ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Generic Algorithm for Detecting Duplicate Code

P.Nagalakshmi, S.V.P.K.Satyadev

PG Scholar, Dept. of CSE, Vignan's Institute of Information Technology, Visakhapatnam, India

Asst Professor, Dept. of CSE, Vignan's Institute of Information Technology, Visakhapatnam, India

ABSTRACT*:* Code Cloning is a programming term for a sequence of source code that occurs more than once, either within a program or across different programs. It is also called Duplicate code. Code duplication may increase maintenance costs. Code duplication may occur for many reasons like programmers reuse code through copy paste.Clones are classified into different classes based on the extent of modifications that we performed on the cloned code. They classified into type1 same code fragments without any modification, type2 identical in structural or syntactical, type3 same code fragments but small changes in statements and type4 functionality and computation of two or more code fragments are same but they implementation is different. We are having different techniques for finding code clones they are mainly classified into Textual Comparison, Token Comparison, Metric-Based Comparison, Abstract Syntax Tree Comparison and Program Dependency Graph for finding the code clones. In this paper we are try to find out code clones and percentage of code cloning by using textual comparison. For this purpose we are using multiway tree for indexing and knuth morris pattern matching algorithm for the comparison of the code. This code detection generic algorithm will construct a heirarical tree for the given code and it is used for the indexing and to find the extent of code duplication. This generic algorithm is used for heirarical indexing for tracing the programming process development and kunth morris pattern matching algorithm for pattern matching which is having time complexity of $O(n^2)$.

KEYWORDS: code cloning, textual comparison, way indexing tree, knuth-morris part algorithm, token comparison

I.INTRODUCTION

Code Cloning is a term programming for a sequence of source code that occurs more than once, either within a program or across the different programs. It is also called Duplicate code. Duplicate code is generally considered as undesirable for a number of reasons. A minimum requirement is usually applied to the quantity of code that must appear in a sequence for it to be considered duplicate rather than coincidentally similar. The automated process of finding duplications in source code is called clone detection.

It is still a common habit of programmers to reuse code through copy paste. The interests of these strategies must be paid back later by increased maintenance through replicated changes in all copies if the original code must be corrected or adapted. Various techniques have been proposed to find duplicated redundant code or software clones.

Clones are of different types based on their textual or functionality. They are

- **Type1:** same code fragments without any modification except there is variation in spaces and comments
- **Type2:** identical in structural or syntactical except change in variables, datatypes, literals and comments
- **Type3:** same code fragments but small changes in statements either added or removed and also changes in literals, variables and comments

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

- **Type4:** functionality and computation of two or more code fragments are same but they implementation is different

Code clone terminologies

- **Clone Pair:** A pair of code portion/fragments is called clone pair if there exists a clone relation between them.
- **Clone class:** A clone class is the maximal set of code fragments in which any two of the code fragments holds a clone relation i.e. form a clone pair. A clone class is therefore, the union of the clone pairs which have code portions in common. Clone classes are also called clone communities.
- **Clone Class Family:** The group of all clone classes that have the same domain is called clone class family. Such a clone class family is also termed as super clone. In their context, multiple clone classes between the same source entities are aggregated into one large super clone.

So many techniques have been proposed for these types of clones

Textual Comparison: In these technique whole lines to each other compared textually by using hash function for strings. Same partition lines are need to be compared.

Token Comparison: Code clone are transformed into tokens that are formed from lexical analysis. These tokens are compared and analyzed to find similar clone pattern.

Metric-Based Comparison: in this technique gather different metrics for code fragments and compare these metrics instead of comparing code directly. An allowable distance (for instance, Euclidean distance) for these metrics can be used as a hint for similar code. Specific Metric-based techniques were also proposed for clones in web sites.

Abstract Syntax Tree Comparison: partition sub trees of the abstract syntax tree of a program based on a hash function and then compare sub trees in the same partition through tree matching (allowing for some divergences). A similar approach was proposed earlier by Yang using dynamic programming to find differences between two versions of the same file.

Program Dependency Graph: Control and data flow dependencies of a function may be represented by a program dependency graph; clones may be identified as isomorphic subgraphs because this problem is NP hard, Krinke uses approximative solutions.

II.RELATED WORK

In this study three duplicated code detection techniques are considered. By means of five small to medium cases (some of them including generated code, hence having lots of duplication) we compared the results, focussing on those portions where the techniques performed differently.

Simple line matching (representative for the string-based techniques) gives a crude overview of the duplicated code that is quite easy to obtain, hence is most appropriate during problem detection and problem assessment.

Parameterized matching (representative for the token-based approaches) provides a precise picture of a given piece of duplicated code and is robust against rename operations. Therefore it works best in combination with fine-grained refactoring tools that work on the level of statements (i.e. Extract Method, Move Behavior Close to Data, and Transform Conditionals into Polymorphism.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

III.PROPOSED WORK

In this paper we are try to find out code clones and percentage of code cloning by using textual comparison. For this purpose we are using multiway tree for indexing and knuth morris pattern matching algorithm for the comparison of the code. This code detection generic algorithm will construct a heirarical tree for the given code and it is used for the indexing and to find the extent of code duplication. This algorithm is used for heirarical indexing for tracing the programming process development and kunth morris pattern matching algorithm for pattern matching which is having time complexity of $O(n^2)$.

Generic Algorithm:

Input: source code of any programming language.

Output: duplicate code detected and percentage of code duplicated.

Step1: call m-way tree indexing algorithm

For tree generation.

Step 2: Algorithm

Input: tree generated by m-way indexing
for each node in the m-way tree
{
for each line i at the node
{
k=i;
l=k+1;
if(compare(k,l))
{
l++;
}
until(compare(i,j));
else
{
i++;
}
let i be the first line in the first node
let j be the first line in any other node in the tree
k=i;
if(compare (i,j))
{
i++;
j++;

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

}until(compare(i,j))

else

{

i=k;

j= 1st line in the next node;

}

Parsing

Parser is used in this code clone detection system. The Parser is used to parse any programming source code file and desired output can be made available as per the action code implementation. The Action-Code written into parser would help us to prepare input as a key-value pair. Key would be position of syntax with structure as File No, Method No, Block No, Line No and value would be statement as a string object corresponding to the line no.

Architectural Strategy

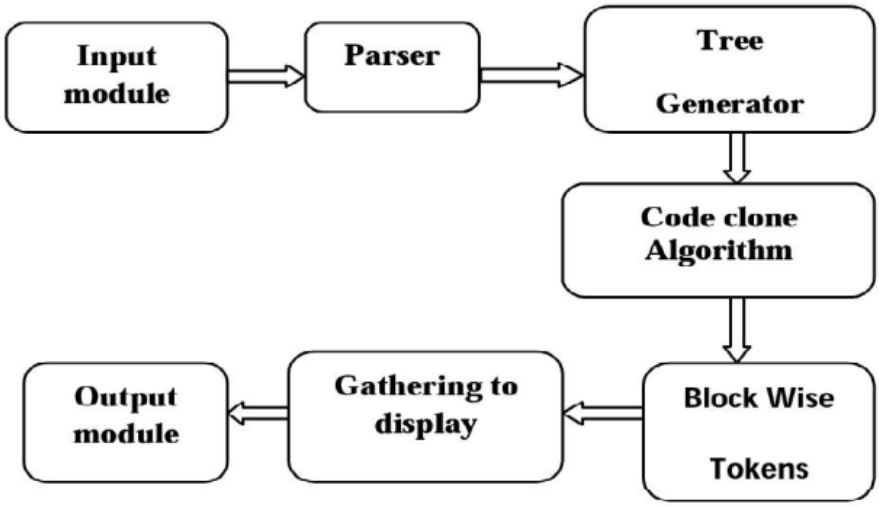

Fig.1. System Architecture

The system architecture explains how the process is working to get an output module. The tree is generated when input module is given and obtains the output module.

Tree Generator

Tree can be generated in different ways and formats. There is no standard specification for generation of Tree. Tree generated by different compliers are different and depend on the requirements. In this system, we are using m-way trees for generating the tree and scope for the generating tree is by considering the each and every module of the starting and ending braces.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Algorithm of m-way tree:

Input: A source code file

Output: Multiway index tree

Algorithm: multiway indexing tree Generator

Step1: in the each beginning of the scope define a node in the tree.

//that node represents all the code present in that scope with each index and combined all the node id of that scope.

Step2: for each such scope represented by the given parent node define a child for that parent node.

//this child nodes represent the code present at the scope.

Step3: do the above two steps recursively until the end of the parent scope.

Step4: do the above steps for all the different scope.

Code Clone Algorithm

Sub-trees are Type-1 clones if the entire tree including operand and operator of sub-tree is equal. Sub-trees are Type-2 clones operator node of both subs tree is same but operand may not be same. For Type-2 clones, operator nodes must be same but operand nodes can be different. Sub-trees in this case are structurally same. The applied code clone algorithms gives result in Key Value format.

Algorithm: kunth-Morris Pratt algorithm

Input: tree nodes generated by multiway indexing tree

Output: lines which are found similar

```
function kmp integer;
var i, j: integer;
begin i:= 1; j:= 1; initnext;
repeat if (j = 0) or (a[i] = p[j])
then begin i:= i+1; j:= j+1 end
else begin j:= next[j] end;
until (j > m) or (i > n);
if  j > m then kmp= i – m
 else kmp= i;
end;
procedure initnext;
var i,j: integer;
begin i:= 1; j:= 0; next[1]:= 0;
repeat if (j = 0) or (p[i] = p[j])
then begin i:= i+1; j:= j+1; next[i]:= j; end
else begin j:= next[j] end;
until i > M;
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

end;

Block wise Tokens

This process is essential to detect the block level clone as tokens are divided into groups called block, which is a block of code starts with opening brasses and ends with closing brasses. In this, compare all keys with each other except line no. Any difference found will lead to add a new Block Token with key value structure.

Gathering the data to display

Collect all the data i.e., tree generated from the m-way indexing tree and similar code found using kmp algorithm and duplicate code found from generic algorithm and percentage code duplicated found by dividing the total number lines by lines duplicated. All data is displayed in result.

IV.RESULT ANALYSIS

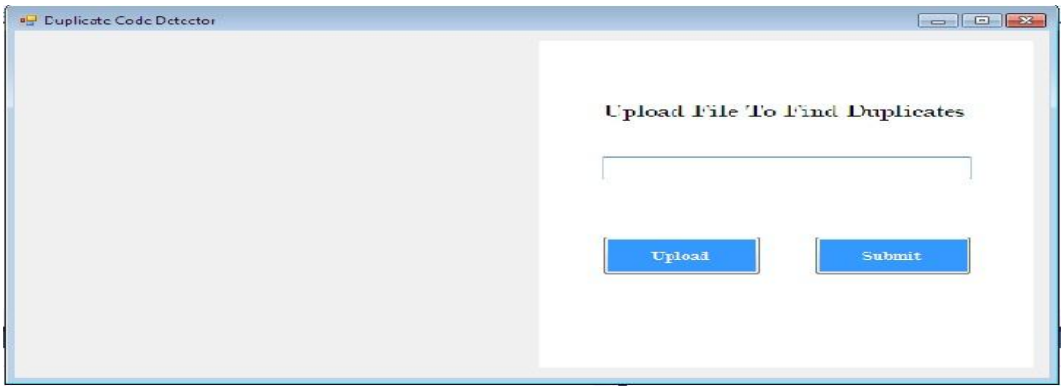

Fig .2.upload the source files

For experimental analysis we implemented the proposed approach with .NET programming language, we upload the any programming source file for detection of the duplicate code

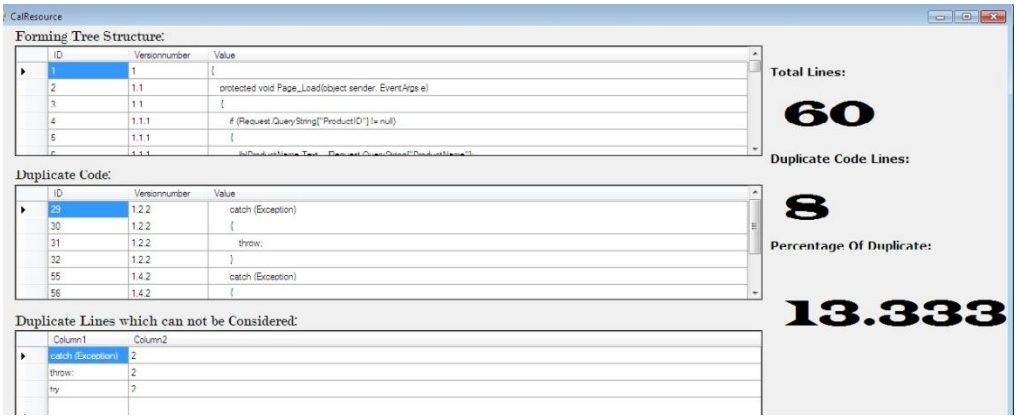

Fig .3.duplicated code detected and percentage of code duplicated

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

After uploading the source code file tree is generated by multiway indexing tree and similar code found using kmp algorithm and duplicate code found from generic algorithm and percentage code duplicated found by dividing the total number lines by lines duplicated. All data is displayed in result.

V.CONCLUSION

We have been concluding our current research work with efficient generic algorithm for detecting duplicated code present in the source code file that is uploaded. It gives code duplicated and percentage of code duplicated. Further by using percentage of code duplicated effort estimation can be calculated and using this cost and quality can be calculated by some methodologies.

REFERENCES

[1]. Evaluating Clone Detection Tools for Use during Preventative Maintenance Elizabeth Burd, John BaileyThe Research Institute in Software Evolution University of Durham South Road Durham, DH1 3LE, UK.
[2]. Parameterized Pattern Matching: Algorithms and Applications*,Brenda S. Baker- ATHT Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974 Received September 28, 1993.
[3]. J. H. Johnson. Identifying redundancy in source code using fingerprints. In CASCON, pages 171–183. IBM Press, 1993.
[4]. T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code. IEEE Transactions on Software Engineering, 28(7):654–670, 2002.
[5]. C. Kapser and M. Godfrey. A taxonomy of clones in source code: The reengineers most wanted list. In WCRE. IEEE CS Press, 2003.
[6]. R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. IBM Journal Research and Development, 31(2):249–260, Mar. 1987.
[7]. R. Komondoor and S. Horwitz. Using slicing to iden-tify duplication in source code. In Proc. Int. Sympo siumon Static Analysis, pages 40–56, July 2001.
[8]. K. Kontogiannis, R. DeMori, M. Bernstein, M. Galler, and E. Merlo. Pattern matching for design concept localization. In WCRE, pages 96–103. IEEE Computer Society Press (Order Number PR07111), July 1995.
[9]. K. Kontogiannis, R. D. Mori, E. Merlo, M. Galler, and M. Bernstein. Pattern matching for clone and concept detection. Automated Software Engineering, 3(1/2):79–108, June 1996.
[10]. J. Krinke. Identifying Similar Code with Program De-pendence Graphs. In Proceedings of the Eighth Work-ing Conference On Reverse Engineering (WCRE'01), 2001.
[11]. B. Lagu¨e, D. Proulx, J. Mayrand, E. M. Merlo, and J. Hudepohl. Assessing the benefits of incorporating function clone detection in a development process. In International Conference on Software Maintenance, pages 314–321, 1997.
[12]. F. Lanubile and T. Mallardo. Finding function clones in web applications. In CSMR, pages 379–386, 2003.
[13]. A. M. Leitao. Detection of redundant code using r2d2. In SCAM, pages 183–192. IEEE CS Press, 2003.
[14]. A. Marcus and J. Maletic. Identification of high-level concept clones in source code. In ASE, pages 107–114, 2001.
[15]. J. Mayrand, C. Leblanc, and E. M. Merlo. Experi-ment on the Automatic Detection of Function Clones in a Software System using Metrics. In Proceedings of the International Conference on Software Mainte-nance, pages 244–254, Washington, Nov. 4–8 1996. IEEE Computer Society Press.