Detection of Code Clone Functions Extract Using Metric Based Technique in Software Engineering

Gagandeep Kaur[1], Dr. Bikrampal Kaur[2]
[1]M.Tech Scholar, [2]Professor
Department of CSE, Chandigarh Engineering College, Landran, Punjab, India

Abstract-- Object-oriented programming is today the main paradigm in mainstream software development. As their requirement is increasing day by day they are becoming greater and complex. Large scale software organizations are expensive to build and, are even more expensive to maintain. In PC software, we could probably have different sorts of repetition.Duplicated code proves easy and inexpensive during the software expansion phase, but it makes software maintenance much harder. Software clone has a number of undesirable effects on the quality of the software. So there is a need to detect the clones to figure out the problems and to help better software understand ability and maintenance. This paper proposes a various method that association's neural network with metric based technique to yield structurally meaningful near-miss clones and implemented using MATALB. It is a new clone detection method that has been shown to yield gets high precision and high recall in detecting near-miss intentional clones.

Keywords-- Object-Oriented Programming, Maintenance, Software clone and fragments.

I. INTRODUCTION

Software Engineering is the request of engineering to the development, testing, design, implementation and maintenance of the software in an efficient technique [1]. The design, develop and testing window system level software, compilers and network division software for industrial, military and business etc. Code cloning is a form of software reuse, and exists in virtually every software project. This ad-hoc form of reuse consists in copying, and eventually modifying, a block of present code that device a piece of required functionality. Duplicated blocks are called clones and the act of replication, including slight modifications, is said cloning. The consequences of several studies indicate that a considerable fraction of the basis code in large software systems is duplicate code. Software clone is usually generated by programmer's copy and paste actions. Programmers habitually copy and paste an existing similar code and further modify it according to their need. Code duplicating or the act of copying code wastes and making minor, non - functional alterations, is a well-known problematic for evolving software [2] systems leading to replicated code fragments or code clones. Code cloning also leads to difficulty in code maintenance. Duplicate code also leads to complexity when some enhancement or modification is going to be done. Code detection is very important in software industry due to following reasons:

- Plagiarism detection
- Code mining
- Copyright Protection and ;
- Code Compaction[3]

Over the last years many techniques has been recommended for code cloning. In this paper, code cloning optimization will be done using genetic algorithm in addition with metrics based technique to enhance the accuracy of code cloning system [4].

Duplicated code shows relaxed and inexpensive during the software development phase, but it kinds software maintenance [5] much harder. Software clone has a number of undesirable effects on the excellence of the software. Also increasing the quantity of the code, which requirements to be preserved, it also increases the bug probability.

So there is a need to detect the clones to symbol out the difficulties and to help better software understandability and keep. Regarding the detection of duplicated code, frequent techniques have been positively functional on industrial systems. These practices can be roughly classified into following categories define in table 1:

This algorithm will find out various types of code like type-1, type-2 etc. The remainder of the paper is organized as Section 2, 3 will discuss the proposed techniques basic concept. Section 4 will discuss the proposed work methodology. Section 5 contains the results and analysis. Finally section 6 contains the conclusion.

The work is scheduled as gives Introduction and types of the code clone in section I; a study of the literature of these approaches and techniques for to improve the performance and to analyses and detect the code module in section II. It used for proposed algorithm or simulation is described in section III succeeded by the research technique. Here discussed the problem formulation in section IV, The evaluation of performance parameters and consequences in section V followed by the conclusion and future scope in section VI.

Table no: 1 Types of the Detection of the Code Clone

Sr no.	Technique Name	Description
1.	String Based	The platform is divided into a number of strings (typically lines) and these sequences are equaled against each other to find structures of duplicated strings.
2.	Token-based	A lacer tool divisions the program into a watercourse of tokens and then examines for series of similar tokens.
3.	Syntactic-based	Approaches use a parser to translate source program into explain trees or abstract syntax tree matching or metrics to find clones.
4.	Parse-tree based	After construction a wide-ranging parse-tree one performs pattern matching on the tree to search for similar sub-trees
5.	Metric-based	Metrics are calculated from program and these are used to find duplicated code.

II. RELATED WORK

In this we, discuss the prior work of the code clone detection based on software engineering. Jian Chen et.al ,2015 [6] In this paper, examined the use of a clone sensor to classify known Android malware. They assemble a set of Android submissions known to comprise malware and a set of kind applications. They extracted the Java source code from the double code of the submissions and use NiCad, a near miss clone detector, to invention the classes of clones in a small separation of the malicious presentations. Then used these clone programs as a signature to find related source files in the rest of the hateful applications. The benign gathering is used as a control group. Mr. Ritesh V. Patil et.al,2014[7] examined existing code in software development life cycle. Although code cloning is a suitable way for designers to reuse current code it could possibly lead to negative influences, such as code size needlessly increased and may lead to unused, dead code. There are numerous clone detection techniques based on dissimilar evaluation parameters. Exposed clone detection tools and methods do not sufficiently satisfy with regards to rapidity and correctness. Ritu Garg et.al,2014[8] This paper offered a brief impression to the detection of these risk and contradictions in either of the two stages of software development system i.e. Design phase or the operation phase along with their experts and frauds. Ritesh V. Patil et.al,2014 [9] described as, the clone discovery consequences for a single source code variety gives a developer with particulars about a discrete state in the development of the software system. However, tracing clones through numerous source code versions enables a clone investigation to take into replication a temporal dimension. This nice of an investigation of clone evolution may be utilized to find out the outlines as well as features displayed by clones as they evolve within a system. Developers may apply the consequences of this analysis to recognize the clones more methodically, which may guide them to handle the clones more automatically. Later, studies of clone development provide significant role in observing as well as handling disquiets of cloning in software. Harald Störrle et.al, 2015 [10] described as, Code Duplicates are a main source of software faults. Thus, it is probable that model duplicates have a significant adverse impact on model excellence, and thus, on any software shaped based on those models, notwithstanding of whether the software is

made fully automatically or hand crafted following the drawing defined by the model. Inappropriately, however, model clones are much less well deliberate than code clones. In this paper, presented a clone detection process for UML domain models. A method covers a much better variety of model types than present approaches while providing high clone detection rates at high speed.

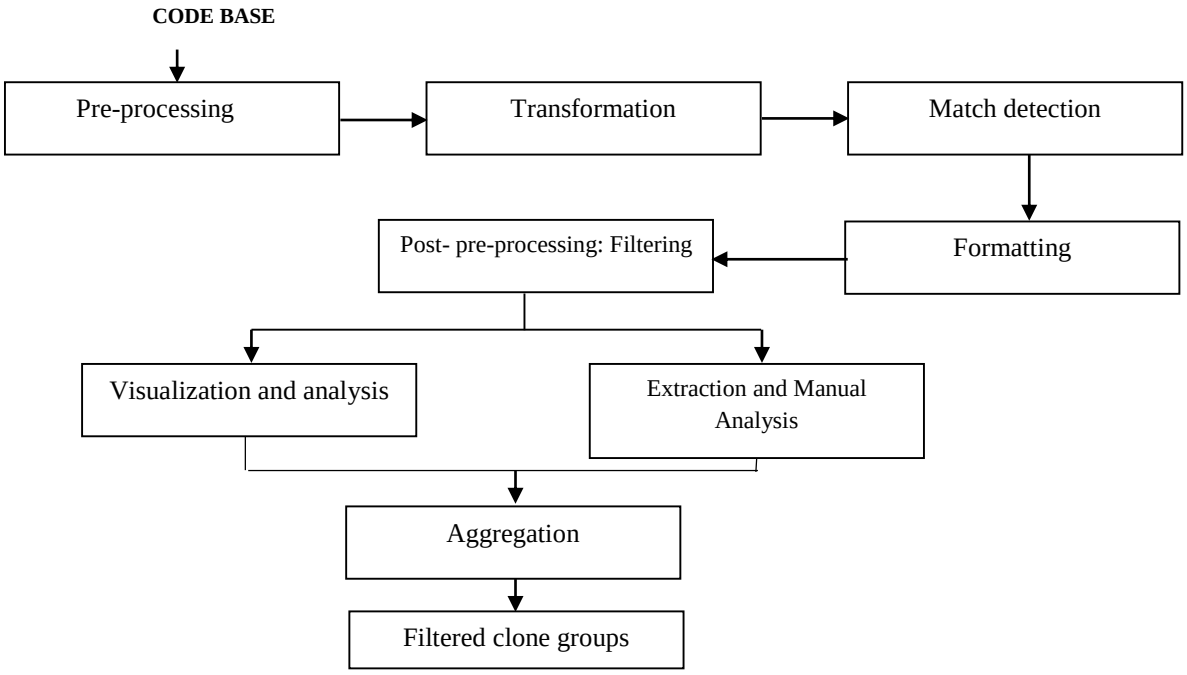

Fig.1: Code Clone Process

III. ISSUES IN CODE CLONE DETECTION

An increasing the amount of the code which needs to be maintained duplication also increases the defect probability and resource requirements. The following list gives an overview of these problems:

- Increased Maintenance Work and Cost Because of duplicated code in the system. One needs additional time and attention to understand the existing code. When programmers maintain a piece of clone code. The changes should also perform on every other clone pairs. Since programmers who usually have no records of this duplicate code, the maintaining work should perform on the entire system. If a cloned code segment is found to be contained a bug, all of its similar counterparts should be inspected for improving the bug in question, as there is no guarantee that this bug has been already eliminated from other related parts at the time of reprocessing or during maintenance [11].
- Increased Defect Probability By simply copying a quantity of code into a new context, which will cause the fight between each other, e.g. conflict and clash between variables from the unoriginal code and variables in the new context. Dependencies of copied code may also not be fully understood by the new context is mother potential defect cause. Duplication of the source code also increases the probability of bug propagation in the system [12].

Various Code Clone detection Techniques each have its own Pros and cons:

- Certain methods find duplicates by associating program script with small or no code normalize and some other methods use a negligent to variety a token sequence for program and find clones by definition common sequence on the token sequence and some make use of parse to build a parse-tree or abstract syntax tree.
- Certain other Methods evaluate some metrics and find duplicity by comparing the build program dependency graphs and find clones.
- As it has been already discussed that a lot of tools, techniques and classifiers has been already tried in this scenario of textual parameter code cloning detection but there are chances of improvement of the accuracy pattern of classification [12].
- The problem of this research work is to enhance the accuracy and FAR rate as well as to reduce the[15] FRR rate of the detection using Soft Computing in addition to this using metric based technique for feature extraction of codes.

IV. PROPOSED WORK

In Section, we discuss the proposed techniques used in Our research Work and Compute the performance parameters i.e Mean Square error, False Acceptance Rate, False Rejection rate and Accuracy. We mainly work in Found the Clone of the MATLAB code and calculate the Accuracy in Manually and Auto Clone. In this work FFNN and metric based approach will be used for clone detection. The whole implementation will take place in following manner:

- Input data: Firstly, we create the dataset in Object Oriented programming Language and MATLAB code.
- Feature Extraction: Apply Metric based method to get features extraction. In Metric based technique- instead of associating the code straight. Different metric of code are collected and these metrics were compared to perceive clones. Many clone uncovering procedures today use metrics for perceiving similar codes. Initially, fingerprinting functions which are nothing but a set of software metrics are calculated for one or more syntactic units such as a function or a class, a method or even a statement and then these metrics values are compared to find clones over these syntactic units. Generally, such metrics are calculated by parsing the source code into ASTFPDG representation. Then the metric were calculated from names, layout, expression and simple control flow of function. A clone is detected only when pair of whole function bodies that have similar metrics values are identified.
- Optimize: After Feature extraction, we apply the optimization technique. The Genetic Procedure is a model of machine knowledge which derives its performance from image of the processes of Evolution in environment. This is done by the creation within a machine of a Populace of Individuals represented by Chromosomes, in spirit a set of character strings that are similar to the base-4 chromosomes that we see in our own DNA. The individuals in the populace then go through a process of evolution. We should note that Evolution is not a purposive or directed process. That is, there is no evidence to support the declaration that the goal of evolution is to produce Mankind. Indeed, the procedures of nature seem to boil down to different Individuals competing for resources in the Environment. Some are healthier than others. Those that are better are more likely to survive and disseminate their genetic material.
- Classification: Last one classification of code clones using Feed Forward Neural Network. For the prediction of code clone, data is collected and normalized. Then a single layer perception neural network is created and trained with the given dataset. Feed Forward Neural Network is an organically stimulated organization algorithm. It consists of amount of simple neuron like processing units, prearranged in layers. Every unit in a layer is related with all the units in the preceding layer [11]. These connections are not all equal: each joining may have a different strength or weight. The weights on these contacts encode the information of a network. Frequently the units in a neural network are also called nodes. Data arrives at the inputs and permits through the network, layer by layer; pending it arrives at the productivities. During consistent operation, that is when it acts as a classifier, there is no comment between layers. This is why they are called feed forward neural networks.
- After training, the network is tested using 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.

V. PERFORMANCE PARAMETERS

- Mean Square Error : the mean squared error or mean squared deviation of an estimator measures the average of the quadrangles of the errors or deviations, that is, the alteration between the estimator and what is appraised.
- Accuracy: It is used to describe the closeness of a measurement to the true value. When the term is practical to sets of quantities of the same measured, it involves a component of random error and a component of systematic error.
- False Acceptance rate: is the probability that the system incorrectly authorizes a non-authorized person, due to inaccurately matching the biometric input with a template. The FAR is normally expressed as a percentage, following the FAR characterisation this is the percentage of invalid inputs which are incorrectly accepted.
- False Rejection Rate: is the probability that the system incorrectly rejects access to an authorized person, due to deteriorating to match the biometric input with a model. The FRR is normally expressed as a percentage, following the FRR explanation this is the percentage of valid inputs which are incorrectly rejected.

VI. CONCLUSION

Cloning of code has become one of the easiest ways to complete a project, who does not want to invest their time on doing programming their project. It's a loss for those who really works hard for the project coding. The date no such method has present who can evaluate the cloning for several languages with one piece of code. The purpose research work has overcome the drawbacks of the previous attempts by removing the bar of the language which follows the architecture of C++. The results have been verified using FEED FORWARD BACK PROPAGATION NEURAL NETWORK over the metrics. We will explain the results in further paper.

REFERENCES

[1] Störrle, Harald. "Effective and efficient model clone detection." In Software, Services, and Systems, pp. 440-457. Springer International Publishing, 2015.

[2] 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, pp. 109-118. IEEE, 1999.

[3] Schuldt, Christian, Ivan Laptev, and Barbara Caputo. "Recognizing human actions: a local SVM approach." In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, vol. 3, pp. 32-36. IEEE, 2004.

[4] Kamiya, Toshihiro, Shinji Kusumoto, and Katsuro Inoue. "CCFinder: a multilinguistic token-based code clone detection system for large scale source code." IEEE Transactions on Software Engineering 28, no. 7 (2002): 654-670.

[5] Ahn, Chang Wook, and Rudrapatna S. Ramakrishna. "A genetic algorithm for shortest path routing problem and the sizing of populations." IEEE transactions on evolutionary computation 6, no. 6 (2002): 566-579.

[6] Chen, Jian, Manar H. Alalfi, Thomas R. Dean, and Ying Zou. "Detecting Android Malware Using Clone Detection." Journal of Computer Science and Technology 30, no. 5 (2015): 942-956.

[7] Wyss-Coray, Anton, Thomas A. Rando, Markus Britschgi, Kaspar Rufibach, and Saul Abraham Villeda. "Biomarkers of aging for detection and treatment of disorders." U.S. Patent Application 13/575,437, filed January 28, 2011.

[8] Garg, Ritu, and Rajesh Bhatia. "Code Clone v/s Model Clones: Pros and Cons." International Journal of Computer Applications (IJCA) 89, no. 15 (2014): 20-22.

[9] Patil, Ritesh V., Lalit V. Patil, Sachin V. Shinde, and S. D. Joshi. "Software code cloning detection and future scope development-Latest short review." In Recent Advances and Innovations in Engineering (ICRAIE), 2014, pp. 1-4. IEEE, 2014.

[10] Roy, Chanchal K., James R. Cordy, and Rainer Koschke. "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach." Science of Computer Programming 74, no. 7 (2009): 470-495.

[11] Baker, Brenda S. "On finding duplication and near-duplication in large software systems." In Reverse Engineering, 1995., Proceedings of 2nd Working Conference on, pp. 86-95. IEEE, 1995.

[12] Yang, Jiachen, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki, and Shinji Kusumoto. "Classification model for code clones based on machine learning." Empirical Software Engineering 20, no. 4 (2015): 1095-1125.