maintenance it turns into a huge problem. Due to presence of these code clones, software maintenance has become really difficult.

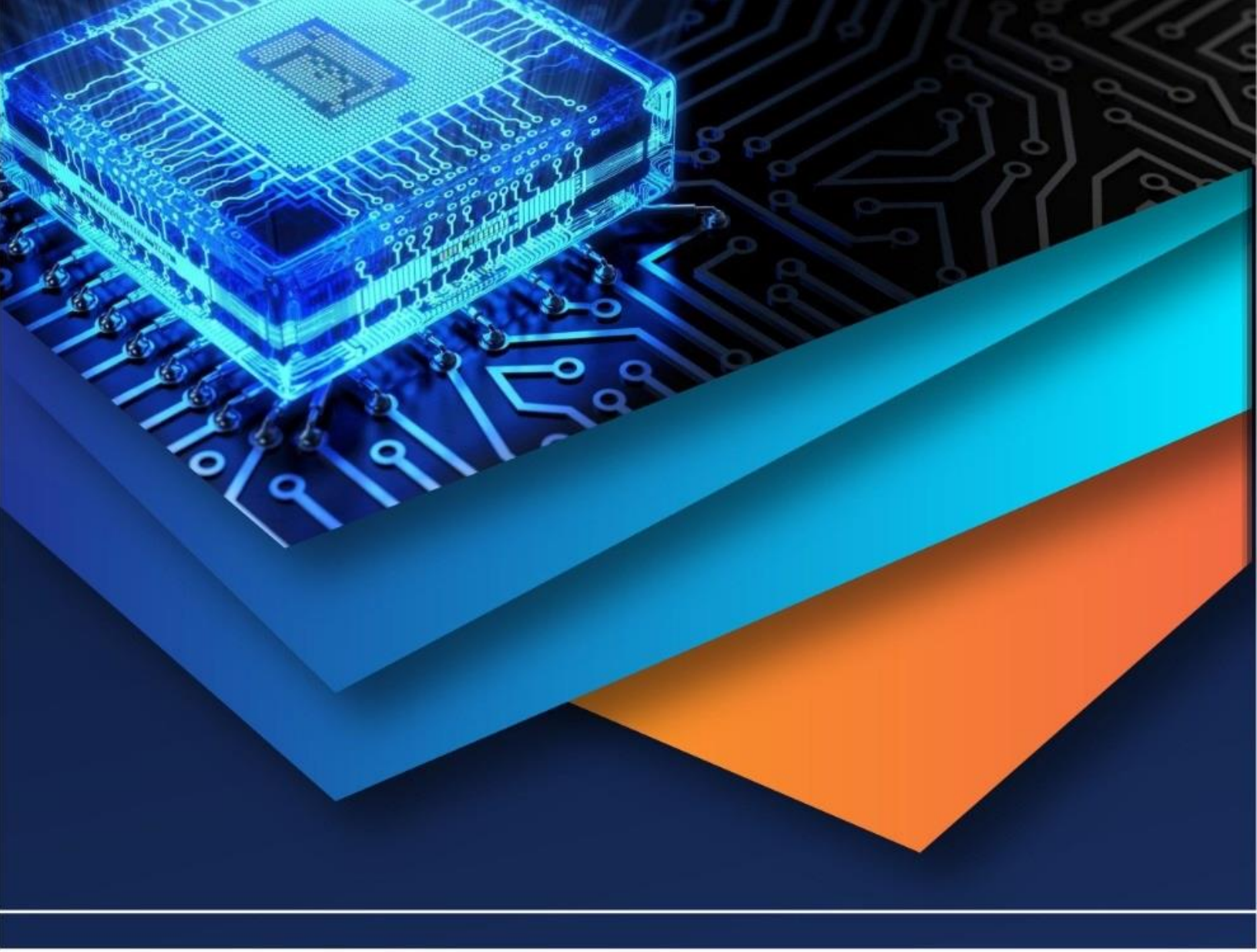
In this paper, I have focussed on various code clone detection techniques and their comparison based on few properties. From this paper, it can be concluded that Metric based technique is the most appropriate and accurate technique for code clone detection as it is able to detect syntactic as well as semantic error with higher accuracy and lesser complexity.

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] 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
[2] 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
[3] 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
[4] 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
[5] Surbhi Sonika, Rajkumar Tekchandani,"Hybrid Approach for Code Clone Detection"Thapar University, Volume 2 Issue 5, Jul-Aug 2014
[6] 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)
[7] 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
[8] 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



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY