The Hybrid Code Clone Detection Technique Using Genetic Algorithm and Artificial Intelligence (FFNN)

Gagandeep Kaur[1], Dr. Bikrampal Kaur[2]
[1]M.Tech Scholar, [2]Professor
CSE department, Chandigarh Engineering College, Landran

Abstract - Code Clone has an adverse influence on code quality and also code clones are most frequent problems that may appear in a software project. Code clones also have an impact on the effort of sustaining code that consequence in loss of time and money. Software systems are getting many unpredictable as the framework develops where keeping up such framework is an essential concern for the software commerce. Code clone is a particular type of the elements which are making software maintenance further challenging. The duplicate of code fragments and after that reuse by sticking with or without minor changes or adjustments and this type of reuse methodology of existing code is called code cloning and the pasted code fragment (with or without alterations) is recognized as a clone of the first one. The significant danger of cloning is that it increases the maintenance process. Cloning is fundamentally the method for software reclaim and also this becomes the important need of today's surroundings. That is the cause why code cloning has been widely utilized as a part of vast software businesses. In this thesis, utilization of feed forward neural network classifier for the detection of code clone. In the first step we use metrics based approach to identify code clone then later. In metric technique identify the function overloading, line of code. After feature extraction we, apply the genetic optimization means identify the reduction of the repeating functions and lines of codes. We use feed forward neural network classifier for classification and then evaluate it in some specific parameters like FAR, FRR and Accuracy. The whole stimulation has been taken place in MATLAB 2016a. From the results, it had been concluded that proposed technique are giving good results.

Keywords - Code clone, metric technique, feed forward neural network, performance metric and function overloading.

I. INTRODUCTION

The Code clone is one of the factors or computer program that makes software maintenance more difficult [1]. It is a code block of source files which is duplicate or similar to another code block. Code clones concept is use in various reasons such as reusing code by 'code-and-paste' and others which make the source files very difficult to change consistently. If the faults are found in one code of block then the entire cloned blocks are needs change or adjustment and it becomes extra difficult tasks to keep if the system becomes very large.so we are use different approach or techniques to detect clone in software. Like Token-Based, text based, metric based and every approach use a unique way to detect code clone in system. But the code clone has many benefit or limitation. In case of advantages we discuss the code clone use Code fragment proves its usability by coping and reusing numerous times in the system that can be combined in a library and announce its reuse potential officially [2]. Finding similar code may also useful in detecting plagiarism and copyright infringement [3]. And used for compact device by reducing the source code size. A part from the profits of code clones, it has more than one impact on the feature, reusability and maintainability of a software system. The following are the lists of some problems are having cloned code in a system. If the code section covers a bug and that section is reused through coping and pasting without or with small adaptations, the bug of the real section may remain in all the pasted sections in the system and therefore, the chance of bug propagation may increase significantly in the system . When increased trouble in a system then maintains or up gradation of system is more difficult. The reason of code cloning, clones does not occur in the software themselves. There are various reasons that tend the developer to do cloning [4]. A short description for some of the factors is discussed in time limit One of the major causes of code cloning is that a certain time limit is assigned to developer to finish a project. To do this developer just copy and paste the existing one and adapt to their current need. Language Limitation Clones can be presented due to the limitations of the language. Sometimes, the developers are forced to copy because of limitations of their knowledge in that particular programming language. Difficulty in Understanding Large System It is generally difficult to understand a big software system. These force the developers to use the example-oriented programming by adapting previously developed existing code. In this section I explained that the code clone definition, detection, reason and list of code clone detection. Section II, we examine the previous work of the code cloning. In Section III, examine that the code clone process and types of the cloning.Section IV defined that the proposed technique and flow of the proposed work with the help of data flow diagram explained. In section V and VI examine the result discussion and conclusion and future scope of the code clone.

II. RELATED WORK

ImanKeivanlooet, al 2015 [4] defined that code clones are unavoidable entities in software ecosystems. A variety of clone-detection algorithms are available for discovery code clones.). For Type-3 clone discovery at technique granularity (i.e., similar methods with changes in statements), dissimilarity threshold was one of the possible configuration parameters. In this paper propose a threshold-free approach to detect the Type-3 clones at method granularity across a large number of uses. Our approach uses an unsupervised learning algorithm, i.e., k-means, to determine true and false clones. **RidhiGarg et.al; 2014 [5]** discussed the Code clone detection was widely accepted and being carried out at the industrial level using a number of tools. But laterally with clone detection, clone management also becomes an imperative area of exploration. This paper discusses a method to rank the clone discovery results in order to manage them easily. **Chanchal Kumar Roy et.al 2007 [6]** described about the Code duplication or copying a code fragment and then reuse by pasting with or without any modifications of code smell in software maintenance. In this paper survey the state of the art in clone detection research. **Balwinder Kumar et.al 2015 [7]** described Code clones are the duplicated code which degrade the software quality and hence growth the maintenance cost. Recognition of clones in a large software system was very tedious tasks but it was needed to improve the design or structure and quality of the software products. So In this paper, various types of metric based clone detection approach and techniques are discussed. **DebarshiChatterji et.al 2013 [8]** defined that the Code clones are a common occurrence in most software systems. Their presence was believed to an effect on the maintenance process. So this paper describes an extended replication of a controlled experiment (i.e. a strict replication with an additional task) that analyses the properties of cloned bugs (i.e. bugs in cloned code) on the databaseconception of programmers.

III. PROCESS OF DETECTION CODE CLONE

Conservation is said to be the most exclusive part of software development. Code cloning not only results in raising the maintenance effort but also increases the prospect of errors in software's. Because of the difficulties caused due to code cloning, it is required to recognize code clones from software's. The ID process is known as clone detection.

a) *Pre-processing:* In the initial phase of clone detection process the objective source is distributed as well as comparison area is determined [9]. The principle targets to be considered in this phase are deciding the source units, evacuating uninteresting parts, and deciding the correlation unit. All the source code uninteresting to the comparison phase is filtered in this phase. After removing the uninteresting code, the remaining main code is partitioned into a set of disjoint fragments called source units.

b) *Transformation:* In the next stage, the source code's comparison unit is transformed to another intermediate internal representation of ease of comparison or for extracting the comparable properties. The transformation could be very simple by just removing the white space and comments [10] or could be very complex by generating PDG representation [11] or extensive source code transformation.

c) *Match Detection:* The transformed code is given as input to a suitable comparison unit, where it is compared with each other in order to find the match. The order of comparison units are used to sum up the adjacent similar units to form larger units. The output of the comparison unit is a list of matches with respect to the transformed code. These matches can be either the clone pair candidate or they have to be aggregated to form clone pair candidates. Then every clone pair is generally represented with the location information of the matched parts in the transformed code.

d) *Formatting:* In this stage, the clone pair list obtained with respect to the transformed code is then converted to a clone pair list obtained with respect to the original code base. This one is at that moment transformed into line numbers on the original source files.

e) *Post-pre-processing:* This stage helps out in filtering the false positives in two ways such as manual analysis and visualization tool.

f) *Manual Analysis:* once the first main code has been extricated, the underdone code of the clones of the clone couples imperilled to the specific manual analysis, with the intention of filtering out the entire possible false positive.

g) *Visualization:* To speed up the manual analysis in filtering out the false positives, visualization tool is used to visualize the clone pair.

In table 1 below described that to detect the clones to image out the problems and to help best software understandability and maintenance.

Concerning the detection of duplicated code, numerous methods have been successfully useful on industrial systems. These techniques can be roughly [12] classified into following categories:

Table 1: Techniques of Code Cloning

Technique Name	**Description**
String –based	separated into a number of strings
Token –based	lacer tool divides the program into a stream of tokens
Syntactic -based	to convert source program into parse trees
Parse-tree based	One performs design matching on the tree to search for same sub—trees.
Metric Based	Metrics are calculated from program and these are used to find duplicated code.
Hybrid –based	Combination of other clone detection techniques.

IV. METHODOLOGY

The proposed algorithm used in code clone i.e genetic algorithm used for optimization and FFNN used for classification.

A. Pseudo Code of Genetic Algorithm

Genetic algorithm is computer programs that simulator the processes of natural evolution in order to solve difficulties and to model evolutionary systems. Differenttypesofthreeoperators [13]:

a) Selection
b) Crossover and
c) Mutation

```
Input :
    a)  population size A
    b) Elitism rate B
    c) Mutation C
    d) Iterations/gens D
Output: Solution Y
//Definition
    1. Initialize A randomly solution feasible;
    2. Save all the random solution in the population popl;
    3. For i=1 to D do
    // elitisim
    Number of elitism ne=A.B;
    Select the positive outfit ne solutions in popl and save in popl;
    // Crossover
    Number of crossover Nc-(A-ne)/2;
    For j=1 to nc do
    Random selection two solutions X1 and X2;
    Save X3 and X4 to popl;
    End for
    //Mutation
    For j=1 tons do
    Select a solution Xi from popl;
    Mutation each bit of Xi with the feasible output by modifying Xj';
    End if
    Change Xj with Xj n popl;
    End for
    // Changing
    Change popl=popl1+popl2;
    End for
    // Refining the fit solution.
    Return the fit output x in popl;
```

B. Feed Forward Neural Network

The Multilayer layer feed forward neural network performs operation in two modes i.e. training and prediction [15]. The training of the MLF neural network and for the prediction of MLF neural network there is need of two data sets, training data set and testing data set for the prediction of accuracy. The training of feed forward neural network is mainly done using Back propagation algorithm [14].

In this work FFNN and metric based approach will be used for clone discovery. The whole implementation will take place in following manner:

- Input data
- Apply Metric based technique to get features extraction.
- After Feature extraction, we apply the optimization technique.
- Last one classification of code clones using Feed Forward Neural Network. For the prediction of code clone, data is collected & normalized. Then a single layer perception neural network is created and trained with the given dataset. After training, the network is tested by the testing dataset and it predicts whether the software project classes have the code clones or not.
- Evaluate the performance parameters like far, frr and accuracy.

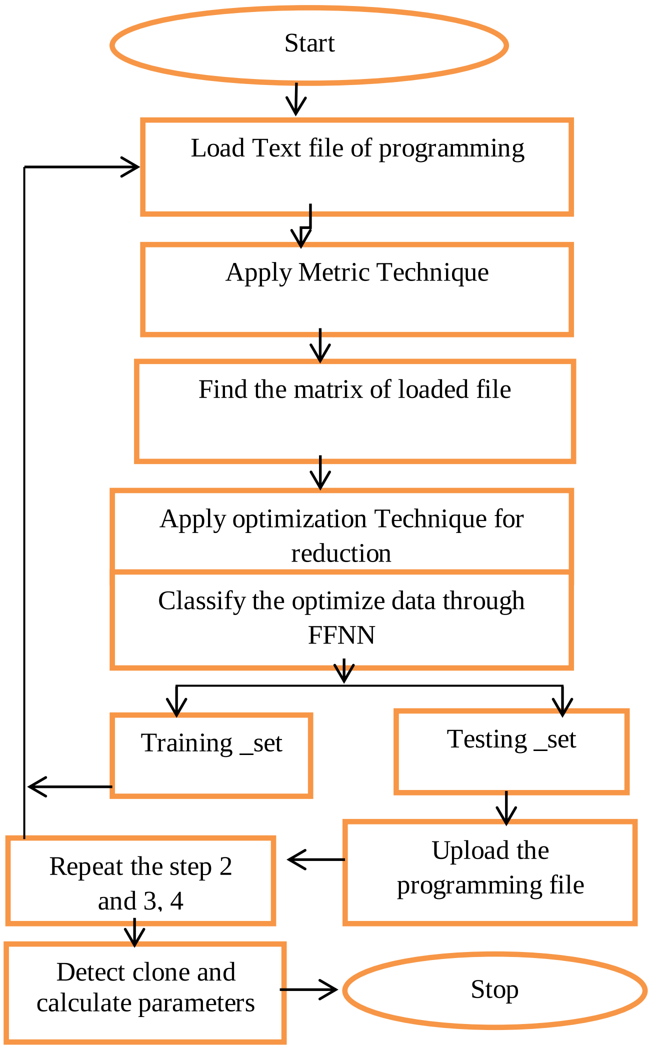
Start
Load Text file of programming
Apply Metric Technique
Find the matrix of loaded file
Apply optimization Technique for reduction
Classify the optimize data through FFNN
Training _set
Testing _set
Upload the programming file
Repeat the step 2 and 3, 4
Detect clone and calculate parameters
Stop

Fig.1 Proposed Flow chart

V. RESULT ANALYSIS

The main GUI of the code clone detection system in which metric based method and neural network has been used. This approach calculates metrics from source code and uses these metrics to measure clones in software. Rather than working on source code directly this approach use metrics to detect clones. After this re-checking will be done using NN classifier. Then Result evaluation will takes place.

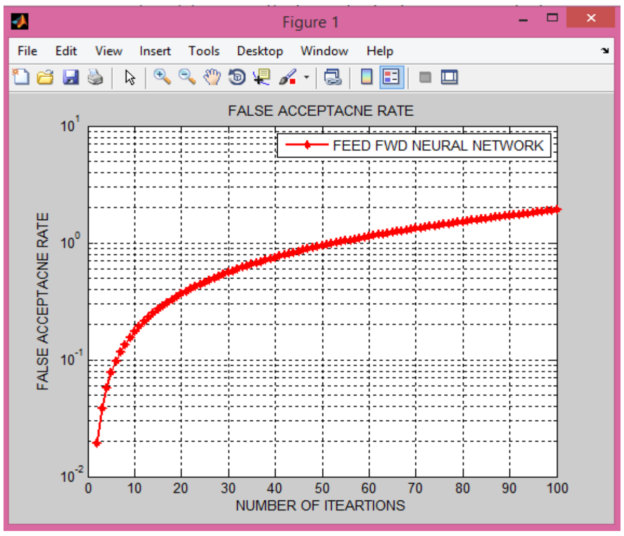

Fig.2 False Acceptance Rate –FFNN

A system's FAR typically is stated as the ratio of the number of false acceptances divided by the number of identification attempts. Same here, plot a graph which uses the FAR parameter for the proposed approach.

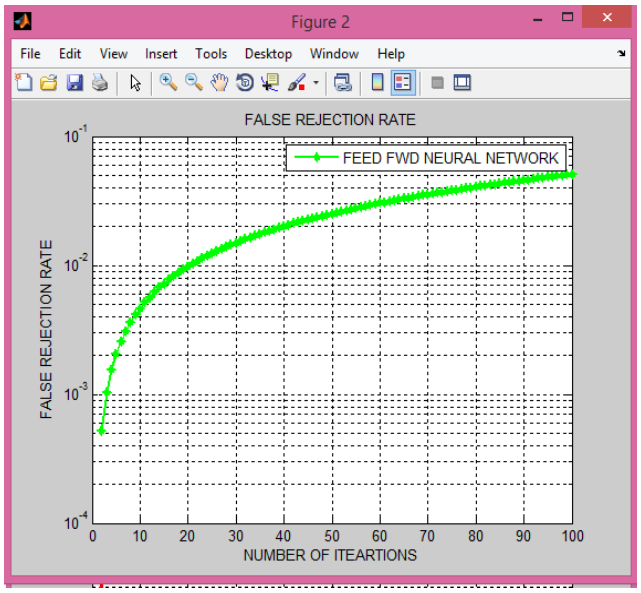

Fig.3 False Rejection Rate –FFNN

A system's FRR typically is stated as the ratio of the number of false rejections divided by the number of identification attempts. Above figure shows the rate of FRR for proposed approach.

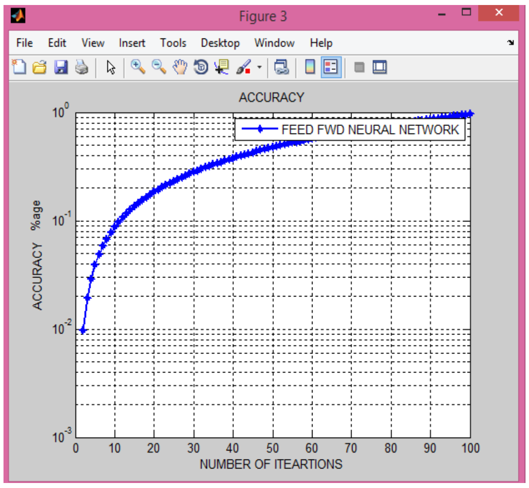

Fig.4 Accuracy-FFNN

Figure defines the accuracy description of the random errors, measures of statistical variability. *Accuracy* is how close a measured value is to the actual (true) value. Above figure shows the accuracy value for proposed method and it has been clearly seen that accuracy for proposed method is good.

Table no: 2 Comparisons between Accuracy (FFNN and ANN)

Category	Accuracy (Existing)	Accuracy(Proposed)
Code 1	96	97
Code 2	95	96.7
Code 3	90	97.8

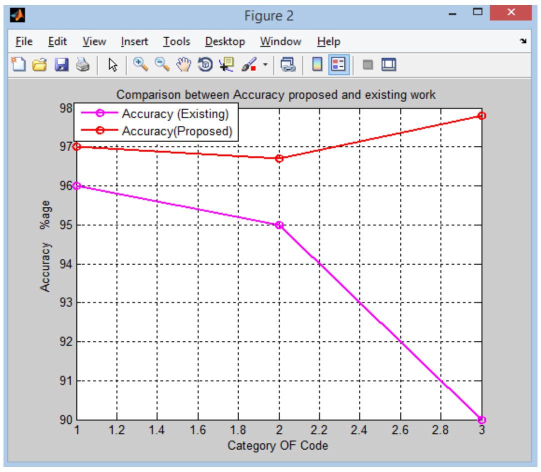

Fig.5 Comparison between Accuracy proposed and existing work

In this figure represents the comparison between proposed and existing work. In feed forward technique improve the accuracy parameters. *Accuracy* is how close a measured value is to the actual (true) value. Above figure shows the accuracy value for proposed method and it has been clearly seen that accuracy for proposed method is good.

Table no: 3 Comparisons between False Acceptance Rate (FFNN and ANN)

Category	FAR(Existing)	FAR(Proposed)
Code 1	0.06	0.04
Code 2	0.08	0.06
Code 3	0.013	0.067

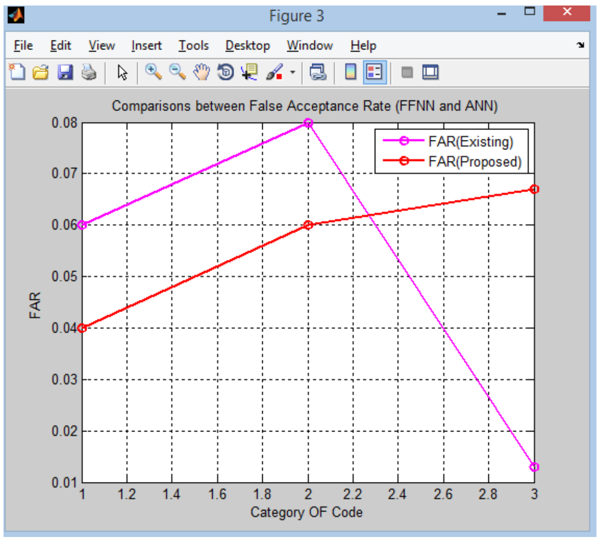

Fig.7 False Acceptance Rate comparison with Existing and Proposed Work

In this figure shows the comparison existing and proposed work, false acceptance rate means positive data find using classification in the testing Module and Extract the unique Features. The false acceptance rate identifies the value is the acceptable error is 0.0489. The False Acceptance rate (FAR) is the probability that the system incorrectly authorizes a non-authorized code, due to incorrectly matching the code input with a template.

Table no: 4 Comparisons between False Rejection Rate (FFNN and ANN)

Category	FRR(Existing)	FRR(Proposed)
Code 1	0.05	0.05
Code 2	0.07	0.058
Code 3	0.08	0.06

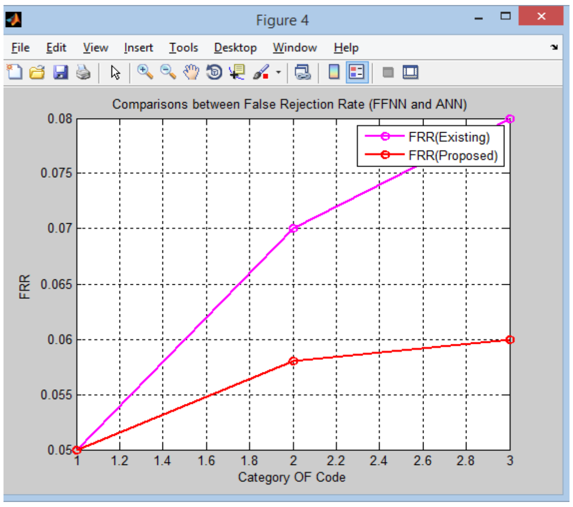

Fig.7 Comparison between FRR (existing and Proposed) Work

Figure shows comparison between Existing and Proposed Work, the false rejection rate (FAR) means negative data collect using Feed forward neural network (FFNN) for classification and feature identifies the scale invariant feature transform. The false rejection rate (FAR) compute the value is 0.05.

VI. CONCLUSION AND FUTURE SCOPE

All the advantages and disadvantages of various approaches discussed but it clearly shows that no one method is able to find the clones correctly. All the methodologies talked about above gives 75-85% exactness in recognition and examination of code clones however nobody methodology has the capacity discover all clones more than this precisely. So it is presumed that metric based system utilizing neural system give more precise results when contrasted with different methods. And infer that the metric based clone discovery methodology utilizing neural system is exceptionally compelling approach as it found the clones furthermore helps in recognizing the clones of every sorts. Utilizing metric based system different elements has been extricated like no. of functions, private functions, public functions, function overloading. Future scope lies in the use of abstract syntax based approach in combination with neural network or SVM classifier.

VII. REFERENCES

[1]. Lakhotia, Arun, Junwei Li, Andrew Walenstein, and Yun Yang. "Towards a clone detection benchmark suite and results archive." In *Program Comprehension, 2003. 11th IEEE International Workshop on*, .vol.no.11, pp. 285-286, IEEE, 2003.

[2]. Baker, Brenda S. "On finding duplication and near-duplication in large software systems." In *Reverse Engineering, 1995., Proceedings of 2nd Working Conference on*, Vol.no.95,pp. 86-95.,IEEE, 1995.

[3]. C .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. May 2009.

[4]. Keivanloo, Iman, Feng Zhang, and Ying Zou. "Threshold-free code clone detection for a large-scale heterogeneous Java repository." In 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 201-210. IEEE, 2015.

[5]. Garg, Ridhi, and RajkumarTekchandani. "An approach to rank code clones for efficient clone management." In Advances in Electronics, Computers and Communications (ICAECC), 2014 International Conference on, pp. 1-5. IEEE, 2014.

[6]. Roy, Chanchal Kumar, and James R. Cordy. "A survey on software clone detection research." Queen's School of Computing TR 541, no. 115 (2007): 64-68.

[7]. Kumar, Balwinder, and Satwinder Singh. "Code Clone Detection and Analysis Using Software Metrics and Neural Network-A Literature Review." Complexity 1, no. 2 (2015): 3.

[8]. Chatterji, Debarshi, Jeffrey C. Carver, Nicholas A. Kraft, and Jan Harder. "Effects of cloned code on software maintainability: A replicated developer study." In WCRE, pp. 112-121. 2013.

[9]. Balazinska, Magdalena, Ettore Merlo, Michel Dagenais, Bruno Lague, and Kostas Kontogiannis. "Measuring clone based reengineering opportunities." In *Software Metrics Symposium, 1999. Proceedings. Sixth International*, vol.no.5, pp. 292-303. IEEE, 1999.

[10].T. Kaniiya. 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. 654-670. July 2002.

[11].Ducasse, Stéphane, Matthias Rieger, and Serge Demeyer. "A language independent approach for detecting duplicated code." In *Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on,* Vol.no.1, pp. 109-118., IEEE, 1999.

[12].Tairas, Robert, and Jeff Gray. "Phoenix-based clone detection using suffix trees." In *Proceedings of the 44th annual Southeast regional conference*, Vol.no.8, pp. 679-684. ACM, 2006.

[13].Patenaude, J-F., Ettore Merlo, Michel Dagenais, and Bruno Laguë. "Extending software quality assessment techniques to java systems." In *Program Comprehension, 1999. Proceedings. Seventh International Workshop on*, vol.no.3, pp. 49-56., IEEE, 1999.

[14].Krinke, Jens. "Identifying similar code with program dependence graphs." In *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, vol.no.4, pp. 301-309. IEEE, 2001.

[15].Marcus, Andrian, and Jonathan I. Maletic. "Identification of high-level concept clones in source code." In *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, vol. 2, pp. 107-114. IEEE, 2001.