RESEARCH ARTICLE OPEN ACCESS

Code Clone Detection and Analysis Using Software Metrics and Neural Network-A Literature Review

Balwinder Kumar [1], Dr. Satwinder Singh [2]
Department of Computer Science Engineering and Information Technology [1] & [2]
Baba Banda Singh Bahadur Engineering College
Fatehgarh Sahib and Punjab Technical University Jalandhar
Punjab - India

ABSTRACT

Code clones are the duplicated code which degrade the software quality and hence increase the maintenance cost. Detection of clones in a large software system is very tedious tasks but it is necessary to improve the design, structure and quality of the software products. Object oriented metrics like DIT, NOC, WMC, LCOM, Cyclomatic complexity and various types of methods and variables are the good indicator of code clone. Artificial neural network has immense detection and prediction capability. In this paper, various types of metric based clone detection approach and techniques are discussed. From the discussion it is concluded that clone detection using software metrics and artificial neural network is the best technique of code clone detection, analysis and clone prediction.

Keywords:- code fragment, code clone, clone detection approach, object oriented metrics, neural network, precision, recall etc.

I. INTRODUCTION

Code clones are similar segment of the source code which may be inserted either by mistake or knowingly. Reusing code fragments by copying and pasting with or without minor adaptation is a common activity in software development. But the presence of these code clones may decrease the design structure and software quality like readability, changeability and maintainability and hence increase the maintenance cost [23]. So the detection of code clones is necessary in the software products. There are various techniques are proposed over the last decade for the identification and prediction of code clones.

CODE CLONE DETECTION TECHNIQUES AND APPROACHES -

There are mainly four types of code clone detection techniques [22]. Table 1 gives the classification of code clone and its techniques.

1) ***Textual approach****:* Textual approaches are the text-based approach which uses little or no transformation on the source code before the actual comparison, and in most cases raw source code is used directly in the clone detection process. For example –SSD, NICAD etc.

2) ***Lexical approach:*** Lexical approaches are the token-based approach which begin by transforming the source code into a sequence of tokens and then scanned for duplicated sub sequences of tokens and the corresponding original code is returned as clones. Examples: Dup, CCFinder , CPMiner etc.

3) ***Syntactic approaches***: Syntactic approaches use a parser to convert source programs into parse trees or abstract syntax trees (AST) which can then be processed using either tree matching or structural metrics to find clones. Examples: CloneDr, Deckard, CloneDigger etc.

4) ***Semantic approaches***: Semantics approach uses static program analysis to provide more precise information than simply syntactic similarity. In some approaches, the program is represented as a program dependency graph (PDG). The nodes of this graph represent expressions and statements, while the edges represent control and data dependencies. Examples: Duplix, GPLAG etc.

5) ***Metric-based approach:*** Metric based approaches gather a number of metrics for code fragments and then compare metrics vectors rather than code or ASTs directly. One popular technique involves fingerprinting functions, metrics calculated for syntactic like a class, function, method or statement that provides values that can be compared to find clones of these syntactic units. In most cases, the source code is first parsed to an AST or CFG (control flow graph) representation to calculate the metrics. Metrics are calculated from names, layout, expressions and control flow of functions.[4] A clone is defined as a pair of whole function bodies with similar metrics values. Metrics-based approaches have also been applied to find duplicate web pages and clones in web documents.

6) ***Hybrid approaches:*** In addition to the above discussed, there are some clone detection techniques for Lisp-like languages. It provides a hybrid approach that combines syntactic techniques (using metrics) and semantic techniques (using call graphs) in combination with specialized comparison functions. Yogita Sharma et. al [23] used text based and metric based technique to identify the clones in c or c++ source code.

TABLE I
CLASSIFICATION OF CODE CLONE AND TECHNIQUES [22]

	Text based	**Token based**	**AST based**	**PDG based**
Category	Textual approach	Lexical approach	Syntactic approach	Semantic approach
Clone Type	Type-1	Type-1,2	Type-1,2,3	Type-1,2,3
Complexity	O(n)	O(n)	O(n)	$O(n^3)$
Meaning of n	Lines of code	No. Of token	Nodes of AST	Node of PDG

II. RELATED RESEARCH

The literature review of code clone detection and analysis begins with a basic concept of clone detection terminology.

A. Code Fragment (CF)

A code fragment (CF) is any sequence of code lines with or without comments. It can be of any granularity level for example function definition, begin-end block, or sequence of statements. A CF is identified by its file name and begin-end line numbers in the original code base and is denoted as a triple (CF.FileName, CF.BeginLine, CF.EndLine).

B. Code Clone

A code fragment CF2 is a clone of another code fragment CF1 if they are similar or identical by some given definition of similarity, that means, 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

Code fragments are of two main types either based on textual similarity or functional similarity. The first type of clone is often the result of copying a code and paste code fragment into another location. In the following the types of clones based on both the textual (Types 1 to 3) [11] and functional (Type 4) similarities are described:

Type-1: Identical code fragments except for variations in whitespace, layout and comments.

Type-2: Syntactically identical fragments except for variations in identifiers, literals, types, whitespace, layout and comments.

Type-3: Copied fragments with further modifications such as changed, added or removed statements, in addition to variations in identifiers, literals, types, whitespace, layout and comments.

Type-4: Two or more code fragments that perform the same computation but are implemented by different syntactic variants.

D. Precision and Recall

These are the two terms used when discussing the characteristics of the candidate code clones returned by a clone detector. Precision refers to the quality of the candidates returned by the detection method: high precision indicates the candidate code clones are mostly correctly identified as code clones and low precision indicates the candidate code clones contain many candidates that are not actual code clones. Recall refers to the overall percentage of artifacts that exist in the source code that have been detected by the clone detector: high recall indicates most of the code clones in the source code have been found, low recall indicates most of the code clones in the source code have not been found.[2]. When comparing code clone detection techniques, precision and recall are often referenced as measures of the accuracy and completeness of the candidate code clones.

$$\text{Precision (P)} = \frac{\text{Number of clones correctly found}}{\text{Total number of clones found}}$$

$$\text{Recall (R)} = \frac{\text{Number of clones found correct}}{\text{Total number of clones found}}$$

The detection of code clones is a two phase process which consists of a transformation and a comparison phase. In the first phase, the source text is transformed into an internal format which allows the use of a more efficient comparison algorithm. During the succeeding comparison phase the actual matches are detected. Due to its central role, it is reasonable to classify detection techniques according to their internal format.

Roy and Cordy [1] did comparison of different techniques of clone detection such as textual approach, lexical approach, semantic approach and metric based approach and also comparing and evaluating clone detection tools such as Duploc, simian and NICAD. They proposed that NICAD tool is the best among all others. Moreover they explain four category of clone viz-Type-1, Type-2, Type-3 and Type-4.

Roy and Cordy [2] also survey the state of the art in clone detection research. Firstly they describe the clone terms commonly used in the literature along with their corresponding mappings to the commonly used clone types. Secondly they give the review of existing clone detection approaches and techniques.

Gayathri et.al [3], detect the different types of clones using different algorithm like textual analysis, metric based distance algorithm and mapping algorithm. The detected clones are-extract clone, renamed clone, gapped cloned and semantic clone. They used clone detection and metrics to

evaluate quality. Then they discuss several approaches used in clone detection. Metric based clone detection approach uses the metric based distance algorithm. Then they compared different types of approach using different algorithm and calculate their metrics, speed, cost and quality.

J. Mayrand et.al [4] present metric based technique to detect functional clones automatically from source code of any language. They developed a "DATRIX" tool framework which is a source code analyser [4] which converts the source code into some Intermediate Representation Language. Out of which only control flow metrics and data flow metrics were selected because they provide internal characteristics information about functions. For automatic detection, this proposed approach experimented on 2-Telecommunication monitoring system in which for finding function clones, 4 points of comparison is performed which are- name of function, layout of function, expression in function and control flow.

Pavitdeep et.al [5], developed a tool "Software Quality assurance tool" in dot net framework using C# as programming language. This tool generates the software code metrics for C# projects using the AST technique. This tool works at method and class level metrics. This tool also detects the clones of Type-1 and Type-2.

They also compared different types of tools [5] with each other and explaining their merits and demerits of each tool. The tool of software quality assurance predicts and calculated the metrics of modern languages like c# using the technique of abstract syntax tree (AST).

Sandeep Sharawat [6] uses the neural network technique for the prediction of maintainability. The object oriented metrics like DIT, NOC, SIZE, WMC, RFC, NOM, LOCM etc are used for the computation of maintenance index which is composite metric that incorporate a number of traditional source code metrics into a single number that indicate relative maintainability. Matlab is used for the implementation of project. The training data are collected from Li-henry dataset. Neural network created and various training algorithm are applied on the tested data to find the minimized error like trainlm, traingdm, trainscg, trainr, trainrp etc. Trainlm is found to be the best algorithm for the prediction of maintainability.

k.k Aggarwal et.al [7] uses ANN technique to predict the maintenance effort of the classes. The inputs to the network were all the domain metrics P1, P2 and P3 [7]. The network was trained using the back propagation algorithm. Table II [7] shows the best architecture and table IV [7] shows the MARE, MRE, r and p-value results of ANN model evaluated on validation. The correlation of the predicted change and the observed change is represented by the coefficient of correlation (r). The significant level of a validation is indicated by a p-value. A commonly accepted p value is 0.05. For validate data sets, the percentage error smaller than 10 percent, 27 percent and 55 percent is shown in Table V [7].

Sanjay dubey et.al [8], uses the MLP with software metrics for the prediction of software maintainability which is an imperative attribute of software quality. Maintenance effort was chosen as dependent variable and object-oriented metrics as the independent variables. Prediction of maintainability is performed by Multi Layer Perceptron neural network.

Thwin and Queh [9] present a neural network modelling technique along with regression analysis called GRNN to improve the quality of software products. In this paper, ward neural network and General regression neural network are used. First on predicting the number of defects in a class and the second on predicting the number of lines changed per class.

Kodhai.A and Kanmani.[11] present a novel code clone detection approach using textual analysis and software metrics. 12 software metrics at method level instead of 7 are used. It has also been implemented as a tool using Java. The tool efficiently and accurately detects type-1, type-2, type-3 and type-4 clones found in source codes at method level in JAVA open source code projects. The main limitation of this research is that it is language dependent and detect clone in JAVA open source project only.

Rubala et.al[12] did research of code clone detection in web based application. Web based applications used the commerce functionality in web sites. Scripting languages such as ASP, JSP, PHP etc are used in the development of web sites in which code duplication practice usually involved in making of several web pages. Hybrid approach (textual and metric based) is used. The proposed method is implemented as a tool in .NET. A set of 7 existing function level metrics are used for the detection of all types of clone functions in web application. The proposed tool gives its evaluated result in precision and recall parameter which then further compared with the other existing tool called eMetrics. The result of comparison showed that the value of precision and recall in term of percentage with the proposed tool using .NET gives higher value with accuracy than the eMetrics tool. The limitation of this research is problem with working on larger and even more complex system.

Dr. C.R.K Reddy et.al [13] also uses metrics and textual based technique to find the code clones in a software projects. They use a tool to implement the proposed work in JAVA. The technique easily deals with type-1 and type-2 clones.

Yogita Sharma et.al [23] present hybrid approach for detection of code clones. In this research, object oriented metrics and text based technique are used for the detection of exact clones. An automated tool for exact code clone detection was developed in VB.Net which calculates the metrics of the C/C++ projects and also performs the analysis of code clone detection that is which project function or the class had the code clone by using textual comparison. This approach has the limitation that it is only limited for C/C++ projects or software.

III. ADVANTAGES & DISADVANTAGES OF CODE CLONE DETECTION

A. *ADVANTAGES OF CODE CLONE DETECTION*

Code clone duplication has many advantages in the development of software project. Some of them are discussed as-

1) ***Detects library candidates:*** Code fragment proves its usability by coping and reusing multiple times in the system that can be incorporated in a library and announce its reuse potential officially [23].
2) ***Understanding program:*** It is possible to get an overall idea of other files containing similar copies of the fragment, if the functionality of a cloned fragment is understood. For example, when we have a piece of code managing memory we know that all files that contain a copy must implement a data structure with dynamically allocated space. [22][24].
3) ***Helps aspect mining research:*** Detecting code clone is also necessary in aspect mining to detect cross-cutting concerns. The code of cross-cutting concerns is typically duplicated over the entire application that could be identified with clone detection tools. [23][24].
4) ***Finds usage patterns:*** The functional usage patterns of the cloned fragment can be discovered if all the cloned fragments of the same source fragments are detected [23][24].
5) ***Detects malicious software:*** To detect malicious software clone detection techniques can play a vital role. By comparing one malicious software to another, it is possible to find the evidence where parts of the one software system match parts of another [23][24].
6) ***Helps Detecting plagiarism copyright content:*** Finding similar code may also useful in detecting plagiarism and copyright infringement [23][24].
7) ***Software evolution:*** Clone detection techniques are successfully used in software evolution analysis by looking at the dynamic nature of different clones in different versions of a system [23][24].
8) ***Code compacting:*** Clone detection techniques can be used for compact device by reducing the source code size [23][24]

B. *DRAWBACKS OF CODE CLONE*

Apart from benefits of code clones, it has severe impact on the quality, reusability and maintainability of a software system. The following are the list of some drawbacks of having cloned code in a system.

1) ***Increased probability of bug propagation:*** If a code segment contains a bug and that segment is reused by coping and pasting without or with minor adaptations, the bug of the original segment may remain in all the pasted segments in the system and therefore, the probability of bug propagation may increase significantly in the system [23][24].
2) ***Increased probability of introducing a new bug:*** In many cases, only the structure of the duplicated fragment is reused with the developer's responsibility of adapting the code to the current need. This process can be error prone and may introduce new bugs in the system [23][24].
3) ***Increased probability of bad design:*** Cloning may also introduce bad design, lack of good inheritance structure or abstraction. Consequently, it becomes difficult to reuse part of the implementation in future projects. It also badly impacts on the maintainability of the software [23][24].
4) ***Increased difficulty in system upgradation:*** Because of duplicated code in the system, one needs additional time and attention to understand the existing cloned implementation and concerns to be adapted, and therefore, it becomes difficult to add new functionalities in the system, or even to change existing ones [23][24].
5) ***Increased maintenance cost:*** If a cloned code segment is found to be contained a bug, all of its similar counterparts should be investigated for correcting the bug in question as there is no guarantee that this bug has been already eliminated from other similar parts at the time of reusing or during maintenance[23][24].
6) ***Increased resource requirements:*** Code duplication introduces higher growth rate of the system size. While system size may not be a big problem for some domains, others (e.g., telecommunication switch or compact devices) may require costly hardware upgrade with a software upgrade. Compilation times will increase if more code has to be translated which has a detrimental effect on the edit-compile-test cycle.

IV. CONCLUSION

All the advantages and disadvantages of various approaches discussed but it clearly shows that no one technique is able to find the clones correctly. All the approaches discussed above gives 75-85% accuracy in detection and analysis of code clones but no one approach is able to find all clones 100% accurate. So it is concluded that metric based technique using neural network give more accurate results as compared to other techniques. It is much faster and less complexity as compared to other techniques. It can be applied to any types of application software. In this survey paper we focused on code clone detection and analysis methods which help us for understanding code clones and the different techniques used. We conclude that the metric based clone detection approach using neural network is very effective approach as it discovered the clones and also helps in identifying the clones of each types. A neural network has the immense detection and prediction capability and it can be applied to any types of programming languages.

ACKNOWLEDGMENT

First and foremost, I would like to express my eternal gratitude to Almighty GOD, who has given me the power and ability to complete this work. I would like to express my heartfelt gratitude to my Guide Dr. Satwinder Singh for his valuable advice and healthy criticism throughout my research which helped me immensely to complete my work successfully. Finally, I am grateful to my parents and my wife, for their faith in me and allowing me to be as ambitious as I wanted. It was under their watchful eye that I gained so much drive and an ability to tackle challenges ahead on me.

REFERENCES

[1] Roy CK, Cordy JR, Koschke R. "Comparison and evaluation of clone detection techniques and tools: A qualitative approach. Science of Computer Programming "2009; 74:470–495.

[2] Roy CK, Cordy JR. A survey on software clone detection research. TR 2007-541, Queen's School of Computing, 2007; 115

[3] D.Gayathri Devi, Dr. M. Punithavalli "Comparison and Evaluation on Metrics based Approach for Detecting Code Clone" in IJCSE, ISSN: 0976-5166 Vol. 2 No. 5 Oct-Nov 2011 page no- 750.

[4] Jean Mayrand, Claude Leblanc, Ettore M. Merlo " Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics" ICMS 1996 1063-6773

[5] Pavitdeep Singh, Prof. Satwinder Singh, Prof. Jatinder Kaur "Tool for Generating Code Metrics for C# Source Code using Abstract Syntax Tree Technique" ACM Sigsoft, software engineering notes-vol-3 sep-2013 pages1-6

[6] Sandeep Sharawat : Software maintainability prediction using Neural Network, Vol. 2, Issue 2,Mar-Apr 2012, pp.750-755

[7] K.K Aggarwal, Yogesh Singh et.al : Application of artificial neural network for predicting maintainability using object oriented metrics, world academy of science, engineering and technology, 22, 2006.

[8] Yajnaseni Dash, Sanjay Kumar Dubey, Ajay Rana, "Maintainability Prediction of Object Oriented Software System by Using Artificial Neural Network Approach", in IJSCE ISSN: 2231-2307, Volume-2, Issue-2, May 2012.

[9] Mie Mie Thet Thwin, Tong-Seng Quah" Application of Neural Networks for Software quality Prediction Using Object-Oriented Metrics" in ICSM, ISSN: 1063-6773 Sep-2003.

[10] Filip Van Rysselberghe, Serge Demeyer. Evaluating Clone Detection Techniques. In Proceedings of the International Workshop on Evolution of Large Scale Industrial Applications (ELISA'03), 12pp., Amsterdam, The Netherlands, Sept 2003.

[11] Kodhai. E, Perumal. A, and Kanmani. S,"Clone Detection using Textual and Metric Analysis to figure out all Types of Clones", in IJCCIS, Vol2. No1. ISSN: 0976–1349 July-Dec 2010.

[12] Rubala Sivakumar, Kodhai. E,"Code Clones Detection in Websites using Hybrid Approach", in IJCA (0975 – 888) Volume 48–No.13, June 2012.

[13] Dr. C.R.K Reddy, Dr. A.goverdhan and G.Anil kumar, "An efficient method-level code clone detection scheme through textual analysis using metrics" IJCET, ISSN: 6375, Vol-3, Issue-1, Jan-June 2012, pp 273-288.

[14] R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design." IEEE Trans. Software Eng., vol. 20, no. 6, 1994, pp. 476–493.

[15] E. Burd, J. Bailey, Evaluating clone detection tools for use during preventative maintenance, in: Proceedings of the 2nd IEEE International Workshop on Source Code Analysis and Manipulation, SCAM 2002, 2002, pp. 36–43.

[16] R. Koschke, R. Falke, P. Frenzel, Clone detection using abstract syntax suffix trees, in: Proceedings of the 13th Working conference on Reverse Engineering, WCRE 2006, 2006, pp. 253–262.

[17] B. Baker, A program for identifying duplicated code, in: Proceedings of Computing Science and Statistics: 24th Symposium on the Interface, vol. 24, 1992, pp. 49–57.

[18] B. Baker, On finding duplication and near-duplication in large software systems, in: Proceedings of the 2nd Working Conference on Reverse Engineering, WCRE 1995, 1995, pp. 86–95.

[19] J. Krinke, Identifying similar code with program dependence graphs, in: Proceedings of the 8th Working Conference on Reverse Engineering, WCRE 2001, 2001, pp. 301–309.

[20] I. Baxter, A. Yahin, L. Moura, M. Anna, Clone detection using abstract syntax trees, in: Proceedings of the 14th International Conference on Software Maintenance, ICSM 1998, 1998, pp. 368–377.

[21] Sugandha Saha, "Comparison of performance analysis using different neural network and fuzzy logic model for prediction of stock price", M.E Thesis submitted at NIT Rourkela Odisha , 2013.

[22] Mrs.Prajila Prem, "A Review on Code Clone Analysis and Code Clone Detection", IJEIT, Issue-12, vol-2, 2013.

[23] Yogita Sharma, Rajesh Bhatia, "Hybrid technique for object oriented software clone detection'- a thesis submitted in june-2011, Thapar University, Patiala.

[24] Md. Arifur Rahman and Salah Uddin , "A Literature Review of Code Clone Analysis to Improve Software Maintenance Process", Published in The Computing Research Repository (CoRR) of arXiv.org arXiv 2012.