IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-4, Issue-11 E-ISSN: 2347-2693

Clone Detection Using Abstract Syntax Trees

L Sridevi[*1] and R.Kannan[2]

[1*] Department of Computer Science, Bharathiar University Arts and Science College, Gudalur, The Nilgiris. India. *e-mail: sricharu28@yahoo.in*
[2] Department of Computer Science, Sri Ramakrishna Mission Vidhyalaya Arts&Science College, Coimbatore. India. *e-mail: ramadosskannan@gmail.com*

Available online at: www.ijcseonline.org

Received: Oct/23/2016 Revised: Oct/30/2016 Accepted: Nov/20/2016 Published: Nov/30/2016

***Abstract*— Clones are the piece of Software, which is creating from the copy of the original software. To be more specific, the idea behind software cloning is to create a new software that replicates the aspect and usefulness of the original software in possible. It is important to understand that cloning does not have to involve any source code in the original software. Software Cloning typically occurs in the source code for the original software is not available. In a result, software cloning does not imply source code copying. Since software cloning goes way beyond simply executing a similar user interface. The goal in cloning is to create a new software program that mimics everything the original software does and the way in which it does.**

Keywords- Code clone, Syntatic method, Clone detect, Clone removal, Abstract Syntax Trees(AST)

I. INTRODUCTION

Code duplication falls in the development of large software systems. The improvised form of replication consists in copying, and eventually modifying, a block of existing code that apply a piece of required functionality. Duplication segments are called clones and proceed of copying by slight modifications is also called cloning. Code cloning or copying code fragments and making minor, non–functional alterations, is identified for developing software systems dominant to duplicated code segments or code clones. using the copy and- paste attributes than writing instructions from scrape or applying correct replicating mechanisms, based on invocation or inclusion [4].

Code cloning occurs for other variety of reasons: the short term cost of forming the proper abstractions may be heavier the cost of replicating code and takes place when the developer is alert to the extant of code performs the functionality similar to, or the same as, the functionality required [1].

II. CLONE ANALYSIS

Clone Detection is an advanced analysis engine that quickly detects duplicate patterns within code and allows you to find code clones and difficult-to-detect copy-paste bugs.

A. TOKEN BASED VS AST

The analysis based token-suffix trees provides assorted advantages than other techniques. It measures well due to linear complexity in both time and space, which makes it very attractive for large systems. Moreover, no parsing is necessary and, hence, the code may be even incomplete and incorrect order of the code. Another advantage for a tool builder is that a token-based clone detector can be adjusted to a new language in very short time. As opposed to text-based techniques, this token-based analysis is independent of layout (parameters are not quite true for Baker's technique, which is line based; however, if one uses the original string based technique, line splitting do not have any effect). Token-based analysis are more authentic than metrics that concludes abstractions of a piece of code. since, the level of thickness is typically whole functions rather than individual statements.

For instance, the two program snippets left and right in Listing 1 are considered a clone by a token-based analysis because their token sequence is identical: return id ; } int id ( ) { int id; Although from a lexical point of view, these are in fact rightful clones, a maintenance programmer would hardly consider this finding useful [2].

The Listing 1 as follows

```
r e turn r e s u l t ; r e turn x ; }
}
i n t foo ( ) { i n t b a r ( ) { i n t y ;
i n t a ;
```

[*]Corresponding Author:
L.Sridevi
e-mail: sricharu28@yahoo.in mob: +91 9047558015

B. SYNTATIC ANALYSIS:

Syntactic clones can be found to some extent by token based techniques if the candidate sequences are split in a post processing step into ranges where opening and their corresponding closing tokens are completely implemented in a sequence. For example, by counting matching opening and closing brackets, we could exclude many spurious clones such as the one in Listing. programming languages have many types of establishing tokens after brackets. If, then, else, and end if, constitutes syntax delimiters in Ada. In particular, end if is an interesting example as two continuous tokens form one delimiter and both individual delimiters in syntactic contexts. If one wants to handle these delimiters reliably, one is about to start imitating a parser by a lexer.[3,5,8].

III. CLONE DETECT PROCESS USING AST's

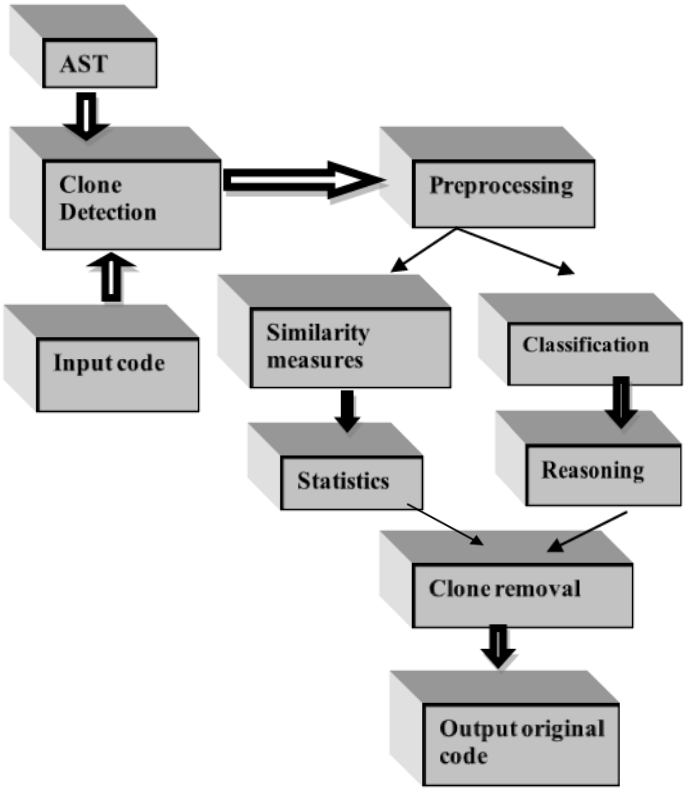

Figure 1: Architecture Diagram

As a first step in the clone detection process, the source code is parsed and an AST is produced . Three main algorithms are applied to find clones. The basis of the first algorithm is the Basic algorithm is to detect sub-tree clones. The second one is sequence detection algorithm which concerned with the detection of variable-size sequences of sub-tree clones. It is also used essentially to detect statement and declaration sequence clones. The third algorithm focus in more complex near-miss clones by seeking to generalize other clones. The resulting detected clones can then be pretty printed [6,7,10].

IV. CLONE REMOVAL

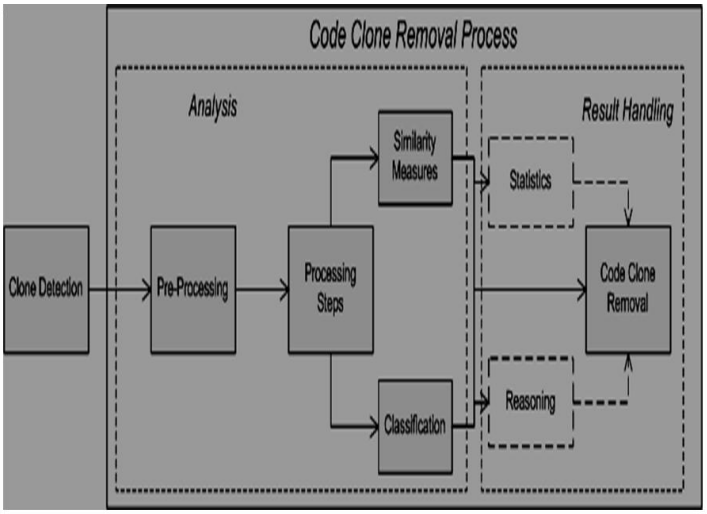

Figure 2: General Structure of the Code Clone Removal Process

Code clone removal is two-staged. In the first stage, a detailed analysis of detection of code clones is performed using the abstract syntax tree. This clone detection presents simple and practical methods for finding exact and near miss clones over arbitrary program segments in source code by using abstract syntax trees. In the second stage, we focus on how the results of stage one can be presented in order to guide an interactive refactoring/clone removal process [6,9].

V CONCLUSION:

The clone detection method is implemented using abstract syntax trees (ASTs), which for finding exact and near miss clones for arbitrary fragments in the source code. Since detection done in the program structure. clones can be factored in the source using standard transformational methods.

The approach is based on variations of methods for compiler common sub expression elimination using hashing. The method is straightforward to implement using standard

parsing technology which detects clones in arbitrary language. It also constructs and computes macros that removes the clones without affecting the operation of the program.

REFERENCES

[1] T. Kamiya, S. Kusumoto, and K. Inoue "CCFinder: a multilinguistic token-based code clone detection system for large scale source code", IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654 - 670, July 2002

[2] M.Kim, and D. Notkin "Mining Software Repositories (MSR): Using a clone genealogy extractor for understanding and supporting evolution of code clones", Proceedings of the 2005 international workshop on Mining software repositories MSR '05, pp. 1-5, May 2005.

[3] M.Kim, V. Sazawal, D. Notkin, and G. Murphy "An empirical study of code clone genealogies", Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering ESEC/FSE-13, pp. 187-196, September 2005

[4] R.Koschke, R. Falke, and P. Frenzel "Clone Detection Using Abstract Syntax Suffix Trees", Proceedings of the 13th Working Conference on Reverse Engineering (WCRE '06), pp. 253- 262, October 2006

[5] C.K.Roy and J.R. Cordy, NICAD, "Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization" in Proceedings of the 16th IEEE International Conference on Program Comprehension, ICPC 2008.

[6] Mohammed Abdul Bari**. "**Code Cloning: The Analysis, Detection and Removal" in proceedings of International Journal of Computer Applications (0975 – 8887)

[7] R.Koschke, R.Falke and P. Frenzel," Clone Detection Using Abstract Syntax Suffix Trees" in Proceedings of the 13th Working Conference on Reverse Engg. WCRE 2006.

[8] J.Krinke"Advanced slicing of sequential and concurrent Programs Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 464-468, September 2004.

[9] C.K.Roy and J. Cordy. NICAD: Accurate detection of near miss intentional clones using flexible pretty-printing and code normalization. In Proc. 16th IEEE International Conference on Program Comprehension, pages 172–181, 2008.

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

.

Authors Profile

*Ms L Sridevi pur*sed Bachelor of Science from Bharathiar University of Coimbatore, India in the year 2005 and Master of Science from Bharathiar University in year 2009. She is currently working as Assistant Professor in Department of Computer Sciences of Bharathiar University College of Arts& Science of Gudalur, The Nilgiris. India since 2009.

Mr R.Kannan working as a Associate professor in the Department of computer Science of Sri Ramakrishna mission vidhyalaya College of Arts & Science, Coimbatore. India

