ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

DETECTING SIMPLE AND FILE CLONES IN SOFTWARE

***S.Ajithkumar, P.Gnanagurupandian, M.Senthilvadivelan ,**
Final year Information Technology
****Mr.K.Palraj ME, Assistant Professor,**

Dept of Computer Science and Engineering/Information Technology
Sri Vidya College of Engineering & Technology, Virudhunagar.

ABSTRACT:

The objective of this project is to develop a clone detection tool which is useful in software industries to detect the duplicate codes where the duplicate codes cause many problems during the maintenance phase. Our proposed tool has the capability to detect both the simple as well as files clones in software and increase the reusability as well as efficiency of the project files. The tool should be platform independent one.

INTRODUCTION:

A clone relation is defined as an equivalence relation (i.e., reflexive, transitive, and symmetric relation) on code portions. A clone relation holds between two code portions if (and only if) they are the same sequences. For a given clone relation, a pair of code portions is called clone pair if the clone relation holds between the portions. An equivalence class of clone relation is called clone class. That is, a clone class is a maximal set of code portions in which a clone relation holds between any pair of code portions. For example, suppose a file has the following 12 tokens: a x y z b x y z c x y d: We get the following three clone classes:

C1. a x y z b x y z c x y d.
C2. a x y z b x y z c x y d.
C3. a x y z b x y z c x y d.

Note that subportions of code portions in each clone class also make clone classes (e.g., Each of C3 is a subportion of C1). In this paper, however, we are interested only in the maximal portions of clone classes so we only discuss the maximal portions (e.g., C1). The three code portions in C2 constitute a clone class since the third one is not a subsequence of any code portions of C1. In our approach, identifying the clone relation is made for the transformed token sequences in order to extract many clones whose token sequences are slightly modified.

IMPLEMENTATION METHODOLOGY:

Architectural Design

The overall architecture of our proposal has been given as follows:

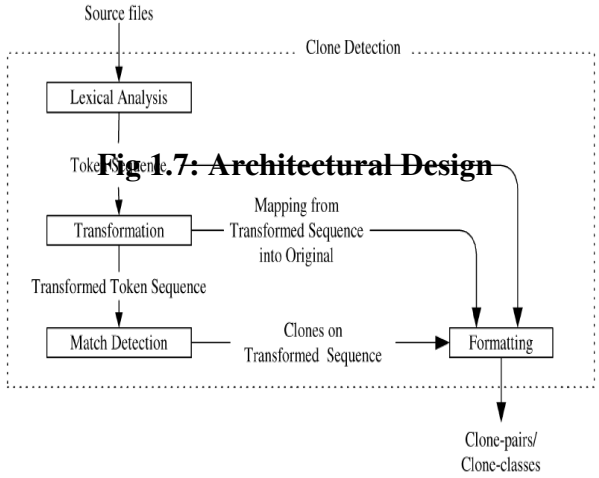


Fig 1.7: Architectural Design

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

This depicts the flow of the project from initial files to output clone pairs or clone lines. This design also indicate input and output of each phase of our tool.

Here, the design of the project is divided into input, process and the output. The process involves the following modules.

They are as follows:

Module 1: Preprocessing

The input program files may contain valuable and unnecessary statements. For cloned detection process, these unnecessary statements must be removed first. So, we removed the unnecessary lines present in the given input files. This module greatly increase the efficiency of the tool.

Examples:

1. Comment line(//xxxx)
2. Header files(import java.xx,…)
3. Reading Statements(scanf(), System.out.println()…)
4. Writing Statements(printf(),…)

Module 2: Lexical analysis

In this module, we perform tokenization on the preprocessed file. Our tool can be able to split the statements into tokens. Tokenization done perfectly on the files follows proper indentation or improper indentation. The tokens are then stored into the temporary file. Here the characters like tab, next line and space are also removed. Number of tokens present in each line is calculated and stored.

Module 3:Trasformation

Here in this module we perform the classification of tokens. Classify the tokens based on return type, keyword, expression, punctuation.

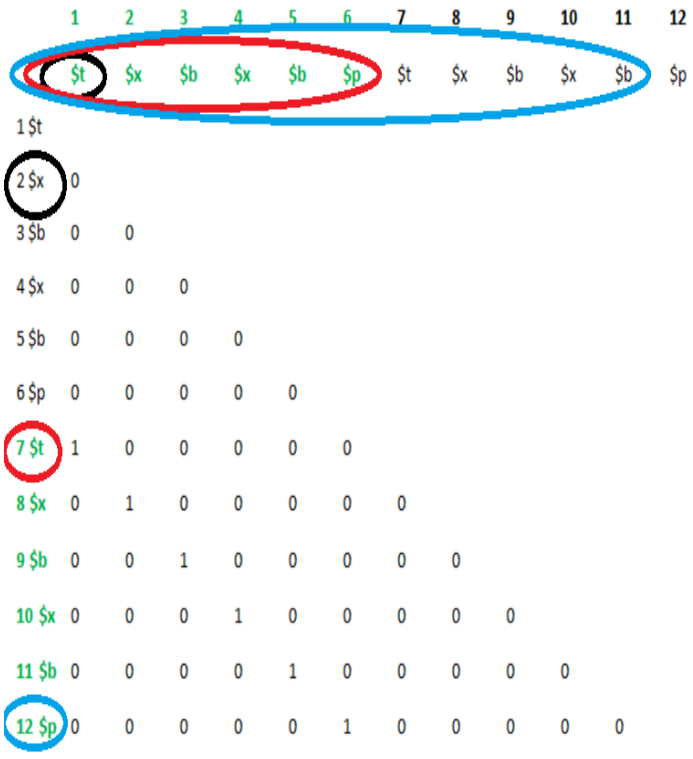
Example:

int
float ------→ $t
double

public
private ------→ $a
protected

Module 4: Match Detection

Here the actual cloned tokens are calculated from the transformed tokens.



In this module compared the transformed tokens in nth position with n-1

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

tokens of the same. If there is match we set the values as 1 else 0.The cloned token is the linear pattern of ones.

Module 5: Formatting

The output of match detection is cloned tokens which is not understandable by the developer or coder so we format the cloned tokens into cloned lines/cloned pairs which is understandable by the user.

RESULT

The below figure shows the Front end of the MNP miner tool. Through this user can choose the input files.

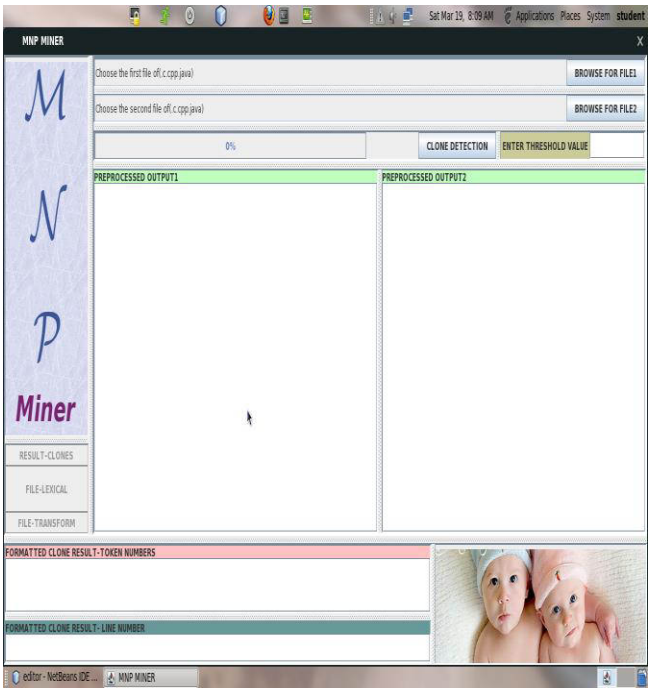
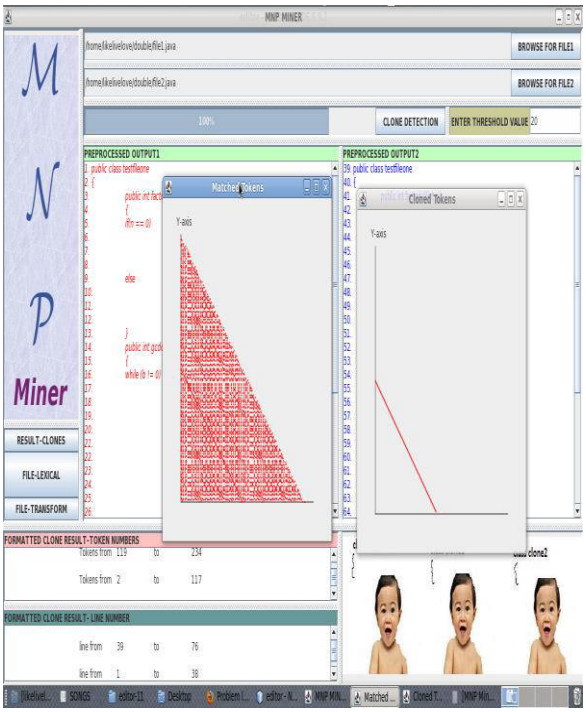


Fig. Front End of MNP Miner

The below figure shows the results after detection of clone for the given files.



ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

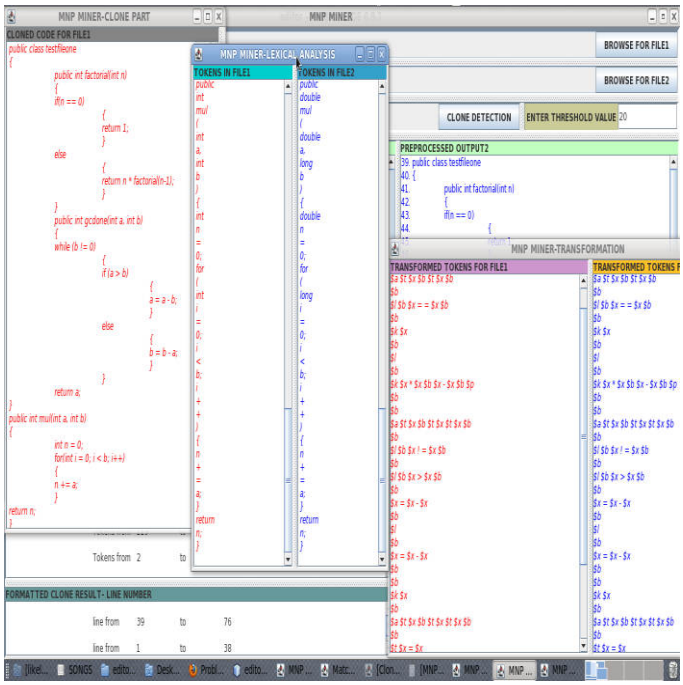
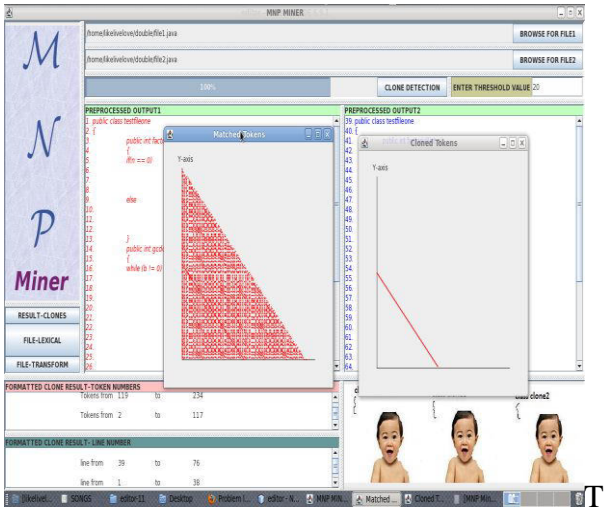


Fig. Intermediate files after processing The below figure shows the matched tokens and cloned tokens in graphical representation



The below figure shows the Cloned files that are opened in a text editor for editing the code.

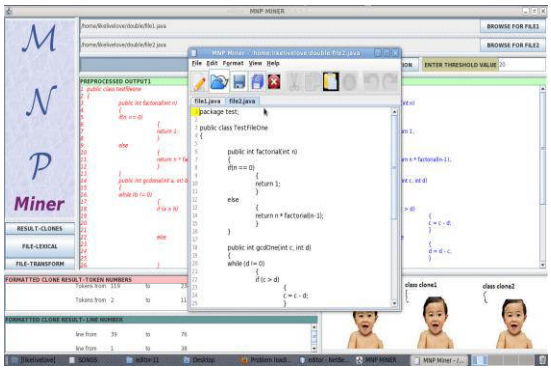


Fig. Cloned files in text editor

The below figure shows the Validation checking of MNP Miner

Display error message if the user doesn't enters the threshold value.Display Error message if user chooses improper file type other than C, C++ and Java.

InvalidFile typeDisplay Error message if character is entered as threshold value.

CONCLUSION

Thus we proposed that by using our clone mining tool we can able to reduce the problems caused due to duplicate codes in the maintenance phase. At the same time we increase efficiency of the project by alerting the developer that these cloned code can be reused or generalized.

Indirectly our project can be used by lab technician or lab staff to monitor whether any students copied their lab exercise. Since the copied students change the variable name and include some printf and scanf statements our tool generalize the variable names and removed the printf and scanf statements.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 15, March 2016

Through our tool the reusability and efficiency of the project is very much since we alert the user about the duplicate code which reduce the space and time of the project as well as increase the problems when the reconstruction is made to the codes.

APPLICATIONS

- Detecting clones in software files.
- Increase the reusability of software files.

REFERENCES

1. Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue. CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. In: Proceedings of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 7, JULY 2002.

2. Hamid Abdul Basit, Stan Jarzabek Detecting Higher-level Similarity Patterns in Programs, ACM SIGSOFT, Sept. 2005

3. Jarzabek, S. and Shubiao, L. Eliminating Redundancies with a "Composition with Adaptation" Meta-programming Technique. In Proc. ESEC-FSE'03, European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, ACM Press, September 2003, Helsinki, pp. 237-246.

4. Java Technology at http://java.sun.com/

5. Mayrand J., Leblanc C., and Merlo E. Experiment on the "automatic detection of function clones in a software system using metrics". In Proc. Intl. Conference on Software Maintenance (ICSM '96), pp. 244-254.

6. XVCL website at : http://xvcl.comp.nus.edu.sg/overview_brochure.php

7. Stan Jarzabek and Shubiao Li "Unifying clones with a generative programming technique".

8. Used College forum for some doubts during implementation of our algorithm in java