Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 12, December 2014, pg.289 – 294

RESEARCH ARTICLE

A Weighted Layered Approach for Code Clone Detection

Himanshu[1], Dr. Sushil Garg[2]

[1]Student, Dept. of Computer Science Engineering, RIMT-IET, Mandi Gobindgarh

[2]Professor, Dept. of Computer Science Engineering, RIMT-IET, Mandi Gobindgarh

Abstract

Code coning is considered as the challenge to the software quality. To develop a reliable and authenticated software system, it is required to remove the clone based bugs over the software system. There are number of associated approaches but no one is able to provide the reliable solution. In this work, a combined mathematical model is presented to overcome the limitations of existing approaches. The presented model is divided in four main layers. Each layer is represented by a separate mathematical approach. In the later stage, the weighted phenomenon is used to combine all these approaches to obtain the overall code similarity. The work is presented in the form of an automated tool. The obtained results shows the robustness and accurate detection of code cloning for software system.

Keywords : Code Cloning, Software Quality, Layered Model

I. INTRODUCTION

The software reuse is one of the common activity applied during the development phase. This activity is about to use the existing code and fragments to develop a new software system. But if these existing code segments are not authorized to recuse, it is considered as the theft. The existing code can be taken from some web source or the effort of some other programmer. This kind of code reuse is required to identify to save the code level plagiarism. This kind of identification is considered as code clone detection. It is observed that more than 50% code cloning is done intensionly by using differnt techniques. These kind of approaches are not beneficial while testing or maintaining the software system. To develop a reliable and significant software system, it is required to maintain the code quality analysis. Code clone identification is one of the major approach that comes under code quality analysis. There are number of aspects of code quality analysis shown in figure 1.

The foremost aspect of code quality analysis is to use the authenticated code. The code that is available for private use is considered as the authenticated code. The code extracted from the web from various sources comes under plagriarised code. It is required to identify such kind of code over the web to maintain the code quality.

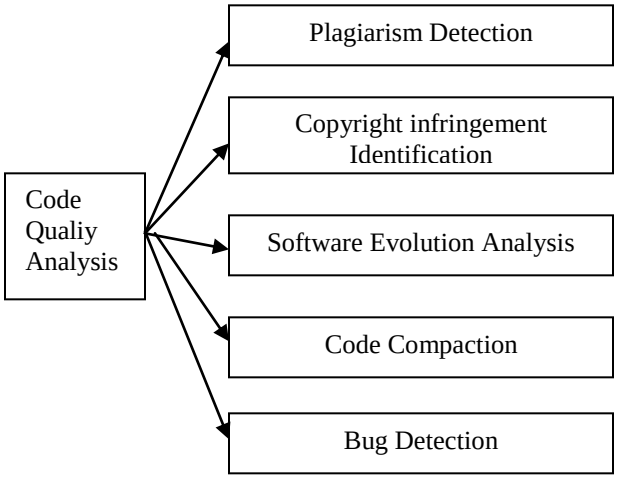

Figure 1 : Aspects of Code Quality Analysis

Another aspect of code quality is not to violate the copyright rules. It means only the authorized code must be used for software development. If the code is outsourced or taken from some external bodies, it is required to take the copyrights from that body. This kind of prevention improves the code quality. Another quality analysis vector is the evolution analysis of software system. This kind of analysis is performed respective to the code segment reusability. It is defined to identify the code fragment under various reusability parameters and identify whether the code is effective to reuse or not. A code having the better evolution vector is considered as the effective and reliable codes. Another issue that codes under code quality analysis is code compaction. Code compaction is actually to remove the duplicate code over the software system. This reduces the code size. Smaller the software code, more reliable the software is considered. The last vector that describes the code quality vector is bug identification. A code having the lesser bugs or errors is considered as the reliable code. This kind of software system have the lesser chances of software fault and failures.

One of the effective approach to overcome some of these problems is code clone detection. There are a number of automated as well as semi automated approaches to identify the code clone. These approaches include the mathematical as well as evolutionary approaches. Different researchers provided different quantitative evaluation approaches based on different language system.

The presnted work has combined some of such mathematical approaches to identify the code cloning in C programs. The work is defined as the weighted composite model that is able to identify the code cloning over the software system. In this section, a brief introduction is given to identify the requirement of quality software as well as different aspects associated with it. One of such aspect is code clone detection. In section II, the paper has defined some of the existing code clone detection approaches defined by different researchers. In section III, the proposed model is shown alogn with relative algorithm. In section IV, the results obtained from the work are presented and discussed. In section V, the conclusion obtained from the work is presented.

II. EXISTING WORK

Different researchers has provided lot of work to identify the code clone for different language systems. In this section, some of the work defined by earlier researchers is shown. Girija Gupta[1] has designed a tool to detect the code cloning for java programs. Author defined a metrics based analysis approach to quantize the cloning ratio over the system by refactoring the code. Author has defined a layered model by adjusting the code in required format. Once the code is organized, later on the clone identification is performed. Author used some class metrics as well as function level metrics for the analysis. Obtained results shows the code effective and efficient mechnism is obtained from the work. Paul Baker[2] has defined an approach to identify the cloning the software structure analysis. This analysis includes the detection of pathologies under differnt vectors. Author presnted an improvement over existing UML system to identify the cloning at design phase. Author presented a theoritical framework to analyze the communication between differnt software components and identify the cloning ratio under differnt vectors. The work is able to resolve the semantic error

over the software system. Huiqing Li [3] has defined a work on code similarity identification under unification approach. This approach is based on the refactoring vectors. Author has designed a semi automated tool that required the user support to identify and remove the duplicate code in software system. Author also presnted the case study to show the effectiveness of work under differnt platforms and environments. Bakr Al-Batran[4] has defined a model based system to detect the code cloning automatically over the system so that the maintainability of the system will be improved. Author defined a syntatic tool to analyze the identical structure of software code. Author analyzed this model on simulink models under pattern based consideration so that the robust and effective clone detection will be done.

Yang Yuan[5] has defined a token based approach for code clone detection. In this approach, the token count mechnism is implemented under different metrics to analyze the distictiveness in software modules. Author has defined a structured model in the form of a syntactic tool. This tool had divided the system in the form of code segments and later on the component extraction is performed. Once the component extraction and categorization is done, the metrics are applied over it to perform the similarity check. Radhika[6]has defined a priortization effective approach for clone detection over the code. Author has defined a criteria specific priortization approach to analyze the code under industrial requirements and later on the quality standard based analysis is performed over the code. Author has identify the level of fixing over the code to improve the quality standard. Author also presented the technology level analysis based on the project model so that the cloning over the software system will be done. Hiroaki Murakami[7] has defined an improved token based approach for code clone analysis. Author has applied multi factor analysis to identify the duplicacy on two two statements. The redundancy or overlapping is identified so that the repeated code will be removed and code compaction based quality improvement will be done. Author presented the mechanism in the form of a automated tool and obtained the effective results from the system.

Miryung Kim[8] has defined a study to explroe the assocaited terms with code cloning and relative approach. Author analyzed various tools and improved the assocaited under differnt case studies. Author explored the challenges assocaited with work and various tools so that the exploration to the limitations will be done. D. Gayathri Devi[9] also presnted a comparative as well as analytical study on differnt code clone detection approaches. Author analyzed the similar patterns under design and code level to formulate the problem and to identify the associated issues with the software system. Author defined the work as the approaches to identify the semantic and syntactic cloning over the code. Author has applied token based approaches to summaries the study and explore the limitations. Chanchal K. Roy[10] also presnted a work to explroe the code cloning by presneting the analytical study over them. Author analyzed various aspects and dimension with these software system so that the conceptual framework will be explored.

III. RESEARCH METHODOLOGY

In this present work, a mathematical layered framework is presented to identify the code cloning over the software modules or the programs. The work is here presented as an automated tool to identify the similar code over the code segments. The identification of code cloning is here under under different methods for C code segments. This kind of duplicate code identification is done in the form of a four layered model. Where each layer has represented a separate method and each method worked individually to analyze the code similarity. At the end these all methods are collected under the weighted approach so that the detection of code cloning

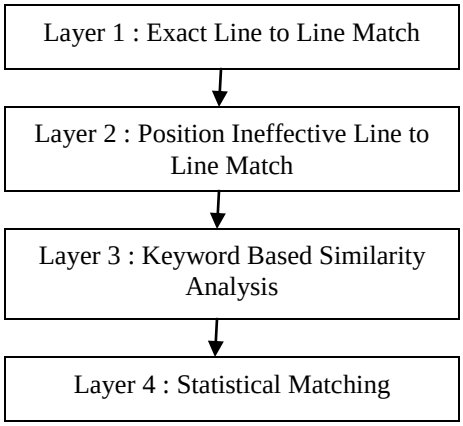

Figure 2 : Proposed Framework

over the system will be done more accurately and reliability. The use of multiple approaches in the system has improved the accuracy and robustness of work. If one approach is not able to identify the code cloning, other approach can overcome its limitation. It means, the presented method is having the maximum chances of detection of cloning over the clode. To basic framework of work is shown in figure 2. The figure has explored all the four approaches associated with this work.

A) Layer 1 : Exact Line By Line Match

This layer is able to identify the cloning in the simplest case when some lines of code are altered by the user. In this such case, both the codes will be analyzed respective to the line by line code segments and overall clone detection and identification will be performed. Let we have two code segments of line N and M. Then the similarity analysis will be performed by

$$\text{SimAnalysis} = \frac{\text{Code1}(1..N) \cap \text{Code2}(1..M)}{\text{Max}(N,M)}$$

The complexity of this method is linear so that the method is considered as the effective approach.

B) Layer 2 : Position Ineffective Approach

This approach is able to identify the code cloning if the user has shifted the codes over the code. It means the code statements are shuffled. It may be possible some lines are altered or removed from the code. In such case, to identify the cloning, it is required to compare each line of orignional code over the copied code. This approach will perform the same kind of check to detect the code. The algorithmic approach used in this stage is shown in table 1.

Table 1 : Algorithm

```
Algorithm (Code1, Code2)
/*Code1 is the Origional Code and Code 2 is the copied
code segment*/
{
    1.  For i=1 to Length (Code1)
        {
    2.  For j=1 to Length(Code2)
        {
    3.  Line1=Code1(i)
    4.  Line2=Code2(j)
    5.  If (Line1=Line2)
        {
    6.  SimCount=SimCount+1
        }
        }
        }
    7.  Sim=SimCount/(Length(Code1)+Length(Code2)
    8.  Return Sim
}
```

As shown, in the table, the complexity of this approach is higher i.e. O(NxM).

C) Keyword Based Matching

In this approach, instead of performing the line by line match, complete code segments are divided in the form of smaller tokens. These tokens are identified using the separators. These separators includes the punchuaction marks, spaces, line break etc. An effective approach is defined to identify the tokens from the code. Once the tokens are extracted, the next work is to arrange the tokens in specific order. After this arrangement, the token wise match is performed on input codes. Maximum the number of token matched between code, more the similarity ratio is obtained.

D) Statistical Matching

This is considered as the metrics based approach. In this approach, the size materics are obtained for the code segments to perform the statistical matching. The size metrics is here performed based on the different kind of components used in the software system. These components includes the data types, keywords, branch statements etc. Once these all statistics are obtained for software system, the similarity analysis is performed based on different component types. Later on aggregative decision is taken for the similarity match.

In this work, four different approaches are combined to perform the similarity match between the codes. Each method will identify the similarity ratio independently. Once the individual ratio is obtained, the weighted approach is applied to obtain the aggregative similarity ratio for the system.

SimRatio = Layer1 * W1 + Layer2 * W2 + Layer3 *W3 + Layer4 * W4

The manual decision is required about the weightage assignment. According to the code complexities the weightage can be assigned to the analysis system under the given constraint

$$W1 + W2 + W3 + W4 = 1$$

It means the aggregative weightage assigned to the system can be maximum 1.

IV. RESULTS

The presented work is implemented as an automated tool in .net environment. The tool is defined as the graphical interface to accept the C code as input. The work has provided the layer based implementation of work. The results obtained from the system for different code segments is given here under in table 2. In this work, equal weightage is assigned to all the parameters

Table 2 : Analysis Results

File	Layer1	Layer2	Layer3	Layer4	Agg Result
a.c	0.73	0.46	0.47	0.38	0.51
b.c	0.29	0.24	0.26	0.78	0.3925
c.c	0.67	0.36	0.38	0.69	0.525
d.c	0.49	0.31	0.49	0.76	0.5125
e.c	0.55	0.19	0.69	0.46	0.4725
f.c	0.69	0.59	0.48	0.1	0.465

The aggregative results obtained formt he work work is shown in figure 3.

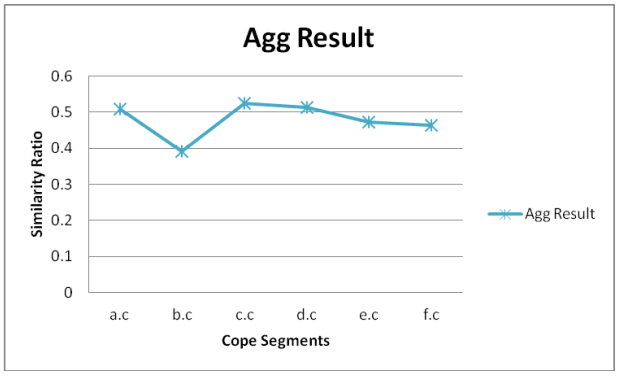

Figure 3 : Analysis Results

Here figure 3 is showing the analytical results obtained from the system.

V. CONCLUSION

Code cloning is having the complex architecture so that it becomes one of the major challenge for code quality maintenance. In this work, a layered weighted model is presented to perform the code quality analysis under code clone detection. The work is here presented as the automated mathematical tool. The obtained results from shows the effective detection of code cloning ove the C code.

REFERENCES

[1] Girija Gupta, Indu Singh, "A Novel Approach Towards Code Clone Detection and Redesigning", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 3, Issue 9, pp 331-339, 2013

[2] Paul Baker, Paul Bristow, "Detecting and Resolving Semantic Pathologies in UML Sequence Diagrams", ESEC-FSE ACM, pp 50-59, 2005

[3] Huiqing Li and Simon Thompson, "Similar Code Detection and Elimination for Erlang Programs", Springer, pp 104-118, 2010

[4] Bakr Al-Batran1, Bernhard Schätz1, and Benjamin Hummel, "Semantic Clone Detection for Model-Based Development of Embedded Systems", Springer, pp 258-272, 2011

[5] Yang Yuan and Yao Guo, "Boreas: An Accurate and Scalable Token-Based Approach to Code Clone Detection", ASE ACM, pp 286-289, 2012

[6] Radhika D. Venkatasubramanyam, Shrinath Gupta, Himanshu Kumar Singh, "Prioritizing Code Clone Detection Results for Clone Management", IWSC IEEE, pp 30-36, 2013

[7] Hiroaki Murakami, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki and Shinji Kusumoto, "Folding Repeated Instructions for Improving Token-based Code Clone Detection", International Working Conference on Source Code Analysis and Manipulation, pp 64-73, 2012

[8] Miryung Kim, Vibha Sazawal, "An Empirical Study of Code Clone Genealogies", ESEC-FSE ACM, pp 187-196, 2005

[9] D. Gayathri Devi, "COMPARISON AND EVALUATION ON METRICS BASED APPROACH FOR DETECTING CODE CLONE", Indian Journal of Computer Science and Engineering (IJCSE), pp 747-751, 2011

[10] Chanchal K. Roy,, James R. Cordya, Rainer Koschke, "Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach", Science of Computer Programming, pp 1-43, 2009

