ISSN No. 0976-5697

Volume 7, No. 6(Special Issue), November 2016

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Hybrid approach for Detecting Code Clone by Metric and Token based comparison

Deepali
Assistant Professor
Guru Nanak College
Budhlada,India

Ankur Gupta
Software Engineer
Amadeus India Pvt. Ltd.
Banglore

Chirag Batra
Technology Lead
TCS
USA, New York

***Abstract*—*In software development process, coping of existing code fragment and pasting them with or without modification is a frequent process. Clone means copy of an original form or duplicate. Software clone detection is important to reduce the software maintenance cost and to recognize the software system in a better way. There are many software code clone detection techniques such as text-based, token-based, Abstract Syntax tree based etc. and they are used to spot and finding the existence of clones in software system. One of the approaches to detect code clones is by analysis of different metrics for the programs. Another approach is token based comparisons of two programs to detect code clone. Our technique uses the combination of metrics and token based approach using hashing algorithm by analysis of two source codes in the process of finding clones among them and the percentage of cloning is reported as result.*

***Keywords*—** *software clone detection, metric based comparison, token based approach using hashing algorithm, java byte code.*

I. INTRODUCTION

In general, clones are set of identical segments of code in a software system, which has a bad impact in the system. In software development approach, duplicating previous code segments in different programs with or without modification is frequent process and that duplicated code is extremely difficult to maintain. The imitation in code is known as software clone and the phenomenon is known as software cloning. Code Cloning considered asa bad smell in software industry and has a bad impact on software quality, software maintenance and also increases maintenance cost. Roy and Cordy [1][2] mentioned software clone as software reuse. Although, it is a fast and instant method of software reuse yet, it is a harmful design procedure.

Baker [11][2] has taken a simple program and concluded that program code can be reduced by 14% based on exact matches and 61% based on parameterized matches. Several studies states that about 5% to 20% of software contains duplicate program segment [1] .Thus, it becomes very important to find the clones in programs accurately and in an efficient manner. Cloning in softwareis really harmful as it becomes really difficult to maintain software and its evolution. The reason behind cloning can be intentional or unintentional. There is major shortcoming of duplication in code fragment that if there is a bug in code segment to be duplicated, then that bug will be propagated at different places and the bug is to be tested multiple times at different places of the code and that would definitely increase the maintenance cost. Cloning on large extent increases the size of a system and results in design problems such as missing inheritance and procedural abstraction. The modification cost carried out after delivering of software product is figured out to be 40% - 70% of the total costs during lifetime of a system [2].

There are many software clone detection techniques and tools that differ from each other on the basis of approach used by them to detect clones. Cloning between two program codes is recognized on the basis of textual similarity and functional similarity. These types of similarities define the clone type. Based on the textual similarity we define the type 1, type 2 and type 3 clones.

Type 1 (Exact clone):- Identical code fragments except for variations in white spaces, layout and comments.

Type 2 (Renamed/Parameterized Clone): Syntactically same code fragments except for changes in identifiers, literals, types, whitespaces, layouts and comments.

Type 3 (Near miss clone):- Copied with further modification such as “change, add or remove” statements in addition to changes in identifier, literal, types, whitespaces, layouts and comment.

Based on the functional similarity we define type 4 clones imitated as a semantic clone.

Type 4 (Semantic clone):- Code fragment which are functionally similar but not textually similar.

This paper describe the hybrid clone detection technique using metric based comparison and token based comparison approach using hashing algorithm on the java source code. The metric based comparison is straightly applied on the source code for which many tools are available to find out metrics for the source code program. Later on, we apply a hashing based algorithm to execute the tokenbased comparison of the two programs to find more accurate clone results. This paper has 5 sections, section 2 describes the related work, section 3 describes the proposed work, section 4 describes the results, and section5 describes the conclusion and future scope.

II. BACKGROUND

Software clone detection is mainly implemented for the reorganization, preservation, refactoring and reengineering ofsoftware [4] [1]. By detecting clones, the computation cost and complexity of the software system can be minimized. Because of these factors software clone detection is new and an important research area.

Following are the software clone detection approaches:-

978-93-85670-72-5 © 2016 (RTCSIT)

CONFERENCE PAPER
International Conference on
Recent Trends in Computer Science & Information Technology (RTCSIT-2016)
21st August 2016
Guru Nanak College Budhlada, Punjab India

A. Textual approach:

Two code fragments are compared with each other to find matched sequence of texts or strings. If match is found then code is a clone pair by the detection technique.

B. Token based approach:

It needs a parser or lexer to normalize the code in form of token. So every line of source code is transformed into tokens then comparison applied on intermediate representation of code. The sequences of lines are compared through different algorithms. This technique is slower than text based approach and it is more robust.

C. Abstract syntax tree approach:

The actual source code is parsed into abstract syntax tree (AST) or parse tree and traverse the tree for finding similar sub tree. If match is found for sub tree is termed as clone.AST based approach finds even better results than the text and token based approaches but it is very complex to create an abstract syntax tree and clone detection using AST is acostlyprocedure on both time and memory.

D. Program dependency graph based approach:

It shows the control flow and data dependencies. When PDG is achieved from source code, graph isomorphism based comparison is applied to find match. For larger code it is very difficult to obtain PDG.

E. Metric based approach:

As an alternative of comparing two codes directly, metrics from source codes are obtained and these metrics are compared to detect clone. To calculate metrics,there are numerous softwares that canbe used such as Columbus, Source Monitor, Datrix etc. This technique is more scalable and accurate for big software system.

F. Hybrid based approach:

Hybrid based clone detection technique combines two or more than two different approaches to detect clones which increases its complexity but it is very active technique for detecting clones as compared to above discussed technique.

III. PROPOSED WORK

Figure 1 depicts the basic block diagram for the clone detection process. The proposed technique consists of two stages. First stage is a metric based comparison and Second stage is token based approach for clone detection. Firstly, the metrics are calculated as shown in figure 2 and then the comparison algorithm is applied on these metrics to detect clones.

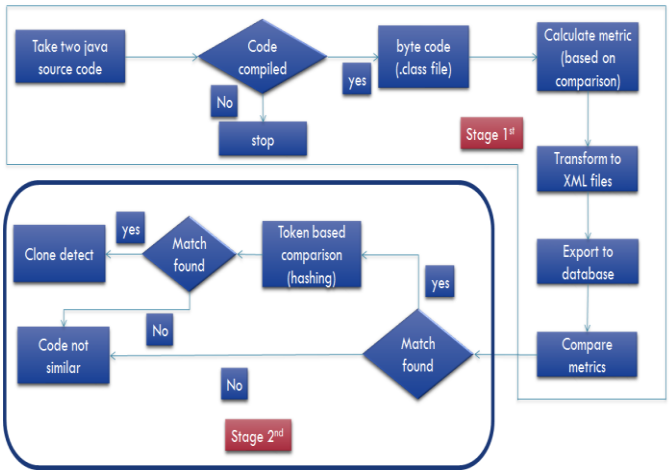


Figure 1: Block Diagram for Clone Detection approach

STAGE 1:

In metric based approach, different metrics are calculated from source code tool and then compared to detect code clone. It is more appropriate approach because it gives more precise result in large software system and also it can be appliedstraight on the source code.Straight Due to its application in large software systems, the technique holds more scalability. There are many tool that detect metric such as Source Monitor, Columbus, Datrix, MCD Finder etc. The Source Monitor tool is used to detect metric of java source code.

Metrics that are calculated from Source Monitor tool: -

1. No of files
2. No of lines
3. No of statements
4. Percentage branch statements
5. No of method call statements
6. Percentage lines with comments
7. Classes and interfaces
8. No of Methods per class
9. Average Statements per method
10. Line no of most complex method
11. Maximum complexity
12. No of line number of deepest block
13. Maximum block depth
14. Average block depth
15. Average complexity.

978-93-85670-72-5 © 2016 (RTCSIT)

CONFERENCE PAPER
International Conference on
Recent Trends in Computer Science & Information Technology (RTCSIT-2016)
21st August 2016
Guru Nanak College Budhlada, Punjab India

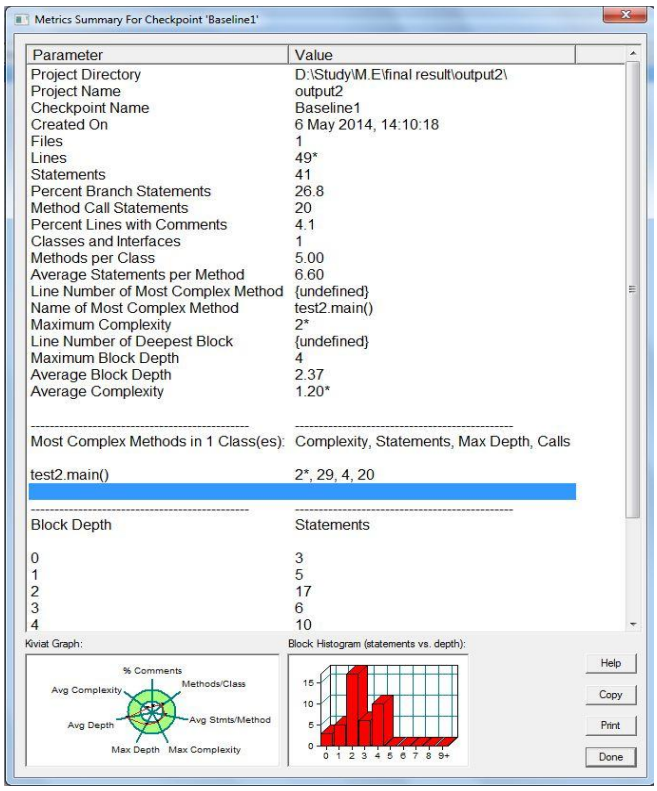
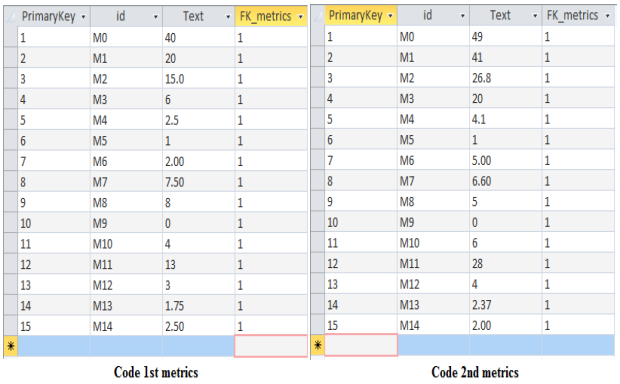


Figure 2: Result of Source Monitor tool for Code 1st

The results of the calculated metrics of java source code as shown in figure 2are transformed into theXML file (figure 3).

Figure 3: XML view of metrics

The next step is to extract the metrics from the XML file in to database (figure 4) that can later be used for comparison using JDBC.Once the metrics are extracted, they are compared for the two source programs to find out similarity among them and that similarity is reported as result in from of percentage of cloning (figure 5).



PrimaryKey	id	Text	FK_metrics
1	M0	40	1
2	M1	20	1
3	M2	15.0	1
4	M3	6	1
5	M4	2.5	1
6	M5	1	1
7	M6	2.00	1
8	M7	7.50	1
9	M8	8	1
10	M9	0	1
11	M10	4	1
12	M11	13	1
13	M12	3	1
14	M13	1.75	1
15	M14	2.50	1

Code 1st metrics

PrimaryKey	id	Text	FK_metrics
1	M0	49	1
2	M1	41	1
3	M2	26.8	1
4	M3	20	1
5	M4	4.1	1
6	M5	1	1
7	M6	5.00	1
8	M7	6.60	1
9	M8	5	1
10	M9	0	1
11	M10	6	1
12	M11	28	1
13	M12	4	1
14	M13	2.37	1
15	M14	2.00	1

Code 2nd metrics

Figure 4: Metrics result transform into database

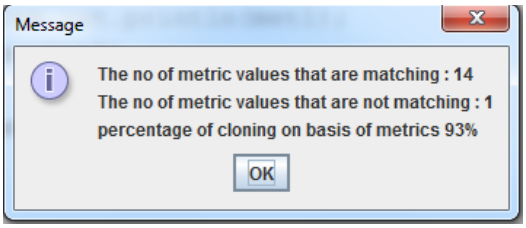


Fig 5: Result of Cloning on the basis of metrics

STAGE 2:

After calculating metrics and comparing them, once it is identified that the extent of cloning is good enough to go to next stage, then token based approach will be applied on thetwo source programs to calculate more precise code clones.In token based approach, firstly we have to form an intermediate representation of code in which we interchange the identifiers and keywords with some pre-defined tokens. Once the intermediate representation of the two source codes is accomplished, then we have to apply the comparison algorithm. A hashing based comparison algorithm has been developed to compare the two intermediate representations. The hash algorithm gives us good result with less time complexity because searching using hashing can be done in O(n). Figure 6 shows the intermediate representation for the two source codes. Finally, the percentage of cloning is reported as result as shown in figure 8. In next section, the algorithm for token based comparison is discussed.

A. Algorithm for implement hashingbased comparison

G. Take two programs and remove all types of comments in the program depending on the language of interest.

H. Also removes tabs, and new line(s) and other blanks spaces from the java program.

I. Then we have to perform an intermediate representation of programs in which we substitute all identifiers and keywords by some tokens.

J. Then, a hash value is calculated for every statement of intermediate representation of program and will be stored in a list. Note that, this hash function must be implemented in a way, such that the hash value returned must be different for different statements, and it must be same for two same statements.

Input: F1 (program in a text file)
Output: F2 (An intermediate representation for F1)
Begin:
Take two programs& ignore all kinds of comments in the F1 depending on the language of interest and remove blank lines from F1.

Store all keywords with their corresponding tokens in a 2-D array to map them later for our intermediate representation of F1.
Read F1 != end

a. Read each character one by one and store it in an array until space or new line encounter.

b. If the character is from the set {(,),:,;, . ,{ ,}, =} -----special characters.
Retrieve the string from the character array and trim it to remove spaces from string.

else ,
then store the character into the character array

c. Check whether the string is corresponding to any keyword

if yes,
put the token corresponding to the string into
the F2.
else ,
put "$" for that string into F2

Check whether the next character is ' ','\t' or '\n'
if yes,
put the character in F2
else ,
put "$" for that character into F2as a specialcharacter.
End of file,

Read F2 != end,
For each statement,
Calculate a hash value and store that hash value in a list.

End of file,
End

K. This algorithm produces two lists containing the hash values for each statement of the intermediate representation of the two programs.

L. Finally, the contents of two lists are compared to find out similarity among two codes. The algorithm is given in the table shown below.

IV. RESULTS

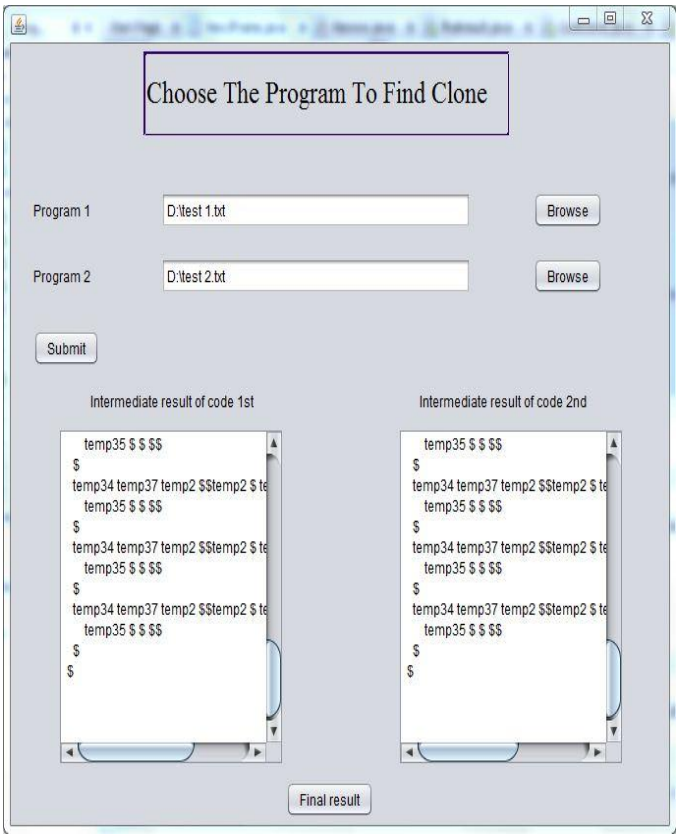


Figure6:Intermediate Stage for Clone Detection approach

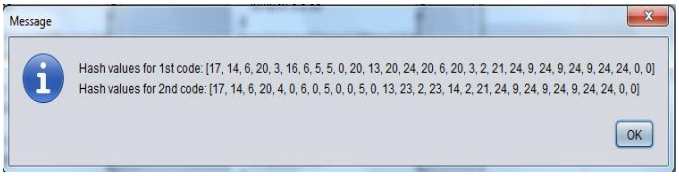


Figure7:Hash values returned from the intermediate stage

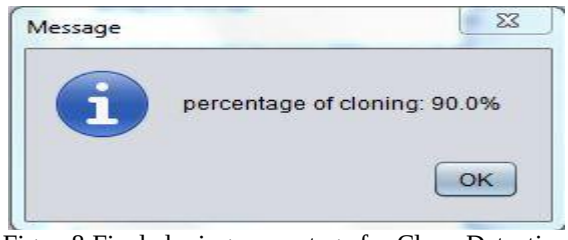


Figure8:Final cloning percentage for Clone Detection approach

V. RELATED WORK

In last decade many algorithm are proposed on software clone detection technique and every algorithm has its own

advantage and disadvantage. This unit describes the summary and overview of recent research in the areaof metric based software clone detection approach.

Y. Yuan et al. [19] proposed a count matrix based clone detection (CMCD) method, which is produced while counting the rate of frequencies of every variable in conditions specified by pre-determined counting condition. The projected technique is language-independent as it depends only on variable count. That is, if we have to count the rates of frequencies of variable in certain conditions with special standards, these standards are called as counting condition.Counting condition is used to select when the count should begin. The count matrix (CM) is a group of n count vectors (CV) and compares these Counting vectors with the help of Euclidean space. The variation between two vectors is calculated by the Euclidian Distance among them in the space, i.e.

$$D(v1, v2) = \|v1 - v2\|2 = \sqrt{\sum_{i=1}^{13} (v1i - v2i)\text{^}2}$$

The CMCD perform well in extracting count-based information and it is language independent. It supports to detect clone in large programs (> 1M LoC) also it has a abilities to perform well in scenario-based evaluation.

Vidhya et al. [20] proposed an emergent technique on java directories by using a metric based approach. The proposed system has been tested with two directories of JAVA files as input and the outcomes are produced based on the matching among files in directories. The percentage of the comparison is calculated by implementing the line by line comparison of the intermediate form of the files. This proposed technique merge both the textual based approach and metric based technique. Metric based approach isstraightforward hence it is a light weight method. The textual based approach is the one which give high exactness. This proposed technique also helps to notice the directory level cloning that is not structurally correlated but functionally similar.

K. Raheja et al. [4] proposed another approach using metric based technique for clone detection on byte code. Firstly the metrics are calculated from MCD Finder (java based) tool and then comparison technique is applied on these metrics to detect clone. MCD Finder tool works only for the Java language and it is easy to use. The metric based technique is used to identify the potential clone which does not directly work on source code. The proposed technique can also be merge with other techniques like abstract syntax tree based and the program dependence graph based technique to make this a hybrid method to proficiently detect semantic clones.

Zhuo Li et al. [3] proposed a technique, metric space based software clone detection by an iterative method. This technique transforms the main source code fragment into metric space member through the retrieved coordinate value, and then calculates similarity level through all members on their distance within the similar metric space. The nearer the two members are, the more similar they are, from code perception. As the distance between two members become lesser, then it means they are more similar and if the distance between them increases, it means they are less similar. This technique gives advantage like exactness and scalability with the help of metric space.

VI. CONCLUSION AND FUTURE WORK

The technique detects clones (type-1 and type-2) by metrics based approach for filtering code and after that it uses token based comparisons to detect code clone. The technique detects clones by hash algorithm in token based comparison to detect whether two clones really are clones of each other and it is also able to detect the type 3 clone near miss clone by using hash algorithm. The technique can also detect code plagiarism in student's computer lab programs.In future this approach can be integrated with other approaches like abstract syntax tree based approach and the program dependence graph approach to make this a hybrid approach to efficiently detect semantic clones.

REFERENCES

[1] Roy, C.K., and Cordy, J.R., "A Survey on Software Clone Detection Research" School of Computing TR 2007-541, Queen's University, 115 pp., 2007.

[2] "Software clone detection: A systematic review" Dhavleesh Rattan, Rajesh Bhatia, Maninder Singh. Information and Software Technology,Volume 55, Issue 7, July 2013, Pages 1165–1199.

[3] "An iterative, metric space based software clone detection approach" Zhuo Li ; Jianling Sun Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on Publication Year: 2010 , Page(s): 111- 116.

[4] "An Emerging Approach towards Code Clone Detection:Metric Based Approach on Byte Code", Kanika Reheja,Rajkumar Tekchandani,in IJARCSSE,Volume 3,Issue 5,May 2013.

[5] R. Wettel, R. Marinescu, "Archeology of code duplication: Recovering duplication chains from small duplication fragments", in: Proceeding of the 7th International Symposium on Symbolic and Numeric Algorithms for Scientific, Computing, 2005, p. 8.

[6] K. Roy, J.R. Cordy and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of computer programming, Vol. 74, No. 7, pp. 470-495, 2009.

[7] Amandeep Kaur, Mandeep Singh Sandhu , "Software code clone detection model using hybrid approach", in IJCT, Volume 3 No.2, OCT, 2012.

[8] C.K. Roy, J.R. Cordy, NICAD: "Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization", in: Proceeding of the 16th IEEE International Conference on Program Comprehension (ICPC'08), Amsterdam, The Netherlands, 2008, pp. 172–181.

978-93-85670-72-5 © 2016 (RTCSIT)

CONFERENCE PAPER
International Conference on
Recent Trends in Computer Science & Information Technology (RTCSIT-2016)
21st August 2016
Guru Nanak College Budhlada, Punjab India

[9] C.K. Roy, "Detection and analysis of near miss software clones", in: Proceeding of the 25th IEEE International Conference on Software Maintenance (ICSM'09), Edmonton, AB, 2009, pp. 447–450.

[10] S. Lee, I. Jeong, SDD:"High performance code clone detection system for large scale source code", in: Proceeding of the Object Oriented Programming Systems Languages and Applications Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA Companion '05), San Diego, CA, USA, 2005, pp. 140–141.

[11] Baker, "On finding duplication and near-duplication in large software systems", in: Proceeding of the 2nd Working Conference on Reverse Engineering (WCRE'95), Toronto, Ontario, Canada, 1995, pp. 86–95.

[12] Gehan M. K. Selim ,King Chun Foo, Ying Zou "Enhancing Source-Based Clone Detection Using Intermediate Representation", 17th Working Conference on Reverse Engineering, 2010.

[13] G. Anil Kumar, Dr. C. R. K. Reddy, Dr. A. Govardhan, "an efficient method-level code clone detection scheme through textual analysis using metrics", International Journal of Computer Engineering and Technology (IJCET) Volume 3, Issue 1, pp. 273-288 , January-June (2012).

[14] Jean-Francois Patenaude, Bruno Lagu¨e, "Extending Software Quality Assessment Techniques to Java Systems", Seventh International Workshop on Digital Object Identifier, pp. 45- 56, 1999.

[15] MooseGager,a Software Metrics Tool based on Moose Student Project Author Thomas B¨uhler October 2003 Supervised by: Dr. Michele Lanza Prof. Dr. Oscar Nierstrasz Institut f¨ur Informatik und angewandte Mathematik Universit¨at Bern.

[16] Kodhai.E, Perumal.A, and Kanmani.S, "Clone Detection using Textual and Metric Analysis to figure out all Types of Clones" International Journal of Computer Communication and Information System(IJCCIS)- Vol2. No1. ISSN: 0976–1349 July – Dec 2010.

[17] J. Mayrand, C. Leblanc, and E. M. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. In International Conference on Software Maintenance (ICSM), pages 244–253, 1996.

[18] Yoshiki Higo, Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue. Refactoring Support Based on Code Clone Analysis. In Proceedings of the 5th International Con- ference on Product Focused Software Process Improvement (PROFES'04), pp. 220-233, Kansai Science City, Japan, April 2004.

[19] Yang Yuan, Yao Guo, "CMCD: Count Matrix Based Code Clone Detection," apsec, pp.250-257, 2011 18th Asia-Pacific Software Engineering Conference, 2011.

[20] "Identifying Functional Clones Between Java Directories Using Metric Based System", Vidhya.K, Thirukumar. K, in IJARCSSE, Volume 3,Issue 8,ISSN: 2277 128X,May 2013.

978-93-85670-72-5 © 2016 (RTCSIT)

CONFERENCE PAPER
International Conference on
Recent Trends in Computer Science & Information Technology (RTCSIT-2016)
21st August 2016
Guru Nanak College Budhlada, Punjab India