Volume 2, Issue 8, August 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Developing a Novel and Effective Clone Detection Using Data Mining Technique

D. Gayathri Devi
Department of Computer Science, Karpagam University, Coimbatore, Tamilnadu

Dr.M.Punithavalli
Director and Head, Sri Ramakrishna Engineering College,Coimbatore, Tamilnadu

***Abstract** − A code clone is a code portion that is identical or similar to another in a source files. These code clones are called simple clones. And higher level similarities are called structural clones. Since several techniques have been proposed to detect this simple clone, still they occur in large software systems. This new algorithm detects the code clone for control structures such as for, while and do statements. First we extract the structural clone. Secondly we split the code clone from the original source code called source units and assign index values for each statement. Source units are then encoded in tuple form as one indicating the statements of source units and the other represents the index occurrence. Finally we calculate total number of occurrence for each index value.*

***Keyswords**− Code Clone, Cluster, Detection, Frequent, Itemset, Index.*

I. INTRODUCTION

Code Clones are similar program structures recurring in variant forms in software. Code clone are known as cloning. In some cases we would not consider the whole code fragment as a duplicate, but only those lines that get matches. We consider both the fragments to have a similar structure or a part of the code fragment is common. Code clone can be defined as identical code that represents duplication in multiple locations among a collection of source files. Though cloning is an active area of research, still there are code fragments of duplicate code that affects quality and maintenance.

The existence of code clone becomes part of the software development process in which the clones are in need to be maintained. Code clone causes additional maintenance effort. As changes to one segment of code may need to propagate to several other codes, which lead to increasing maintenance cost. Maintenance effort can be increased when bugs have to be fixed multiple times, and these changes could be prone to errors [10], [11].

To detect code clone we apply the techniques of data mining concept where the frequent itemset mining technique is used. In frequent item mining, the items in transaction are considered unique, where as in detecting code clone multiple instances are considered.

II. LITERATURE REVIEW

A. *Occurrences of code clone*

Code clone is a widespread technique, although it is known to be a bad practice. But still the code clone exists? Why?

Code clone generally happens by the programmer who does not know to use the concepts of polymorphism in an optimal way. For instance, we may have code clone in a class hierarchy, without knowing that the super class has some similar that has already implemented. Cloning also occurs when they are quickly needed to implement some functionality that is already implemented. Although code clone is a bad practice, still it occurs in existing software development [17], [18].

B. *Impact of code clone*

Code clone have a severe impact on the quality, reusability and maintainability of a software system.

When a programmer copies an error code, the copy also includes the error in the code. This process can be error prone and introduce new bugs in software. Since code clone indicates bad design, it lack in good inheritance and abstraction. This makes very difficult to reuse the part of code implementation [19], [20].

III. CODE CLONE

A. Definition

A code clone, in general, means a code fragment that has identical or similar code fragment in source code. However, there is no generic definition for code clone. Several methods of code clone detection have been proposed, and each of them has its own definition about code clone. Still code cloning is considered as a serious problem in industrial software [12], [13], [14], [26], [27], [28], [29].

The literature on the topic has been described many situations that can lead to the duplication of code within software system and are considered to be spiteful cloning. Developers may duplicate code because the short term cost of forming the proper abstraction may prevail over the cost of duplicate code [15], [16]. Developers may also duplicate code when they fully understand the problem, or the solution, but they are aware of the code that can provide some functionality [25].

B. Code clone types

Type I: Identical code fragments except for variations in whitespace and comments called Exact clone.
Type II: Structurally or syntactically identical fragments except for variations in identifiers, literals, types, layouts and comments called Renamed clones.
Type III: Copied fragments with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layouts and comments called Gapped clones.
Type IV: Code fragments that perform similar functionality, but are implemented by different syntactic variants called Semantic clones.

C. Association Rules

Mining association rules is particularly useful for discovering relationships among items from large database. A standard association rule is a rule of the form x →y which says that if x is true of an instance in a database, and y is true of the same instance, with a certain level of significance as measured by two indicators, support and confidence [21] , [22], [30], [31].

The goal of standard association rule mining is to generate all rules whose support and confidence are respectively above some given support and coverage threshold. Association rules encapsulate the relational associations between selected attributes in the database and signify a positive relationship between x and y [23], [24], [32], [33].

The mining process of association rules are divided into two:

1. Frequent Item set Generation: Generate all sets of items that have support greater than a certain threshold called minsupport.
2. Association Rule Generation: From the frequent item sets, generate all association rules that have confidence greater than a certain threshold called minconfidence. Apriori is a renowned algorithm for association rule mining primarily because of its effectiveness in knowledge discovery. However, there are two bottlenecks in the Apriori algorithm. One is the complex frequent itemset generation process that uses most of the time, space and memory and another is the multiple scan of the database.

IV. DISCUSSION ON CODE CLONE USING CLUSTERING

A. Fuzzy C Means algorithm

The Fuzzy C Means algorithm has the ability to assign one data point to more than one cluster. The main purpose of this algorithm is trying to minimize the objective function. The FCM calculates membership value to all clusters for each data point, it usually generates a more consistent and trust-worthy results [3] [4].

In cluster-based methods, individual image pixels are considered as general data samples and assumed correspondence between homogeneous image region and cluster in the spectral domain [7] [9].

B. Hierarchical Clustering Algorithm for Crosscutting concerns Identification

HACO is based on the idea of hierarchical agglomerative clustering, and uses a heuristic for determining the number of clusters. Each method from the software system is put in its own cluster. HACO algorithm provides a partition of a software system S that ideally contains separate clusters of a set of classes where each clusters containing one or more methods [1] [2].

A clustering approach for identifying crosscutting concerns is proposed and a partitional clustering algorithm named kAM is introduced. It is based on K-means clustering algorithm and uses a heuristic method for choosing the initial centroids and the initial number of clusters. The similarity between two methods is computed using a vector space model based approach [8].

C. Semantic Clustering Algorithm

It is a novel technique to analyze the semantics of a software system. It offers a highlevel view on the domain concepts of a system, abstracting concepts from software aircrafts [5]. Semantic clustering is a non-interactive and unsupervised technique. Latent semantic indexing is used to exploit linguistic information from the source code contained in the names of identifiers and the content of comments [6].

D. Partitional Clustering Algorithm

A well known class of clustering methods is the one of the partitioning methods such as k-means algorithm. The partitioning process is iterative and heuristic. The algorithm starts with k initial representative objects for the clusters, and then iteratively recalculates the clusters, and their means until convergence is achieved.

All classes, methods and relation between them are computed. Then PACO algorithm is used to identify a partition of a software system in which the methods belonging to a crosscutting concern should be grouped together.

V. Methodology

Data processing is often neglected, but it is an important step in data mining process. If the redundant information present is noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation includes cleaning, normalization, transformation, feature extraction and selection.

The sequential logic of program can be expressed as a finite number of single-entry, single-exit control structures. The behavior of every control structure in a program can be extracted and composed in step wise process based on algebra function that traverses the control structure hierarchy. This process detects the code clones that occur in control structures, such as for, while and do.

This takes two steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence constrain are used to form the rules.

VI. Algorithm

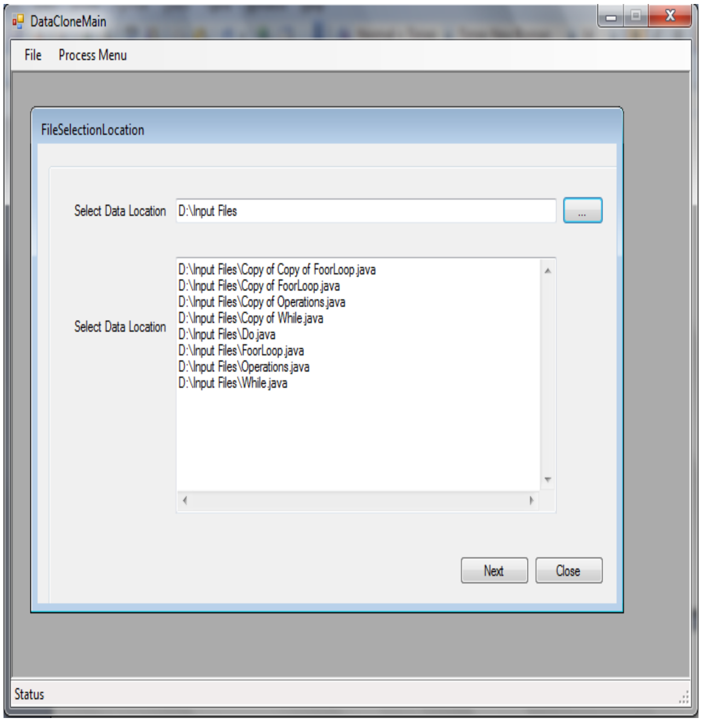
Step 1: For each frequent item.
Step 2: Obtain the split set.
Step 3: List all the statements that are the instance of simple cloned set.
Step 4: For each statement if frequent item is a subset of simple clone.
Step 5: Assign an index value for different statements.
Step 6: Output the result containing the index value and the number of occurrences of each index.

VII. Approach

Clone detection is a process in which the input is a list of source files and the output is a list of duplications. In this process it finds out the duplications that occur in the control looping statements of the source files. It takes the following steps:

A. Extraction

Examining the small piece of code, for which the amount of work is reduced by the clone detector and makes the result more relevant. The clone extractor is responsible for clone restrictions. It extracts structural clones from the source code. The code clone is found for the control structures or the structured statements such as for, while and do statements that reside with in source code. The extract function, extracts a set of all structured statements. Each clone is extracted only once, but if the clones are located at nesting, then the clones is listed twice.

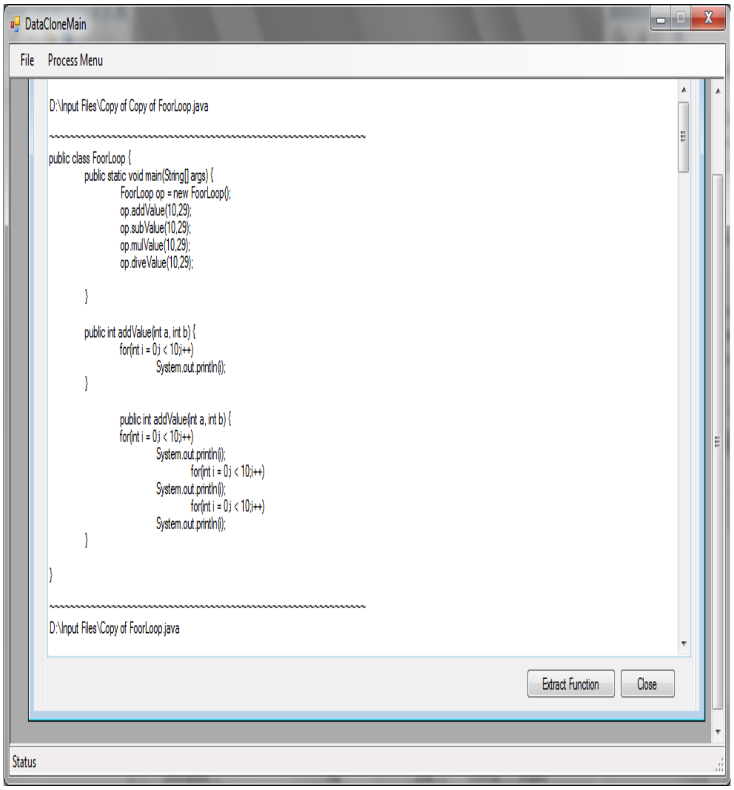


Select Data Location D:\Input Files

Select Data Location

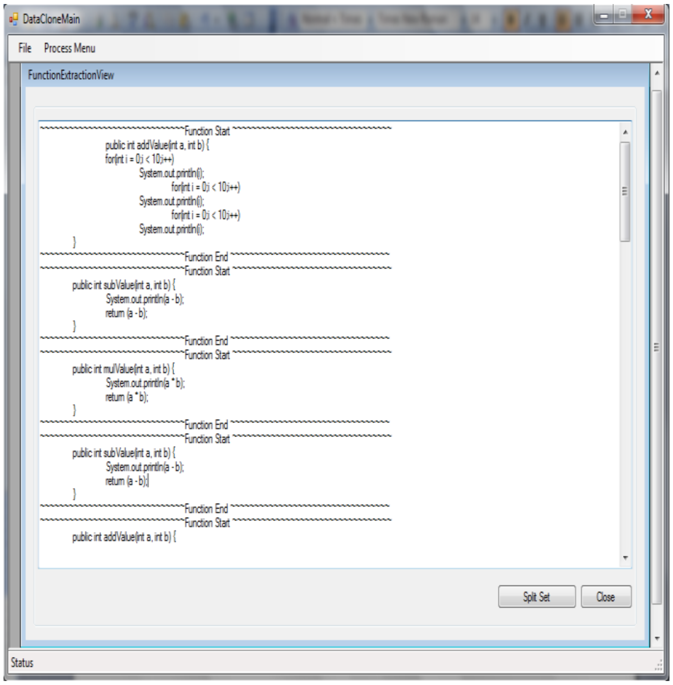
```
D:\Input Files\Copy of Copy of FoorLoop.java
D:\Input Files\Copy of FoorLoop.java
D:\Input Files\Copy of Operations.java
D:\Input Files\Copy of While.java
D:\Input Files\Do.java
D:\Input Files\FoorLoop.java
D:\Input Files\Operations.java
D:\Input Files\While.java
```

Fig.1 Selection of Source File



```
D:\Input Files\Copy of Copy of FoorLoop.java

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

public class FoorLoop {
        public static void main(String[] args) {
                FoorLoop op = new FoorLoop();
                op.addValue(10,29);
                op.subValue(10,29);
                op.mulValue(10,29);
                op.diveValue(10,29);

        }

        public int addValue(int a, int b) {
                for(int i = 0;i < 10;i++)
                        System.out.println(i);
        }

                public int addValue(int a, int b) {
                for(int i = 0;i < 10;i++)
                        System.out.println(i);
                                for(int i = 0;i < 10;i++)
                        System.out.println(i);
                                for(int i = 0;i < 10;i++)
                        System.out.println(i);
        }

}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

D:\Input Files\Copy of FoorLoop.java
```

Fig.2 Selection of Input File

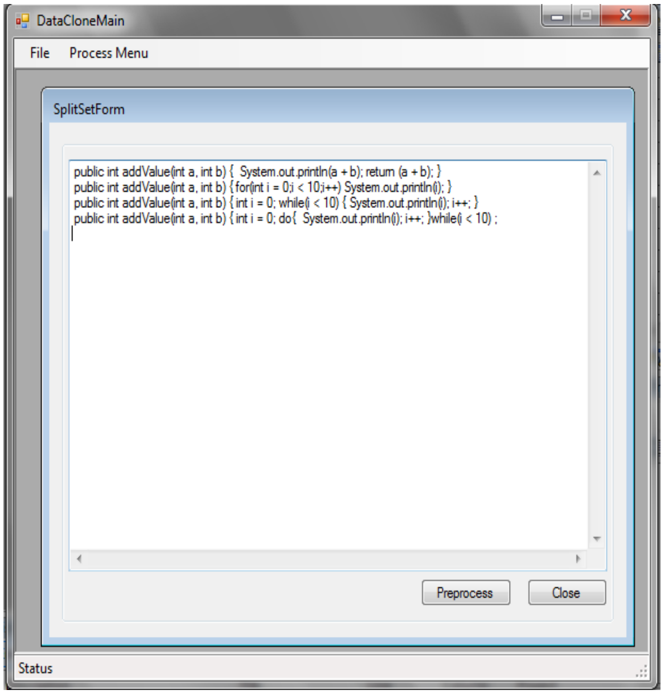


```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function Start ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            public int addValue(int a, int b) {
            for(int i = 0;i < 10;i++)
                  System.out.println(i);
                        for(int i = 0;i < 10;i++)
                  System.out.println(i);
                        for(int i = 0;i < 10;i++)
                  System.out.println(i);
      }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function End ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function Start ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      public int subValue(int a, int b) {
            System.out.println(a - b);
            return (a - b);
      }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function End ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function Start ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      public int mulValue(int a, int b) {
            System.out.println(a * b);
            return (a * b);
      }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function End ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function Start ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      public int subValue(int a, int b) {
            System.out.println(a - b);
            return (a - b);
      }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function End ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Function Start ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      public int addValue(int a, int b) {
```

Fig.3 Function Extraction

B. Split set

Once the entire structural clone has been extracted, they split the clones from the source code. Here we have considered three code fragments for, do and while to be extracted. The extracted files are split from the source code and are placed in split set form.

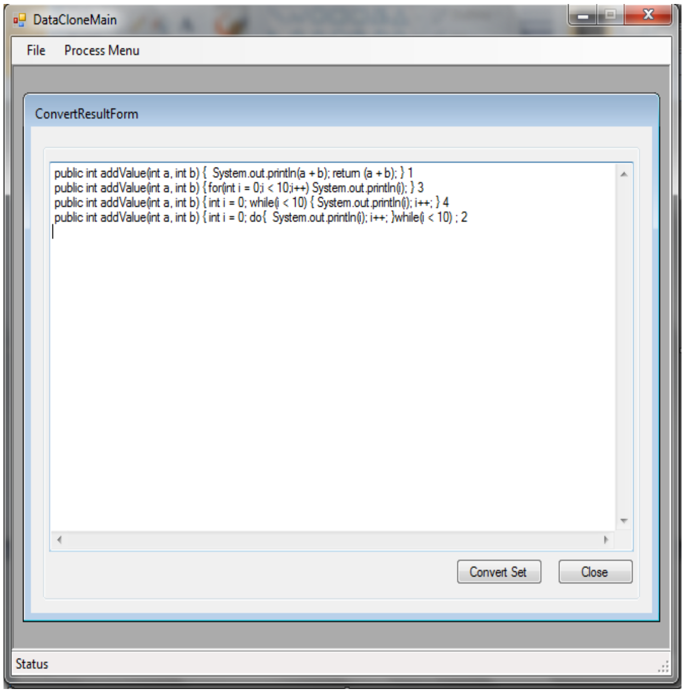


```
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); }
public int addValue(int a, int b) { for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) { int i = 0; while(i < 10) { System.out.println(i); i++; }
public int addValue(int a, int b) { int i = 0; do{  System.out.println(i); i++; }while(i < 10) ;
```

Fig.4Extracted Splitset

C. Conversion

After the split function, we assign values for each statement that is been split. For example, the structures *for* statement is assigned a value 3, the *while* statement takes the value 4 and the *do* statement takes the value 2 etc.



```
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } 1
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); } 3
public int addValue(int a, int b) {int i = 0; while(i < 10) { System.out.println(i); i++; } 4
public int addValue(int a, int b) {int i = 0; do{  System.out.println(i); i++; }while(i < 10) ; 2
```

Fig.5 Conversion of each statement with index value

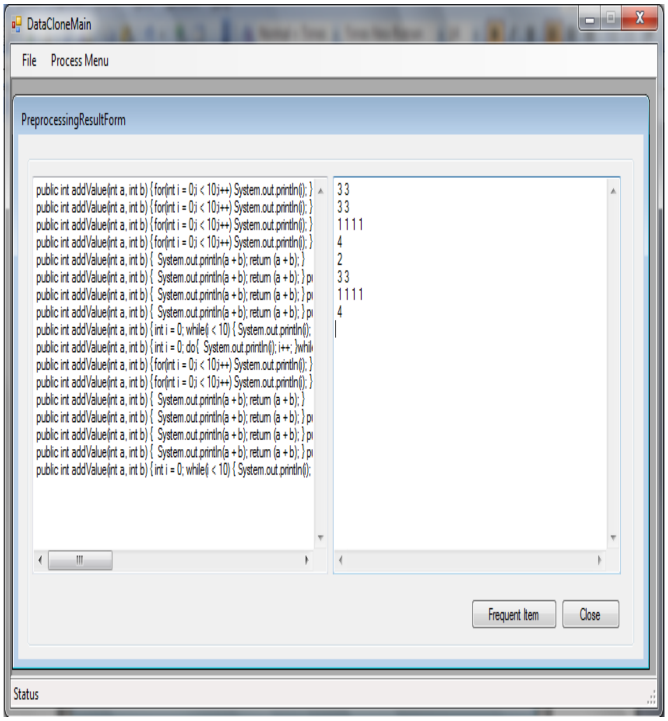
D. Preprocessing

On beginning of any code clone detection approach, the besieged source code is partitioned and the domain of comparison is determined. The source code is partitioned into a set of disjoint fragments called source units. These units are the source fragments that are involved in direct clone relation to each other. These units do not contain any particular order in source code, where the matching cannot be aggregated. In our example it takes source lines of control structures such as *for, while and do* statements. These source units results in simple cloned set.

The comparison units are derived from the source units i.e simple cloned set. Comparison units are derived based on syntactic structure of source code and they are ordered within their corresponding source units. Source units are encoded as tuple as one indicating the statements of source units and the other represents the index occurrence. For example, *for(int i=0;i<10;i++)* represents the statement and the value '*3'* represent the index occurrences.

Mining all source units based on minimum support count or support, it indicates the minimum number of files that contains simple clone set and is reported as frequent. As the support value taken here is 2, it groups the simple cloned set even if it occurs within two files.

The output from this list is a frequent itemsets with support count, indicating the number the statements of simple cloned set that are frequent.

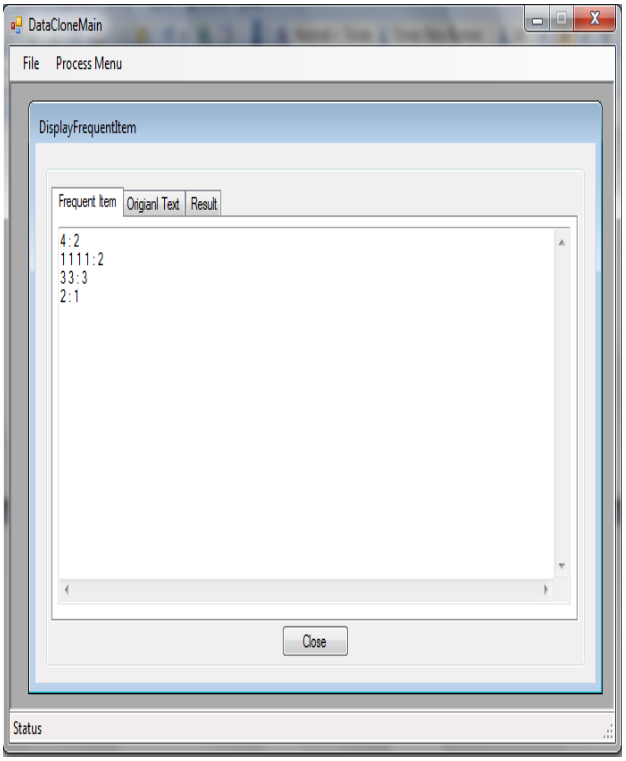


```
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); }
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {int i = 0; while(i < 10) { System.out.println(i);
public int addValue(int a, int b) {int i = 0; do{  System.out.println(i); i++; }whil
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {for(int i = 0;i < 10;i++) System.out.println(i); }
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); }
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {  System.out.println(a + b); return (a + b); } pu
public int addValue(int a, int b) {int i = 0; while(i < 10) { System.out.println(i);
```

```
3 3
3 3
1 1 1 1
4
2
3 3
1 1 1 1
4
```

Fig.6 Preprocessing Form

In fig. 7it counts the number of statements that are similar for each and every control structures. It displays the index of frequent irem set and the number of occurance of each index.



```
4:2
1111:2
33:3
2:1
```

Fig 7 Frequent Itemseet and its Occurrences

IV. CONCLUSIONS

Code cloning is a problem in software maintenance and a lot of work has been done in detecting duplicate fragments. These are termed to be simple clones. The higher level similarities are termed to be structural clones. As lot of works has been proposed till now, there is no clone detection technique for control structure. Detecting these clones brings more benefits towards maintenance and reuse.

We identify and quantify the similarity in source code that is important for many application of code detection. This paper present a clone detection technique based on data mining approach. This paper detects the code clones that occur in control structures such as for, while and do statement.

Since the clones are computed using data mining, the results are intuitive and interesting type 3 clone. This yields better result in efficient and scalable mining of association. There is a considerable future work to be done, that is calculating the timing that is taken for finding the tokens.

ACKNOWLEDGMENT

We would like to thank fellow colleagues for valuable insights and comments that helped to improve this paper. And this work was supported by my parents and Mr.P.RamKumar, Mr.D.Siddharthan for their grand support and making the work successful.

REFERENCES

[1] Chen B He J Pellicer, S & Pan, Y, " *Protein Sequence Motif Super-Rulr-Tree (SRT) Structured Constructed bt Hybrid Hieratchical K-Means Clustering Algorithm".*2008.

[2] Chen B Pellicer, S.Tai, P.C.Harrison, R & Pan, Y,"*Efficient Super Granular SVM Feature Elimination (Super GSVM-FE) model for protein sequence motif information extraction*". 2008.

[3] Breu, S. and Zimmermann, T."*Mining Aspects from Version History*". September 2006.

[4] He, L. and Bai H, "*Aspects Mining using Clustering and Association Rule Method*", February 2006.

[5] Mircea Lungu, Adrian Kuhn, Tudir Girba, and Micheele Lanza, "*Interative exploration of semantic clusters", 2005.*

[6] Adrian Kuhn, Stephane Ducasse and Tudor Girba, "*Semantic Clustering: Exploiting source code linguistic information*", 2006.

[7] L.Qu and D.Liu., "*Aspect Mining Using Method Call Tree*", 2007.

[8] G.Serban and G.S.Moldovan.,"*A Graph Algorithm for Identification of Crosscutting Concerns*", 2006.

[9] Z.Li, S.Lu , (2006) *CP –Miner. Finding Copy-Paste and Related bugs in Large-Scale Software Code*, IEEE TSE.

[10] Gayathri Devi.D, .Punthavalli.M, (2011) *A Detecting structural Clones* , In IEEE International Conference on NetworkCommunication and Computer.

[11] Gayathri Devi.D ,.Punithavalli.M (2011) *A Method for Detecting Code Clones,* In IEEE International Conference on Electronics Computer Technology..

[12] Lagu.B,K.Kontogiannis,M.Balazinska,,.Merlo, M.Dagenais, (2000)," *Advnced clone analysis as a basic for object oriented syatem refactoring".* In proc. Working conference on reverse engineering [WCRE] pages, 98-107, IEEE computer society press.

[13] Girard,Juergens,J.F.Deissenboeck,B.Hummel, ,B.sehaetz,S.Wagner,S.Teuchert ,. (2008) "*Clone detection in automotive model based development."* In proceeding of the international conference on software engineering, IEEE computer society press.

[14] Gayathri Devi.D ,.Punithavalli.M (2011) "*Comparison and Evaluation on Metrics Based Approach For Detecting Code Clone*", In IJCSE, Nov 2011.

[15] Baker.B.S, (1995)" *On finding Duplication and Near- Duplication in Large Software Syatem*", Proc. Second IEEE Working Conf. On Reverse Eng., pp. 86-95.

[16] Baxter.I.D, a. Yahin, L.Moure, M.Sant' Anna, and L. Bier, "*Clone Detection Using Abstract syntax Trees*", (1998) Proc. Of IEEE Int'l Conf. on Software Maintenance (ICSM) '98, pp. 368-377, Bethesda, Maryland.

[17] Ducasse.S, M.Rieger, and S.Demeyer. (1999) "*A Language Independent Approach for Detecting Duplicated Code*", Proc. Of IEEE Int'l Conf. on Software Maintenance (ICSM) '99, pp.109-118. Oxford, England

[18] Choh Man Teng, "*A Comparison of Standard and Interval Association Rules*", In Proceedings of the Sixteenth International FLAIRS Conference, *pp.: 371-375, 2003.*

[19] *Suzuki Kaoru, "Data Mining and the Case for Sampling,"* SAS Institute Best Practices Paper, SAS Institute, 1998.

[20] Soo, J., Chen, M.S., and Yu, P.S., 1997, "*Using a Hash-Based Method with Transaction Trimming and Database Scan Reduction for Mining AssociationRules*" IEEE Transactions On Knowledge and Data Engineering, Vol.No.5. pp. 813-825.

[21] En Tzu Wang and Arbee L.P. ChenData," *A Novel Hash-based approach for mining frequent itemsets over data streams requiring less Memory space*"Data Mining and Knowledge Discovery, Volume 19, Number 1, pp 132-172.

[22] Wojciechowski, M., Zakrzewiez, M., *Dataset filtering Techniques in Constraint based Frequent pattern Mining*, Lecture Notes in Computer Science, Volume2447, 2002, pp77-83.

[23] Tien Dung Do, Siu Cheng Hui,Alvis Fong, *Mining frequent itemsets with category Based Constraints. Lecture Notes in Computer Science*, Volume 2843,2003, pp226-234.

[24] Das, A., Ng, W.K., and Woon, Y, K. 2001. *Rapid association rule mining*. In the proceedings of the tenth international conference on Information and knowledge management.. ACM press, 474-481.

[25] Rakesh Agarwal, Ramakrishnan Srikant," *Fast Algorithms for Mining Association Rules*" 20th Intl Conference on VLDB, Santigo, Chile, Set.1994.

[26] Thevar., R.E; Krishnamoorthy, R," *A new approach of modified transaction reduction algorithm for mining frequent itemset*", ICCIT 2008.11th conference on Computer and Information Technology.

[27] W. Evans, C. Fraser, and F. Ma. "*Clone detection via structural abstraction*". In Proc. of the 14th Working Conference on Reverse Engineering, pages 150–159, 2007.

[28] R. Koschke. "*Frontiers of software clone management"* . In 24th IEEE International Conference on Software Maintenance, pages 119–128, 2008.

[29] R. Koschke, R. Falke, and P. Frenzel. *"Clone detection using abstract syntax suffix trees"*. In Proc. 13th Working Conference on Reverse Engineering, pages 253–262, 2006.

[20] Reto Geiger, "Evolution impact of code clones", Departmant of Informatics, October 2005.

[31] C. K. Roy and J. R. Cordy. "*A survey on software clone detection research*" Technical report, Queen's University at Kingston, Ontario, Canada, 2007.

[32] C. Kapser and M. W. Godfrey. *Cloning considered harmful considered harmful*. Presented at WCRE '06: Proceedings of the 13th WorkingConference on Reverse Engineering

[33] Thummalapenta S, Cerulo L, Aversano L, Di Penta M. *An empirical study on the maintenance of source code clones*. Empirical Software EngineeringMar2009.

Authors

1.D.Gayathri Devi M.C.A., M.Phil., She is currently a Assistant Professor Department of computer Science at Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu and pursuing her Ph.D Karpagam University, Coimbatore. She has 8 years of teaching experience and presented 3 papers in National and International Conferences and produced 2 M.Phils so far

2. Dr.M.Punithavalli Ph.D, She is currently a Director and Head, Department of Computer Application at Sri Ramakrishna Engineering College, Coimbatore ,Tamilnadu She has 21 years of teaching experience and presented National and International Conferences and Produced more than 18 M.Phil and more than 7 Ph.D so far