Cross Language Higher Level Clone Detection- Between Two Different Object Oriented Programming Language Source Codes

[1]K. Vidhya, [2]N. Sumathi, [3]D. Ramya,

[1, 2] Assistant Professor [3] PG Student,
Dept. of C.S.E, Dr. MCET, Pollachi, Tamil Nadu, India

Abstract— Similar type of source codes or repetition of source codes in the software is known as code clones. Clone detection technique is capable of identifying the similar type of source codes present in software applications. These code clones increases the fault and maintenance cost. New source codes obtained from another source code without any proper changes lead to error. Detection of code clones help in reducing the software maintenance and plagiarism detection. Some of the clone detection techniques are token based approach, tree based method, etc. But, the process involved in them are very multifaceted and also identifies only lower level clones of same programming language source code. The system mainly focuses on higher level (Method, File, and Directory) clone detection in different programming language source code called cross language higher level clone detection. The metric based approach helps to reduce the computational cost in terms of resources and time with high precision. This cross language clone detection increases the interest in multiple languages. Clone detection results can be measured using precision and recall.

Keywords— Higher level code clone; metric values; high precision

I. Introduction

Clone detection is a major phenomenon used in the field of software engineering to find clones. Clone detection approach helps to find the duplication of programming language source codes that are widely used. The source code that occurs more than once across different programs of the same entity are called code clones [1]. Code clones in the software system may introduce bugs resulting in the decrease of understandability in code snippets [2]. Error containing source codes or inconsistent source codes when extended will increase the effort of modification. There are lots of reasons for the occurrence of clones. Copy paste method is the way of copying source code from the existing methods and reusing it with some modifications. This method is the major reason for producing clones in software. This method increases the maintenance cost due to inconsistent changes in various copies of the source code [3]. Plagiarism is another reason for clones. As the requirement is growing day by day coding is becoming larger and complex. Extensive software systems are pricey to build and, are even more costly to maintain. Sometimes, developers take uncomplicated way of implementation by copying some fragments of the existing programs and use that code in their work. This type of work is called code cloning [20][19].

Simple clones are identical programming language source codes with minor modifications like variable renaming and variation in literals. Continuous occurrence of simple clones with minor modifications between two or more source codes may lead to method level or file level called higher level clones [2]. The methods are extracted from a file and metric values are calculated and compared for different source codes and which are similar termed as method level clones. File level clones are identical source codes present in two different files. The method level and file level clone detection is mainly for reducing the deviation of the source code and also for improving the source code quality. These clone detection techniques are used in various fields such as software evolution analysis, detecting bugs and copyright infringement investigation [17]. In general, there are two ways of identifying the similarity between source

code segments: textual and functional similarity [10]. Textual similarity finds the code fragments that are exactly the same based on the content (text) with minimum modifications. Copy paste method comes under textual similarity. Two source code fragments identical based on the similar pre and post conditions are referred as functional similarity [20]. There are four types of code clones in software. They are type-1, type- 2, type-3, and type-4. They are defined as follows:

Type-1: Two code fragments identical but with some modification like comments, whitespace.
Type-2: Two code fragments similar based on syntax with little difference in identifiers, literals and layouts.
Type-3: Identical code segments with few variations in types, comments, adding or removing statements in the source code.
Type-4: Similar type of code fragments, with different implementation procedures followed [20].

In the above types of clones defined, the first three types come under the category of textual similarity and type-4 clone belong to functional similarity. The metric based approach is used to detect higher level clones present in cross language platform such as Java and C++. This approach indirectly identifies the similar type of programming language source codes. The source code similarity can be identified for detecting clones using metrics. Metric based technique is used for finding clones much easier and also this technique yields high precision and recall. Metrics are individually calculated for both methods and files. This technique is calculating metric values for identifying similar type of source codes across different files. These metric values are calculated using computed metrics.

These metric values are compared to find clones, instead of comparing source code directly. In case of direct comparison (Line by line comparison) of source codes, it is difficult and also takes long time to find clones in source code. This approach can support different type of programming language source code clone detection. The overall system architecture of higher level clone detection is shown in Fig. 1.

In metric based technique, compares two different source code metric values such as method level metric values for method level clone detection. Matches between different source code metric values to be satisfied with the threshold value considered as method level clones. Likewise file level clone detection can be done using file level metrics.

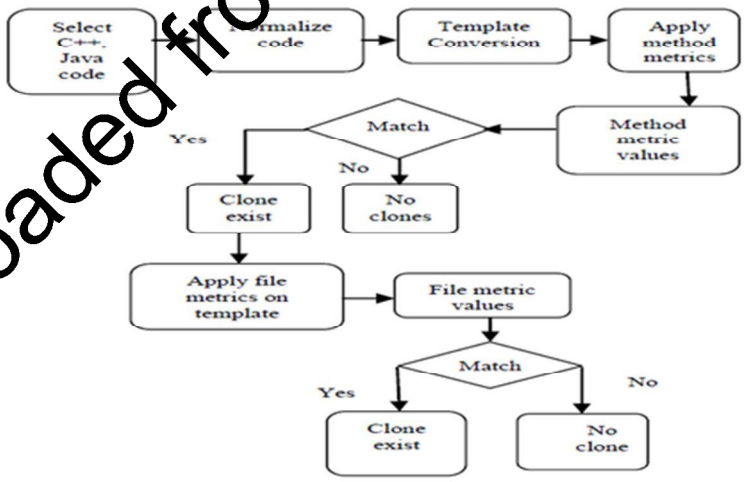

Fig. 1. System architecture

The clone relations are referred using the terms such as clone pair and clone cluster. Clone pairs indicate that two different code fragments or segments are identical to each other. A large number of code segments forming a pair of identical code fragments are referred as clone cluster [2]. All clones grouped under the same domain are called clone class family, otherwise it is known called super clone [17].

Generally clones are broadly classified into two categories: namely exact clones and near-miss clones [2]. Exact clone or type-1 clone is defined as two or more code fragments that are identical in nature with a small variation in blank space, new line and adding or removing tabs [17]. Type-2 and type-3 clones may be termed as near miss clones. Earlier papers dealt with the clone detection in single programming language source code. The widely increased usage of the source codes need a cross language clone detector which is proposed in this paper.

II. Process of Clone Detection

The clone detection process helps to detect higher level clones. Comparison of two or more code fragments takes place for finding similarity between the source codes. The identical code fragments and identical files are concluded as clones [7, 2, 10].

Metrics are classified as method level and file level metrics. The following method level metrics computed for source codes are:

1. Number of lines of code
2. Number of arguments passed
3. Number of function calls
4. Number of local variables
5. Number of conditional statements
6. Number of looping statements
7. Number of return statements
8. Number of assignment statements

The following shows of metrics that are computed for source codes are[19]:

1. Number of lines of code
2. Number of variables declared
3. Number of methods defined
4. Number of function calls
5. Sequence of function calls

A. Input selection

Fig. 2 shows the input source code that is taken for clone detection. Select two different object oriented programming language source code such as Java and C++ for preprocessing. The input source code can be extracted from software component finder and source code repository. Same source code should be selected for both Java and C++ programming language.

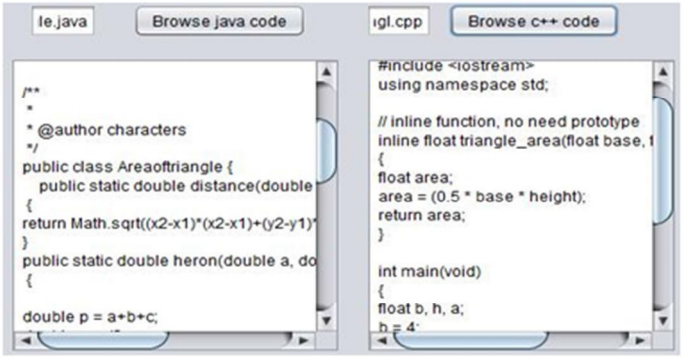

Fig. 2. Input file

B. Pre-processing the Source Code

The screen shot of source code preprocessing is shown in Fig. 3. Selected source code should be normalized. Normalization is the removal of header files, removal of blank lines, removal of whitespaces and removal of single and multiple comment lines. This pre-processing step is most important for every source code helps to remove unnecessary lines [10].This step reduces the number of lines of source code and make the source to standardized format.

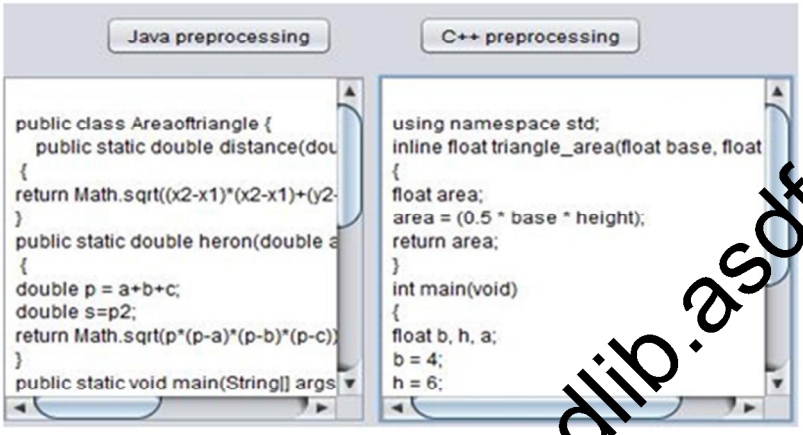

Fig. 3. Source code preprocessing

C. Intermediate Form (IR) – Template conversion

Fig. 4. shows the template conversion of different programming language source code. After preprocessing step, convert the source code into intermediate form. Each line of source code can be converted into tokens to make template conversion easier.

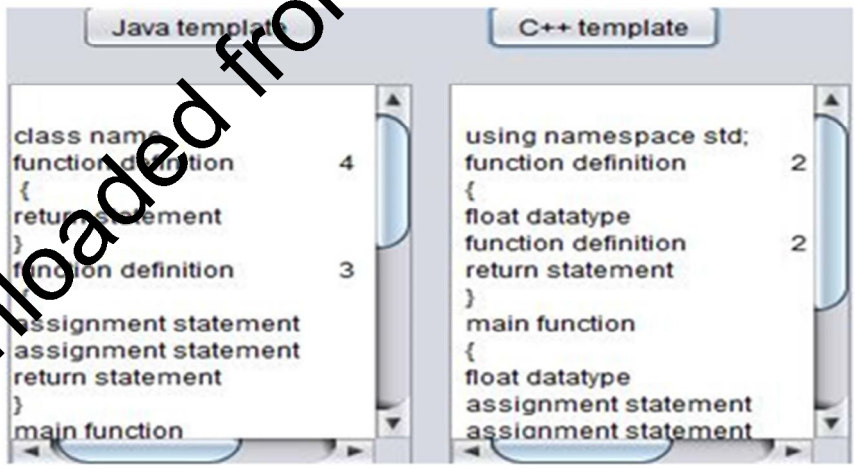

Fig. 4. Template Conversion

D. Method Extraction

The methods are extracted after template conversion. Each and every method is individually extracted from both Java and C++ files. This step is mainly focus on methods alone and avoids remaining lines of source code. The methods are only used to calculate method level metric values

E. Metric value Calculation

Fig. 5. and Fig. 6. listed the calculated metrics values for both Java and C++ source code. Source code metrics are computed for calculating metric values.

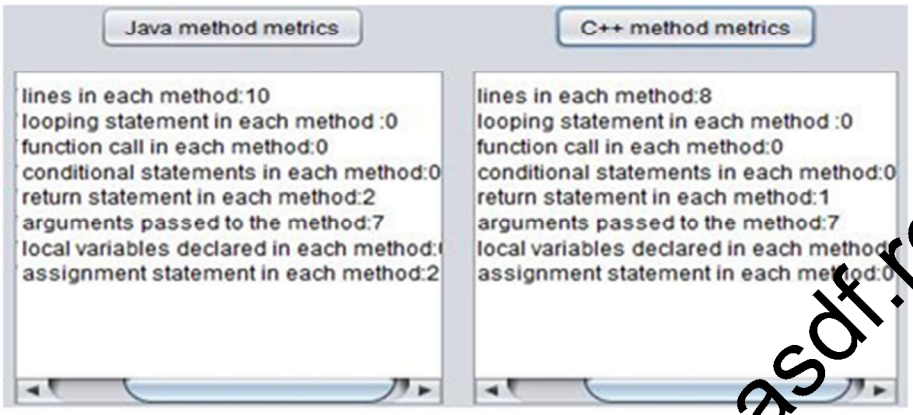

Fig. 5. Method level metrics

File level and method level metrics are listed in section1.The metric values are calculated for different methods and files. The metric values are generated after the template conversion. This approach helps to detect higher level clones easier.

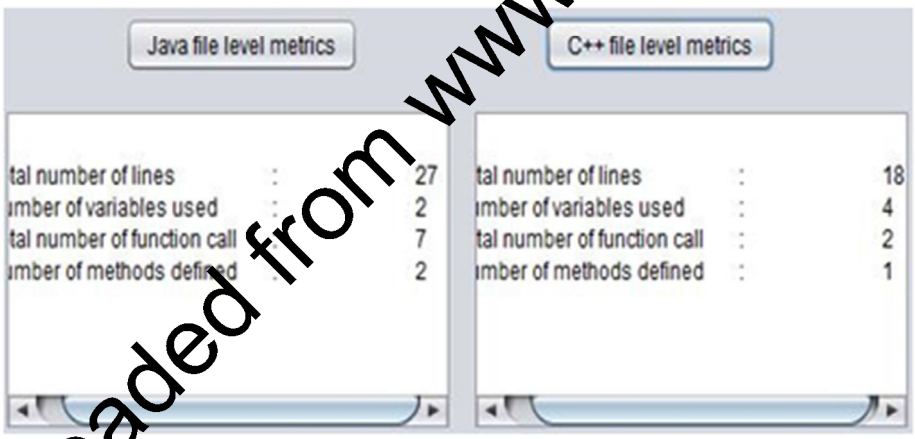

Fig. 6. File level metrics

Few examples for template conversion is renaming variable, renaming of data type etc. This step is carried out for each and every line of source code. This Java and C++ template mainly helps for comparison of two different programming language source code easier [20] [8].

F. Identify Matches

The percentage of method level and file level clone detection to be displayed in the Fig. 7. The metric values are compared for finding similarity between the source code. Java source code metric values are compared with C++ metric values for both method level and file level clone detection [18]. Finally aggregate the matches (identical code fragment) metric values between two different programming language source codes. The aggregation of similar metric values is done based on the threshold value defined by the user and the clone is detected.

III. Result

The metric values of Java and C++ codes are calculated using the above proposed technique. All the metric values are obtained. The difference between each metric values of both Java and C++ are obtained for each metrics listed in the Section I. The average of the metrics is calculated. A threshold is set and the presence of exact and near miss clone is detected accordingly.

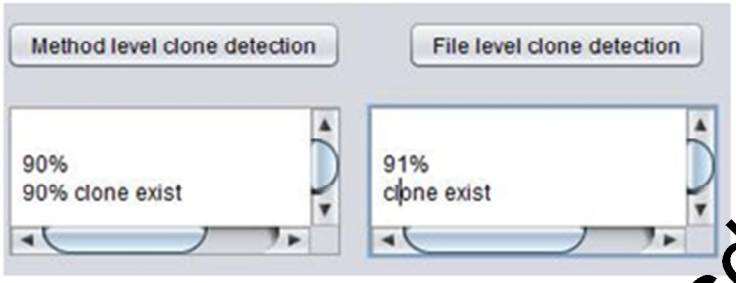

Fig. 7. Clone detection

The clones are classified based on the threshold value as shown in the Table I

Table I: Clones classified based on threshold value

Average (%)	Existence of Clones
>=90	Exact Clone
>=70	Near miss Clone
<=69	No Clone

IV. Conclusion

This metric based technique helps to identify higher level clones in source codes. Additionally in the latter stage the textual comparison of the converted template of the source code is also employed. This type of comparison increases the performance of clone detection measure such as high precision and recall. This clone detection approach also reduces the computational overhead. The file level clone between two different language source codes detected with a higher accuracy.

The proposed cross language clone detection approach reduces the time complexity compared to the direct Comparison approach and also increases the importance in multiple programming languages. This work may be extended by developing a generic tool which accepts source codes of any language for clone detection and also can go for next higher level detection.

References

1. Balint, Mihai, Tudor Girba and Radu Marinescu,"How developers copy". Program Comprehension, ICPC 14th IEEE International Conference, pp.56-58, 2006
2. Basit, Hamid Abdul and Stan Jarzabek, "A Data Mining Approach for Detecting Higher-level Clones in Software." , Software Engineering IEEE Transactions, Vol.35.4, pp. 497-514, 2009
3. Devi, D. Gayathri, and M. Punithavalli, "Detecting Software clones using Association rule mining"., International Journal of Advanced Technology & Engineering Research (IJATER) Volume 3, Jan. 2013
4. Gayathri Devi G , Dr. M. Punithavalli " Comparison and Evaluation On Metrics Based Approach For Detecting Code Clones" Indian Journal of Computer Science and Engineering (IJCSE) Vol. 2 No. 5 Oct-Nov 2011

5. Gehan M. K. Selim ,King Chun Foo, Ying Zou "Enhancing Source-Based Clone Detection Using Intermediate Representation" 17th Working Conference on Reverse Engineering, 2010
6. Hoan Anh Nguyen, Tung Thanh Nguyen, Nam H. Pham, Jafar Al-Kofahi, and Tien N. Nguyen "Clone Management for Evolving Software" IEEE Transactions On Software Engineering, Vol. 38, No. 5, September/October 2012
7. Geiger, Reto, et al, "Relation of code clones and change couplings (2006).", Fundamental Approaches to Software Engineering. Springer Berlin Heidelberg, pp.411-425, 2006
8. Kanika Raheja and Rajkumar Tekchandani,"An Emerging Approach towards Code Clone Detection: Metric Based Approach on Byte Code", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, pp.881-888.
9. Kaur, prabhjot, harpreet kaur and rupinder kaur, "Comparison of clone detection tools : CONQAT and solid SDD", International journal 2.5, 2012
10. Kodhai, Perumal, and Kanmani, "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, pp. 99-103, 2010
11. Kodhai .E, Kanmani .S, Kamatchi .A, Radhika .R, Detection of Type-1 and Type-2 Clone Using Textual Analysis and Metrics in ITC, 2010
12. Krinke ,J., "Identifying Similar Code with Program Dependence Graphs," in Proceedings of the 8th Working Conference of Reverse Engineering, pp.301- 309, Stuttgart, Germany, October 2001.
13. Lanubile .F, Mallardo. T., "Finding Function Clones in Web Applications" Proceedings of the Seventh European Conference On Software Maintenance And Reengineering (CSMR'03), 2003.
14. Mayrand, J., C. Leblanc and E. Merlo, "Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics," in Proceedings of the 12th International Conference on Software Maintenance (ICSM'96), pp. 244–253, Monterey, CA, USA, November 1996
15. Merlo, E., "Detection of Plagiarism in University Projects Using Metrics based Spectral Similarity," in the Dagstuhl Seminar: Duplication, Redundancy, and Similarity in Software, 2007.
16. Nicholas A. Kraft, Brandon W. Bonds, and Randy K. Smith, "Cross language clone detection", SEKE, pp. 54-59, 2008
17. Roy, chanchal Kumar, and james R. Cordy., "A Survey on software clone detection research", Technical report 541,Queen's university at Kingston 2007
18. Rysselberghe, Filip Van, and Serge Demeyer "Evaluating Clone Detection Techniques from a Refactoring Perspective." 19th IEEE international conference on Automated software engineering, IEEE Computer Society, pp. 336-339, 2004.
19. Vidhya .K, Thirukumar .K, "Identifying Functional Clones Between Java Directories Using Metric Based System" , International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, pp. 1255-1261, 2013
20. Vidhya .K and Thirukumar .K, "Detecting Functional Similarity between Java Files using metrics" IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol 3, No.4, pp. 290-296 , 2013