IOSR Journal of Engineering
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719,
Vol. 2, Issue 12 (Dec. 2012), ||V3|| PP 45-47

Abstract Syntax Tree Based Clone Detection for Java Projects

Tahira Khatoon, Priyansha Singh, Shikha Shukla
(*CSE, Institute Of Technology and Management, Gida Gorakhpur, India)*

***Abstract*-A large amount of unavoidable money is spent on the maintenance of any existing software systems. Software maintenance cost is generally higher than the development cost therefore reducing maintenance cost is a necessary task of today's software industries. Software system includes similar bugs at different places which makes system inefficient n takes extra time and effort to deal with them. As a result the cost of software maintenance activities is increased. By detecting duplicate code fragments we can reduce the time and effort as well as maintenance cost. By analyzing the source code of a given software system, code clones can be detected. By this paper an abstract syntax tree based clone detector for java system is designed and implemented easily. There are various technique which can be used to find the clones This paper examines a software engineering process to create an abstract syntax tree based clone detector for java projects.**

Keywords: *clone detection, abstract syntax tree, java, software, maintenance*

I. INTRODUCTION

The presence of code clone that are similar in syntax and semantics are generally considered to be an indication of poor quality of software. A software system usually contains 7- 23% clone code that may be introduced by two ways-either intentionally or accidently. Sometimes, codes are intentionally introduced by programmers by copy/paste the code fragments and use it either with some modification or without any modification. While sometimes code clones are accidently be injected by the programmers due to the reason that they need to comply the to some well known solutions or best practices such as-exception handling, user interface design etc.

Code clone existing in programme are of four types, that are-**type 1, type 2, type 3, type 4.**

Type 1-This type includes the code clones which are exact copy of a code fragment except white spaces and comments.

Type 2 – This type includes code clones which are syntactically equal with some modification like renaming variables and methods.

Type 3-This type of code clones are copied code fragments with slightly syntactical difference and that may contain extra statements etc.

Type 4-Type 4 code clones are functionally equivalent but differ in implementation.

The main goal of this paper is to apply the theory of clone detection and to create a prototype that can implement AST based code clone detection on java projects. The implementation should be able to visualise duplicate code fragments in such a way that it would be able to help the user to distinguish code fragments. By colourizing the duplicate code fragment using the same colour codes and linking the code clones which would make it easier to user to see the similarities. Another way to visualize the result could be by using tree maps. The use of tree maps is an efficient way to visualize code clones.

II. PROBLEM STATEMENT

When software programmers implement software programme, similar code fragment may appear in different parts of the software system. Sometimes it is done intentionally and sometimes clone codes occur accidently. Most of the time, codes are reused by the programmers for the well known solutions. When clone code is intentionally introduced in software programme, the programmer may achieve time efficiency but loses control of the implemented software system.

Apart from above mentioned problem there are some more problems that can occur due to code cloning. These are-

- Probability of bug propagation is increased. If a code segment contains any bug and is reused by copying and pasting with or without any modification then the bug of the original segment remains in all the pasted segments.

- Introduction of new bug can also take place when the programmer only uses the structure of the duplicated fragment and it is his responsibility to implement the code according to the current need. This process can lead to introduction of new bug in the system and makes it error prone.
- Implementation with lack of good inheritance structure and abstraction may result in bad design and also it makes software less maintainable.
- The main problem related with code cloning is that it increases the maintenance. If there exist any bug in the cloned code segment then all other similar fragments are also investigated for the correctness of bug. Therefore maintaining a piece of code, duplication multiplies the work load.

III. EXISTING TECHNIQUES FOR DETECTION OF CODE CLONES

1.1 Line based technique-This technique detect code clone by comparing source code on line before comparison tabs and white spaces are eliminated. This method was used in early days. The detection accuracy is very low because it cannot detect code clones written in different coding styles. For example-'{' position of if-statement or while-statement. And also it cannot detect code clones using different variable names.

1.2 Metric based techniques-In this technique, firstly the source code is divided into different functional units and then metrics for each unit is defined. Units with similar metric value are defined as code clones. Unfortunately this technique performs very poorly due to the high sensitivity of most metrics to minor edits therefore the partly similar units are not detected.

1.3 Token based techniques-This technique came in existence after simple string matching techniques, it was the first approach to deal with source code similarity detection using sequence matching algorithms. In this technique, the source codes are firstly tokenized to provide sequence of tokens that are input to the string matching algorithms. After tokenization token sequences of source code are then compared, and identify the similar subsequences as code clones. This approach is not scalable either for the comparison of a set of projects (all pairs should be compared) nor for the search of similarities of a single project against a database of projects.

1.4 Tree based techniques-In order to use the syntactic properties of the programs, tree based approach considers the syntax trees of the compilation units which are obtained through parsing. In this an abstract version of trees is considered so that better recall can be achieved. In this technique firstly, source code is parsed and after that abstract syntax tree is constructed, then the subtrees which are similar are identified as code clones (differences of code style and variable names are eliminated).

1.5 PDG (Programme dependency graph) based techniques-In this approach, control and data flow dependency of a function may be represented by a program dependency graph; clones may be identified as isomorphic sub graphs. The detection accuracy is very high as it can detect code clones which are not detected in other methods. Such as semantic clones, reordered clones. But it require complex computations therefore, it is very difficult to apply to large softwares.

IV. PROPOSED TECHNIQUE

The above techniques does not provide good solution for the detection of code clones because of the limitations which are mentioned above so to cope up with these limitations an AST based technique was introduced to provide better solution.

In order to find code clones using AST we need to compare each subtree to each other subtree in AST. Computing the similarities of all subtree pairs are not efficient, which complexity of computation is $O(N^3)$,where N is number of nodes in AST. To increase the scalability of the approach a hash function is used that partitions the AST into similar subtrees. If there are two subtrees whose similarity exceeds the threshold then these subtrees are called clones. Hashing function is used to hash subtrees into some buckets if the mass of the subtree exceeds the mass threshold(implemented by basic algorithm given below).The single subtree clone were detected by using hashing function but the subtree sequence clone cannot be detected. To overcome from this problem a list structure is built where each list is associated with a sequence in the programme and stores the hash codes of each subtree element of associated sequence. The algorithm for implementing this is given as below-

4.1 Basic Algorithm

```
Clones=ф
For each subtree i
        If mass(i)>=Threshold
        Then hash i to bucket
```

```
For each subtree i and j in the same bucket
        If Compare tree(i,h)>SimilarityThreshold
        Then For each subtree s of i
                If IsMember r(clone s,s)
                Then RemoveClonePair(clone s,s)
        For each subtree s of j
                If IsMember r(clone s,s)
                Then RemoveClonePair(clone s,s)
        AddClonePair(clones,i,j)
```

4.2 Sequence Detection Algorithm

```
1.      Build the list structure s describing sequences
2.      For k=MinimumSequenceLengthThreshold to MaximumSequenceLength
3.      Place all subsequences of length k into buckets according to subsequence hash
4.      For each subsequence i and j in the same bucket
                If CompareSequences(i,j,k)>SimilarityThreshold
                Then{ RemoveSequenceSubclonesOf(clones i,j,k)
                        AddSequenceClonePair(clones i,j,k)
                    }
```

Clone detection by using abstract syntax tree and comparing each subtree or subtree sequence results in finding out exact and near miss clone.

V. CONCLUSION

In this paper, first of all problem domain is studied and discussed about the types of the clone code generated. According to expected benefits an implied importance of code clone detection a proper solution is selected from various techniques. In which we have proposed abstract syntax tree based clone detection technique using hashing function such that this function can hash a statement to a key which implied parse tree to be simple.The hashing function can hash the same type statements into the same key no matter how many operands they have. Therefore we can detect the clones that are larger than other techniques detected. At the end of this survey, all the goals are achieved and the criteria met as a result we can imply that the problem is solved.

REFERENCES

1). Yoshiki Higo,Toshihiro Kamiya, Shinji Kusumoto and Katsuro Inoue:"Refactoring Support Based on Code Clone Analysis"

2). Chanchal Kumar Roy and James R. Cordy:" A Survey on Software Clone Detection Research" , September 26, 2007

3). Debarshi Chatterji, Jeffrey C. Carver, Nicholas A. Kraft :"Claims and Beliefs about Code Clones: Do We Agree as a Community?A Survey"

4). Hoan Anh Nguyen, Tung Thanh Nguyen, Nam H. Pham, Jafar M. Al-Kofahi,and Tien N. Nguyen :"Accurate and Efficient Structural Characteristic Feature Extraction for Clone Detection"

5). Rainer Koschke, Raimar Falke, Pierre Frenzel:" Clone Detection Using Abstract Syntax Suffix Trees"

6). Toshihiro Kamiya, Shinji Kusumoto, *Member, IEEE,* and Katsuro Inoue, *Member, IEEEE* :"CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code"

7). Huiqing Li, Simon Thompson:" Clone Detection and Removal for Erlang/OTP within a Refactoring Environment"

8). Randy Smith and Susan Horwitz :"Detecting and Measuring Similarity in Code Clones"

9). Chung Yung and Che-Wei Wu :"A New Approach to Parameterized Clone Detection Using Abstract Syntax Tree"

10). Peter Bulychev, Marius Minea :"Duplicate code detection using anti-unification"

11). Miryung Kim, Vibha Sazawal, David Notkin , Gail C. Murphy :"An Empirical Study of Code Clone Genealogies"

12). Denis Bogdanas And Alexandru Archip :"Code Quality Assurance By Detecting Clone Expression Removal Opportunities"

13). Michel Chilowicz, Etienne Duris and Gilles Roussel :"Syntax tree fingerprinting: a foundation for source code similarity detection"

14). Elmar Juergens, Florian Deissenboeck, Benjamin Hummel:" CloneDetective – A Workbench for Clone Detection Research"

15). Peter Bulychev , Marius Minea :"An evaluation of duplicate code detection using anti-unification"