ISSN No. 0976-5697

Volume 3, No. 2, March-April 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Language Independent Approach for Method Level Clone Detection Using Fingerprinting

S.Mythili*
Asst.Prof. & Head,
Department of Information Technology,
Kongunadu Arts and science College,
Coimbatore-29.
smythili78@gmail.com

Dr.S.Sarala
Asst.Prof. in Information Technology,
School of Computer Science and Engg,
Bharathiar University,
Coimbatore -46.
sriohmau@yahoo.co.in

Abstract--Software Maintenance is an important part of Software Engineering activity. Maintenance of a software becomes very difficult when the size and complexity of the program increases. To reduce the complexity and size it is very necessary to find the similar code fragments known as code clones in a software system. Software maintenance is very much dependent on the duplicated code in the code fragment. To reduce the software maintenance cost it is necessary to find the similar code fragments. The capability and the effectiveness of the similarity measurement depends on the measurement technique used for the code clone detection.

In this paper we propose language independent method level clone detection based on the Rabin-Karp fingerprint representation. Rabin-Karp is an effective string matching algorithm for identifying various similar duplications of similar fingerprint fragments in a software system by the method of hashing.

The specific purpose of this system is to detect duplicated code between the programs written in different programming language. This system also uses a tool WordNet to identify the lexical similarity which aids code clone detection.

Keywords: software clones, clone detection, similarity matrix, fingerprinting, Rabin-Karp

I. INTRODUCTION

Copying code fragments and then reuse by pasting with or without minor modifications or adaptations are common activities in software development. This type of reuse approach of existing code is called code cloning and the pasted code fragment is called a clone of the original[1].

The area of clone detection has received wide interest recently as indicated by numerous efforts in clone detection tool development . A clone detector must try to find pieces of code of high similarity in a system's source text. The main problem is that it is not known beforehand which code fragments may be repeated. Thus the detector really should compare every possible fragment with every other possible fragment. Such a comparison is prohibitively expensive from a computational point of view and thus, several measures are used to reduce the domain of comparison before performing the actual comparisons. Even after identifying potentially cloned fragments, further analysis and tool support [2]may be required to identify the actual clones.

Although cloning leads to redundant code, not every redundant code is a clone. There may be cases in which two code segments that are no copy of each other just happen to be similar or even identical by accident. Also, there may be redundant code that is semantically equivalent but has a completely different implementation .

Code cloning is found to be a more serious problem in industrial software systems. In presence of clones, the normal functioning of the system may not be affected, but without countermeasures by the maintenance team, further development may become prohibitively difficult .Clones are believed to have a negative impact on evolution. Code clones may adversely affect the software system's quality, especially their maintainability and comprehensibility.

Software clones appear for a variety of reasons:

a. Code reuse by copy and paste
b. Coding styles
c. Performance enhancement
d. Accidents

During maintenance clones increase the risk of updating different copies. When one fragment is changed the rest of the copied fragment needs to be updated.This leads to software aging.

We distinguish two types of clones namely simple clones and structural clones.

Simple clones - Contiguous segments of similar code such as class method or fragments of method implementation.

Structural Clones-Patterns of inter-related classes emerging from the design and nalysis[3]..In this paper we concentrate on simple clones and this work will be further extended to structural clones.

The remainder of this paper is organized as follows. Section 2 describes about the General terms and definitions . Section 3 gives details of the Proposed Work ,Section 4 discusses about the related work , Section 5 presents the results and experiments and the future work and conclusion is given in Section 6.

II. GENERAL TERMS AND DEFINITIONS

Still now there is not yet a universal definition of clone. It is considered to be a fragment that is repeated in the same project or a program for many times. There are some common definitions a terms used in code clones they are

A. Code Fragment :

A code fragment(CF) [2] is any sequence of code lines . It can be of any granularity, e.g., function definition, begin-end block, or sequence of statements. A CF is identified by its file name and begin-end line

B. Code Clone:

A code fragment CF2 is a clone of another code fragment CF1 if they are similar by some given definition of similarity, that is, f(CF1) = f(CF2) where f is the similarity function .Two fragments that are similar to each other form a clone pair (CF1; CF2), and when many fragments are similar, they form a clone class or clone group.

C. Clone Types:

The clones are classified as

a) *Type I* – They are normally called as exact clones. It is a code clone that is identical to one another without considering variations in whitepaces and comments

b) *Type II* – They are called as Renamed or Parameterized clones. These clones are structurally or semenatically similar but differing in the identifier,literals and their types.

c) *Type III* - They are called as Gapped Clones .In this type the statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout and comments.

d) *Type IV*- They are called as semantic clones. Here the code fragments perform the same computation but implemented through different syntactic variants.

D. Fingerprinting Matching Technique:

Fingerprinting techniques mostly rely on the use of Kgrams because the process of fingerprinting divides the document into grams of certain length k. Then, the fingerprints of two documents can be compared in order to detect plagiarism. It has been observed through the literature that fingerprints matching approach differs based on what representation or comparison unit. There are three types of fingerprint matching technique they are

a) Character Based fingerprinting – It is the most conventional method and it uses sequence of characters to calculate fingerprints for the whole document.

b) Phrase Based fingerprinting - generates fingerprint using phrase mechanism to measure the resemblance between two documents.

c) Sentence based fingerprinting – calculates the fingerprints for each sentence.

A *fingerprint* of an object *Ob* is a small string *f(Ob)* with the following properties:

a. *f* is a function of *Ob*. In particular, if two objects are equal, then so are their fingerprints.

b. 2). Prob($f(Ob_1) = f(Ob_2)$) << 1 for "random" objects $Ob_1 \neq Ob_2$.

Fingerprints are used to:

i. Identify Objects
ii. Compare Objects Remotely
iii. Test an Object for Changes

Since fingerprints are smaller, they are very useful and easy for comparison. These fingerprints are useful in the area of code cloning during the maintenance phase for identifying the clones.

In this paper we use the fingerprinting technique to identify Type I ,Type II and Type III clones .Type IV is not within the scope of this paper

III. PROPOSED METHOD

The Proposed method consist of the following steps

a. Pre-Processing
b. Find the Lexical meaning using the WordNet tool
c. Converting to a general format
d. Fingerprint generation using Rabin-Karp
e. Comparing the fingerprints
f. Creation of a Similarity Matrix
g. Identifying the code clones

This diagram for the overall process is shown in Fig.1

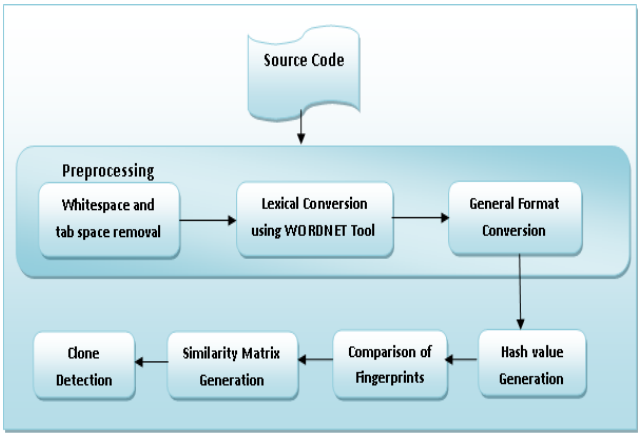

Figure 1. Clone Detection Process

A. Preprocessing :

In this process the given source code Fig.2 is scanned line by line for whitespaces and for tabs .Normally programmers may include many whitespaces and tabs to improve the readability of the code. By removing these characters we get the original code without extra formatting which will be helpful to find clones.

```
int calc(int n){
float tot=0.0;
float mul=1.0;
for(int i=1;i<=n;i++){
 tot= tot+i;
mul=mul*i;
grade(s,p);}}
                    (a)
```

```
void calc(int n) {
float total=0.0;
float multi =1.0;
while(i<=n)
{ if (i>0)
total= total +i;
multi = multi * i;
grade(total,multi); }}   (b)
```

```
void calculate(int n) {
float sum=0.0;
float prod =1.0;//C2
for (int i=1; i<=n; i++) {
sum=sum + i;
prod = prod * i;
grade_calc(sum, prod); }}
                    (c)
```

```
void calu( int n) {
double s=0.0; //C1
double p =1.0;
for (int i=1; i<=n; i++)
{s=s + i;
grade(s, p); }}
                    (d)
```

Figure 2. Sample methods taken for processing

B. Finding the Lexical meaning using WordNet:

It is a common practice for the programmers to copy a code and make it different just by changing the method names. In order to identify such methods we use the Natural language processing tool called WordNet.

For example

void add(int,double)
void sum(int,double)

might have the same code but they may be available with different method names.We use the WordNet tool to identify such differences and convert them to a common name.Due to this the above two line will be identified as clones.

C. Converting to a general format:

Any variable name or a data type found in the source code is converted into a general format. For example int a is changed as DAT $p where DAT refers to any datatype and $p refers to the variable name 1.By doing this any code which appears to be different just by changing the variable names and datatypes will be identified as clones.The general format for the above sorurce code in Fig.2 is shown in Fig. 3.

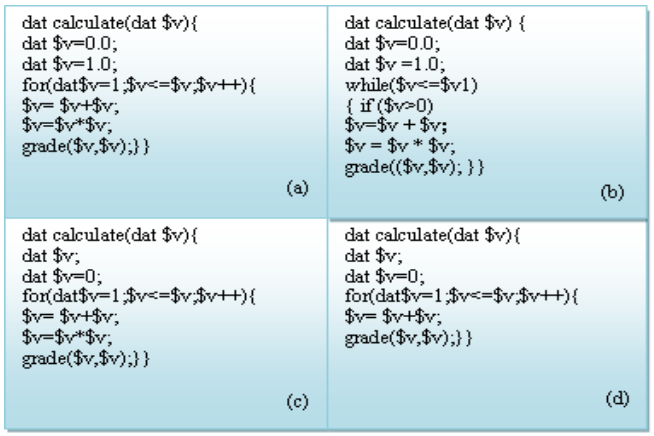

Figure.3 General Format Conversion

D. Fingerprint generation using Rabin-Karp:

After applying the above steps the transformed source code is ready for fingerprinting. Here we use the Rabin-Karp string matching algorithm Fig. 4 for generating the fingerprinting patterns. The fingerprint pattern generated are stored in the arrays and they will be ready for the comparison.

E. Comparing the Fingerprints:

The generated fingerprints of the pattern and the fingerprints of the source code are compared. If hash value of the pattern and the m-token sequence are equal then they are compared character by character to find the similarity.

```
function Rabin_Karp(code, pattern)
{
  - Let n be the size of the source code
  - Let m the size of the pattern,
  - If n < m return no match is possible
  - If n>m  calculate a hash for the pattern, M-tokens
  - If h1!= h2, calculate the hash value for next M- tokens .
  - If h1==h2,a Brute Force Comparision is made
}
```

Figure. 4 Rabin-Karp Algorithm

F. Creation of Matrix:

The result after comparison will generate a matrix of size m X n The matrix secures a value ‘1’ if the hash values of the pattern matches with the hash value of the compared source code. A non-match will secure a value ‘0’.Thus the resultant matrix will be a combination of ‘1’ s and ‘0’s.

G. Identifying the code clones:

To identify a clone we have to highlight or change the color of the line source code which has secured a value ‘1’ in the matrix.The identified clones are indicated in Fig.5

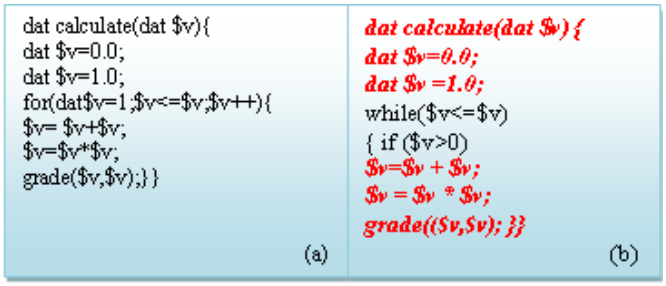

Figure 5. Clone detection by comparing (a) with (b)

IV. RELATED WORK

Different techniques have been used for identifying simple clones They are broadly categorized based on the program representation and the matching technique. There are different techniques like text based , tokens based, AST based, program dependence based and metrics based for code structures .Some of the techniques that are different from the normal matching techniques include suffix tree based token matching and fingerprints matching . In this paper we have used the fingerprinting technique for clone detection.

Ducasse,S.Rieger,M., and Demeyer,S has used a language independent approach for clone detection. In his paper he has used a parser to detect the clones in a more significant manner [4]

CCFinder uses a Token based technique which consists of the transformation of input source text and a token-by-token comparison. CCFinder is easily configurable to read input in different programming languages like C, C+, Java and COBOL. A suffix-tree matching algorithm is used for the discovery of clones and to reduce the complexity of token matching algorithm. [5].Some optimization techniques are applied.

Heejung Kimy, Yungbum Jung, Sunghun Kim, Kwankeun Yi has developed Mecc Memory Comparision based clone detector. This proposes a new semantic clone detection technique by comparing programs abstract memory states, which are computed by a semantic-based static analyzer[6].

Clone Detection Using Abstract Syntax Suffix Trees by Rainer Koschke make use of suffix trees to find clones in abstract syntax trees. This new approach is able to find syntactic clones in linear time and space.[7].

Clone Detection Using Abstract Syntax Trees by Ira D. Baxter presents simple and practical methods for detecting exact and near miss clones over arbitrary program fragments in program source code by using abstract syntax trees[8].

Fingerprinting techniques have been used in different areas of computing research. In software clone detection research a number of approaches used fingerprinting with normalized source code or with Abstract Syntax Trees(ASTs).

Johnson [9] presents a detection mechanism that uses fingerprints to identify exact repetitions.

Md. Sharif Uddin Chanchal K. Roy Kevin A.Schneider and Abram Hindle [10] presents a clone detection method.using simhash..They have used this method for detecting Type-3 near-miss clones in large scale softwares.

Randy Smith and Susan Horwitz has detected and measured similarity in code clones by using fingerprinting tecnnique[11].In their work they have grouped the clones

into a clone cluster based on a user-defined threshold value.They have also ordered them according to the similarity rank.

Minhaz F. Zibran Chanchal K. Roy in their paper" Towards Flexible Code Clone Detection, Management,and Refactoring in IDE"[12] has developed an IDE based clone managenet system to flexibly detect, manage, and refactor both exact and near-miss code clones.

V. EXPERIMENTS AND RESULTS

We have taken some sample methods written in C language for out experiments.The system developed to identify the clones is also developed using the C language .In our result we have generated a matrix which is a combination of '1' and '0's.Where the 1's indicate a match of a particular line and a '0' indicate a non-match of a line with the compared soruce code. Finally the lines which has secured the value '1' has been highlighted. The following code is used for the hash value generation.

```
int hashn(char* str, int n)
{
char ch = str[n];
int sum;
str[n] = '\0';
sum = hash(str);
str[n] = ch;
return sum;
}
```

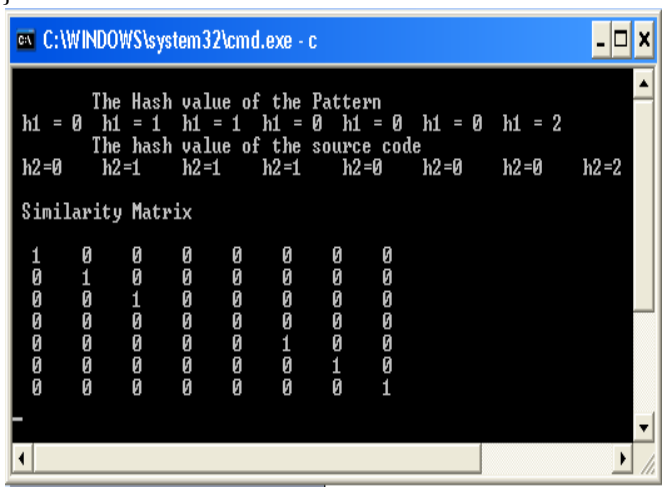

Figure. 6 Generated Output

The generated output is shown in Fig. 6

VI. CONCLUSION AND FUTURE WORK

In Contrast with the other techniques used for clone detection our system uses the Rabin-Karp fingerprinting technique for identifying method level Type-1,Type-II and Type-III clones. It generates the fingerprint and compares the fingerprint. When compared with the other fingerprinting techniques the Rabin-Karp is the best method because it makes the brute-force comparison only when the fingerprints are same. The time taken by the Rabin_Karp Matcher is $\Theta(m)$ preprocessing time. Further this system also concentrates on the Language Independent feature by a general format conversion..The output of this system is a matrix and a set of highlighted lines of code that are identified as clones. This system is further enhanced to find the clone pairs and to make improvements to find the structural clones. It is also necessary for us to concentrate on ranking algorithms to rank the most similar documents. Further it can be enhanced to find the semantic level clones.

VII. REFERENCES

[1]. Chanchal Kumar Roy and James R Cordy, "A Survey on Software Clone Detection Research", Computer and Information Science, Vol. 115, No. 541, pp. 115, 2007.

[2]. Chanchal K. Roy, James R. Cordya and Rainer Koschkeb, "Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach", Journal Science of Computer Programming, Vol. 74, No.7, pp. 470-495, May 2009.

[3]. Hamid Abdul Basit and Stan Jarzabek "Detecting High level similarity patterns in programs",European Software Engineering Conference and ACM SIGSOFT symposium in the Foundations of Software Engineering 2005

[4]. S. Ducasse, M. Rieger, and S. Demeyer. A language independentapproach for detecting duplicated code. In *Proceedings;IEEE International Conference on Software Maintenance*,1999.

[5]. T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for large Scale Source Code," IEEE Trans. Software Eng., vol. 28, no. 7,pp. 645-670, July 2002.

[6]. Heejung Kim, Yungbum Jung, Sunghun Kim, Kwankeun Yi "MeCC : Memory Comparison-based Clone Detector" (ICSE), 2011 33rd International Conference on Software Engineering ,PP 301-310.

[7]. Rainer Koschke "Clone detection for Abstract syntax Suffix tree" 13th Working Conference on Reverse Engineering-2006.

[8]. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier.Clone detection using abstract syntax trees.In proceedings;International Conference on Software Maintenance, 1998.

[9]. J. H. Johnson, "Identifying redundancy in source code usingfingerprints", Proc. *CASCON*, 1993, pp. 171-183.

[10]. Md. Sharif Uddin Chanchal K. Roy Kevin A. Schneider, Abram Hindle "On the Effectiveness of Simhash for DetectingNear-Miss Clones in Large Scale Software Systems" (WCRE), 2011 18th Working Conference on Reverse Engineering,pp 13 – 22.

[11]. Randy Smith and Susan Horwitz "Detecting and Measuring Similarity in Code Clones" Proc of IWSC 2009.

[12]. Minhaz F. Zibran Chanchal K. Roy in their paper" Towards Flexible Code Clone Detection, Management,and Refactoring in IDE" Fifteh International workshop on Software Clones 2011.