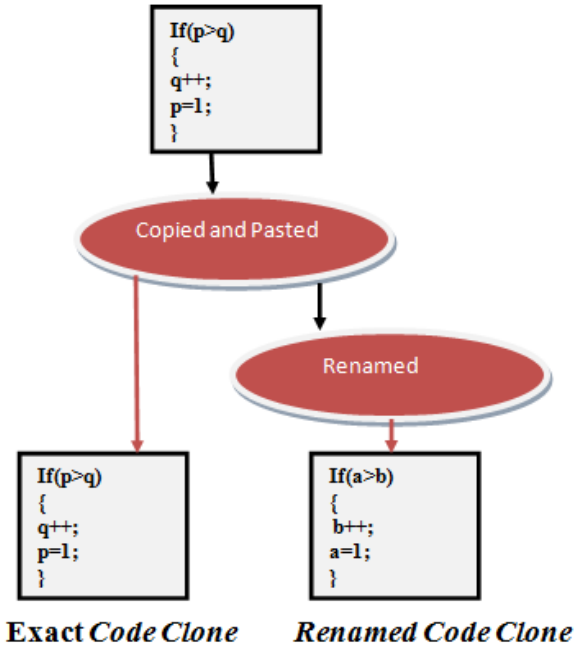
Code Clone Detection Using Various Approaches

Simranjeet Kaur[1], Er. Rupinder Singh[2]
[1]*Computer Science Department, Chandigarh University, Gharuan, Punjab*
[2]*Assistant Professor, Computer Science Department, Chandigarh University, Gharuan, Punjab*

Abstract **-** In the last few decades many techniques for software clone detection have been investigated by various researchers to detect the duplicated code in programs but all of these techniques have different merits and demerits. However, after a decade of this research there has been a lack of progress in understanding where to fit these techniques into the maintenance process and to detect the evolution of software clones. Code clones are basically identical fragments of code that occur at various locations in a program source code. There is great necessity to understand various approaches of clone detection as it provides useful information for the maintenance, reengineering, program understanding and reuse. After comparison of text based, token based and tree based approach it has been analyzed tree based approach is very fast to detect the efficient clones and one of the major area in which code can be semantically and syntax wise checked.

I. INTRODUCTION

Copying of code fragments and either reuse the same code by pasting it in different portions of source code or pasting it with minor modifications .Software cloning is a perception in which source code is duplicated. This type of reuse approach of existing code is called cloning and pasted code fragment (with or without modifications) is called a clone of the original. Code clones do not occur in software systems by themselves but occurs due to different reuse and programming approach. Code cloning is found to be more serious problem in software development activities and processes as cloning increases the probability of inconsistencies in update. Clones increased the probability of bug propagation, introducing a new bug, bad design, difficulty in system improvement/modification increased maintenance cost, increased resource requirements. Therefore, there is a great need to detect the code clones by using various different approaches and prevent their introduction by constantly monitoring and parsing the source code during its evolution as detection is also necessary to find the place where a change must be replicated to monitor the development and to use the refactoring tool which improves the structure of object oriented programs while preserving their external behavior. Refactoring tool is a direct way to improve the quality of the source code; there are several other benefits and applications of detecting clones. A clone detector fined pieces of source code which are similar on the basis of text or functionality.



II. DRAWBACKS OF CODE DUPLICATION[1]

Code Clones can have severe impacts on the quality, reusability and maintainability of a software system.

- *Increased probability of bug propagation:* If a code fragment contains a bug and that code fragment is used by another fragments then bug propagates.
- *Increased probability of introducing a new bug:* If the structure of the fragment is reused not the code, probability of introduction of new bugs.
- *Increased probability of bad design:* It leads to bad inheritance structure, abstraction and maintainability of the software.
- *Increased difficulty in system improvement:* Difficult to understand the existing code implementation and to add new functionality in the system.

[1] Copying a code fragment and reusing it by pasting with or without minor modifications is a common practice in software development is known as code duplication.

- *Increased maintenance cost:* When maintaining or enhancing a piece of code, duplication multiples the work to be done hence lead to increased maintenance cost.
- *Increased resource requirements:* While system size may not be a big problem for some domains while telecommunication switch or compact devices may require costly hardware upgrade with a software upgrade.

III. IMPORTANCE OF CLONE DETECTION

- *Detects library candidates:* Usability proves by multiple times use of copied code fragments. [1, 2]
- *Helps in program understanding:* When we have a piece of code managing memory we know that all files which contain a copy must implement a data structure with dynamically allocated space [1, 2]
- *Detects malicious software:* Clone detection helps in finding malicious software and by comparison of malicious software it is possible to find the matching parts of both the software's [1, 2]
- *Detects plagiarism and copyright infringement:* code detection also help in detection of plagiarism and copyright infringements.

IV. CLONE TERMINOLOGIES

The tools which detect the clones give report in the form of Clone Pairs (CP) or Clone Classes (CC) or both The similarity relation between the cloned fragments is the equivalence relation (i.e., a reflexive, transitive, and equivalence relation) [1, 2].Different definitions of similarity has different kinds and degrees of clones [3].Sequences are sometimes original character strings, strings without whitespace, sequences of token type, transformed token sequences and so on. Following clone pair and clone class are defined in the terms of clone relation:

Fragment 1:	Fragment 2:	Fragment 3:
for (int i=1;i<n ; i++){ sum =sum+ i ; } (a)	for (int i=1;i<n; i++){ sum =sum + i ; } (a)	.....
if (sum <0){ sum =n- sum; } (b)	if (sum < 0){ sum = n – sum ; } (b)	if (result <0){ result=n- result; } (a)
.....	while (sum<n){ sum= n/sum; } (c)	while (result<n){ result=n/resu } (b)

Fig.1: Clone pair and Clone class

Clone Pair: A clone pair is a pair of code fragments or portion if there exist a clone relation between them, i.e., a clone pair is a pair of code fragments which are identical or similar to each other. For the threecodefragments,Fragment1(F1),Fragment 2(F2),Fragment 3(F3) of figure 1,we can get five clone pairs ,<F1(a),F2(a)>,<F1(b),F2(b)>**,** <F2 (b), F3 (b), <F2(c), F3 (b)> and <F1 (b), F3(a)>.

Clone Class: A clone class is a maximal set of code fragments in which any two fragments is a clone pair. For the three code fragments of figure1,we get a clone class <F1(b),F2(b),F3(a)> where the three code fragments F1(b),F2(b) and F3(a) form clone pairs with each other. A clone class is simply the union of all clone pairs that have clone fragments in common.

Clone Class Family: The groups of all clone classes that have a same domain are called a class family [1, 2] or super clone [2]. Example of domains is files, functions, classes and packages.

Code Fragment: Any sequence of code lines which may or may not have comment lines is called code fragment [1]. Code fragments may have any type of granularity level, e.g., function definition, begin block, or sequence of statements. A CF is identified by its file name and begin-end line numbers.

Clone Types: Code fragments are similar in two ways. Fragments are similar based on program text or functionality. There are four types of clones. Type1 to type3 clones are based on text whereas type4 clone is function based [1, 2, and 3]

Type1: Type1 clones are exact copy or identical code fragments except for variations in whitespace, layouts and comments.

Type2: Type2 clones are syntactically identical fragments except for variation in identifiers, literals, types, whitespace, layout and comments.

Type3: Type3 clones are basically the copied fragments with further modifications such as changed, added or removed statements, in addition to variations in identifiers, literal, type, whitespaces, layout and comments.

Type4: Type4 clones are those in which two or more code fragments that perform the same computation but implemented through different syntactic variants.

V. OVERVIEW OF CLONE DETECTION TECHNIQUES:

The detection of code clones is a two phase process which consists of a transformation and a computation phase. In the first phase we use the source text which is transformed into an internal format which allows the use of more efficient comparison algorithm. In the second comparison phase the actual matches are detected. So it is necessary to classify the

detection techniques according to their internal format [4]. The overview of code clone detection techniques are as follows:

A. String Based

String based approach uses these two phases i.e., transformation and comparison algorithms which makes the string based approach independent of the programming language [4].String based approach is known as text based technique which use no transformation or normalization on the source code before the actual comparison [3]. String based comparison are of two types:

1. **Simple line Matching:** In this simple line matching technique only minor transformations are done using string manipulations operations .Typical transformations are removal of white spaces and empty lines. During the comparison phase all lines are compared with each other using string matching algorithm. Before comparing all the lines, large search space which is reduced using hashing buckets as all the lines are hashed into one of n possible buckets then all pairs in the same bucket are compared.
2. **Parameterized Line matching:** This is another technique for comparison of strings which detects both identical as well as similar code fragments. Identifier-names and literals may change when cloning a code fragment; this change is considered as changeable parameters. To perform such parameterization, the set of transformations is extended with an additional transformation that replaces all identifiers and literals with one, common identifier symbol like "$". Due to this additional substitution the comparison becomes independent of the parameters.

B. Token Based

Token Based techniques use transformation algorithm[2] by constructing a token stream from the source code, so lexical analysis is done. Lexical approaches begin by transforming the source code into a sequence of lexical "tokens" using compiler-style lexical analysis. The sequence is then scanned for duplicated subsequences of tokens and the corresponding original code is returned as clones. This technique helps to find type1 and type2 clones. As syntax is not taken into account, clones found by token-based techniques may overlap different syntactic units [2].Efficient token-based clone detection is based on the suffix tree which is a representation of a string as a ordered tree data structure that is used to store an array where keys are strings [3].

Parameterized Matching With Suffix Trees

It consists of three consecutive steps manipulating a suffix tree as internal representation. Firstly, a lexical analyzer passes over the source text transforming identifiers and literals in parameter symbols. One symbol always refers to the same identifier, literal or structure. This transformed source text is known as parameterized string or p-string. Secondly, when the p-string is constructed, it is checked whether the two sequences in the p-string are a parameterized match or not by using some criterion. Two strings are parameterized match if one can be transformed into the other by applying a one-to-one mapping renaming the parameter symbols. An additional encoding is necessary to verify this above criterion .In this encoding, each first occurrence of a parameter symbol is replaced by a 0. All later occurrences are replaced by the distance since the previous occurrence of the same symbol. So it is observed that when the two sequences have the same encoding, they are same except for a systematic renaming of the parameters symbols. When the lexical analysis is done, a data structure called a parameterized suffix tree also called as p-suffix tree which helps in more efficient detection of maximal, parameterized matches. Thirdly, find the maximal paths in the p-suffix tree that are longer than a predefined character length [4].

C. Tree Based

In the tree-based approach a program is converted into a parse tree or abstract syntax tree (AST) with the help of any language parser to yield syntactic clones [1]. The parse tree contains the complete information about the source code. In this approach similar sub trees are the searched in the tree with some tree matching techniques and the corresponding source code of the similar sub trees are returned as a clone pairs and clone classes. To find the clones by using abstract syntax tree we compare each sub tree to each other sub tree. For comparison of sub trees use a hash function[3].The similarity metrics measures the fraction of common nodes of two trees [3]. To find the syntactic units, perform the decomposition of resulting type-1/type-2 token sequence from the serialized AST. The serialized AST is generated by parsing the program. There is a new approach similar to AST for finding the syntactic differences between two versions of the same programs by generating a parse tree for both the versions of the programs. AST based approach disregards the information about identifiers, ignores data flows. This limitation of AST based approach is improved by the PDG based approach by considering the semantic information of the source and it also contains the control and data flow information of the program.

VI. CONCLUSION

Software clone is a phenomenon in large software system. It is usually caused by programmer's copy and paste activities. The reason for the existence of clones in the source code is that

[2] Transformations are done by using Pretty printing (reorganization) of the source code of different layouts, removal of comments, spaces, tokenization, parsing, generating PDG, and normalizing identifiers.

[3] Hash function partitions the AST into similar sub trees.

making a copy of code fragment simpler and faster than writing it from the scratch. Sometimes programmers does not understand the program and re-implement the same functionality. Another reason for code cloning is a time limit that is assigned to the developer to finish the project in that case programmer copy and paste the code and update it according to the new requirements.

Although it seems to be simple and effective method these duplication activities is usually weak documentation that causes a number of negative effects on the quality of the software system and increases the amount of code to be maintained. Duplication also increases the defect probability and resource requirements. Code clone detection is an active research area with plenty of work in detecting and removing the clones from the software system.

After analyzing various code clone detection techniques i.e., string based, token based and tree based it is concluded that these techniques helps us to remove the complexity and big structure of the software system. The analysis based token-suffix trees offers several advantages over other techniques. It scales very well because of its linear complexity in both time and space. Token based analysis is more reliable and independent of the layouts, yields syntactic clones. AST is more expensive than generation of a token sequence as AST nodes are visited many times both in the comparison within a partition and across partitions because the same node could occur in a sub tree subsumed by a larger clone contained in a different partition. This review paper is focused on the technique that helps to check and evaluate the code thoroughly so that unusable (duplicate) code can be reduced.

VII. REFERENCES

[1] Chanchal Kumar Roy and James R. Cordy.(2007,September).A Survey on Software Clone Detection Research.In Natural Sciences and Engineering Research Council .

[2] Basit,H.A.,&Jarzabek,S.(2007,September).Efficie--nt token based clone detection with flexible tokenization.In Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering(pp.513-516).ACM.

[3] Koschke,R, Falke,R., & Frenzel,P. (2006, October). Clone Detection using abstract syntax suffix trees. In Reverse Engineering, 2006 .WCRE'06.13th working Conference on (pp.253-262).IEEE.

[4] Gahlot, Manisha .Comparative Analysis of Tree-Based and Text-Based Technique for Code Clone Detection. International Journal for Advance Research In Engineering and Technology, Vol.2, Issue II, Feb, 2014.

[5] Rysselberghe, F.V., & Demeyer,S.(2004, September). Evaluating clone detection techniques .In Proceedings of the 19th IEEE international conference on Automated software engineering (pp.336-339).IEEE Computer Society.

[6] Smith,R.,& Horwitz,S .Detecting and measuring similarity in code clones.In Proceedings of the International workshop on Software Clones(IWSC).

[7] Bari,M.A.,& Ahamad,S.(2011).Code Cloning: The Analysis, Detection and Removal.International Journal of Computer Applications, 20.