ISSN:0975-9646

Detecting Cross-Project Clones Using Frequent Pattern Mining

Ronica Raj[1] , Vishakha Vinod[2]

[1,2]*Student ,*
Sathyabama University ,
Chennai, India.

***Abstract*—Code Clones are the similar program structures in a software system. It is one of the main factors in the degradation of the design and structure of software. It complicates the maintenance and evolution of software. There are two types of code clones generally encountered in a software system i.e. Simple code and Complex code clones. In this paper our aim is to generate a system to detect the code clones in an input source text i.e. source code file or the application as a whole is given as an input to the system where it is processed. Second step includes segregating the clones detected into efficient and inefficient clones i.e. the clones in the input file is segregated into the ones which are useful and the other ones which are waste and need to be discarded. The third step is discarding the inefficient ones i.e. here the inefficient clones which are found are discarded from the code and saved into a temporary folder.**

***Keywords*—Code clones, detection, efficient and inefficient clones, frequent pattern mining.**

I. INTRODUCTION

Code clones or just clones are the similar fragments of codes used within a program or different programs that are maintained by the same entity. These clones arise due to variety of reasons and can be harmful for program as well as to the entity that owns it. Copy and paste of the program and plagiarism are two most common reasons due to which cones arise. It increases the chances of continuation of the bugs which occurred in the clone copied, increases the maintenance cost, it may involve the increment of the number of resources to be used, may involve a large number of modifications which may also introduce some bugs. It may also be helpful if the cloning is done carefully. Detection of code clones can be done using various algorithms like Baker's algorithm, Rabin karp, Syntax trees, visually and using Pattern mining approaches. In our project to generate a system which detects not only the simple clones but also the structural clones, segregating them into the efficient and non-efficient ones which would help to eliminate the ones which are harmful for the program or the application.

II. LITERATURE SURVEY

Cloning is a common phenomenon found in almost all kinds of software systems. It has a negative impact on the maintenance of large software systems. There are four types of code clones which can be identified in a given source code. Type 1 includes identical code fragments except white spaces and comments; type 2 contains structurally and syntactically identical fragments except identifiers, literals, types, layouts and comments; type 3 includes the copied fragments of a source code that can be further modified and type 4 includes those code fragments which are used for executing the same logic but are implemented through different syntactic variants. There are number of commercial and public software systems that have been generated the above listed types of software clones based on many different approaches. Most commonly used approach for this purpose is an Association Mining technique known as Apriori algorithm. This approach involves generation of a large number of candidate sets ($2^{|s|}$ -1) which makes it a very tedious, hard task and time consuming. Other algorithms which were used for clone detection are Karp-Rabin's string matching, Baxter et al.'s variation on AST, Undocumented relies on Java AST. The tools generated using these algorithms in the previous works could only support a few programming languages for example: a tool named PMD using Karp-Rabin's string matching algorithm could only support C, C++, JSP, Java, PHP, Fortran and Ruby. Similarly a tool called CodePro based on Undocumented relies on Java AST could only support Java programming language. After considering all these tools and their results
It was found that PMD finds a large number of small occurrences and also ignores slightly different clones. Bauhaus allows little differences in the clones resulting in larger detected occurrences. These differences were due to the different types of algorithms for detecting duplicated codes and their features.

III. PROPOSED SYSTEM

The proposed method is based on data mining technique called Frequent Pattern Growth algorithm. This approach allows discovering frequent itemsets without generating candidate itemsets. Two main steps involved in this algorithm are building a compact data structure (known as FP tree) and finally extracting the frequent itemsets directly from the tree. The proposed work consists of 5 phases: Authentication, Component selection, Token with clones, Clone detection and refactoring, Testing and Evaluation.

A. Authentication

This is the first phase of the proposed method. In this phase two features are provided namely login and registration. The new users of this application can register before they get access to the internal features of this clone detector and those who are already registered can provide the application with the user name and password as the input and if the input is valid or authenticated, the person is redirected to the other features of the application.

B. Component Selection

After the authentication phase, when the user is redirected inside the application he has to select the feature he needs to use according to his requirements. Here in the proposed system a feature is provided of using a single program or a whole application source code (Fig 1).

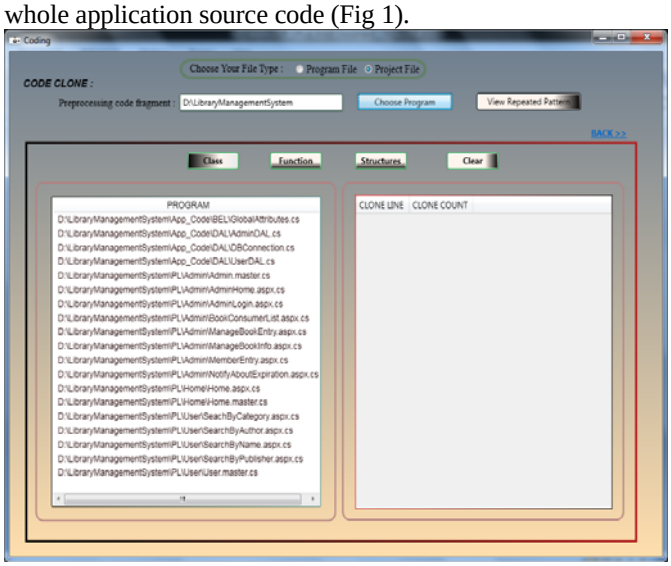

Fig1. Uploading a project

C. Token with clones

After the component selection the next step involved the clone detection is finding the tokens which are repeated in the input source code file.

D. Clone detection and refactoring

After detecting the tokens that occur in the source code the aim is to detect the structural and functional clones (Fig 2) which could either be harmful for developer or may create problems during maintenance of the application. Such clones are removed from the source code and it is restructured to eliminate the negative effects of the detected clones.

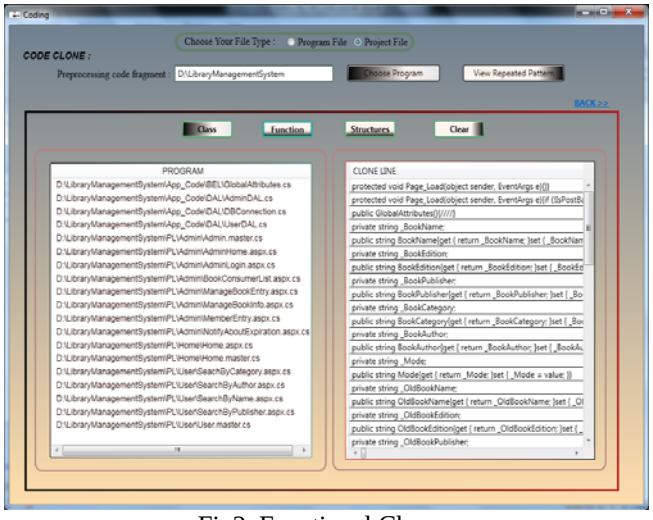

Fig2. Functional Clones

E. Testing and Evaluation

After the clone detection phase and eliminating the clones the source code is tested to verify and confirm its validity. Finally, the analysis helps to refactor the code according to the requirement. Fig 3 represents the flow of the process for the system.

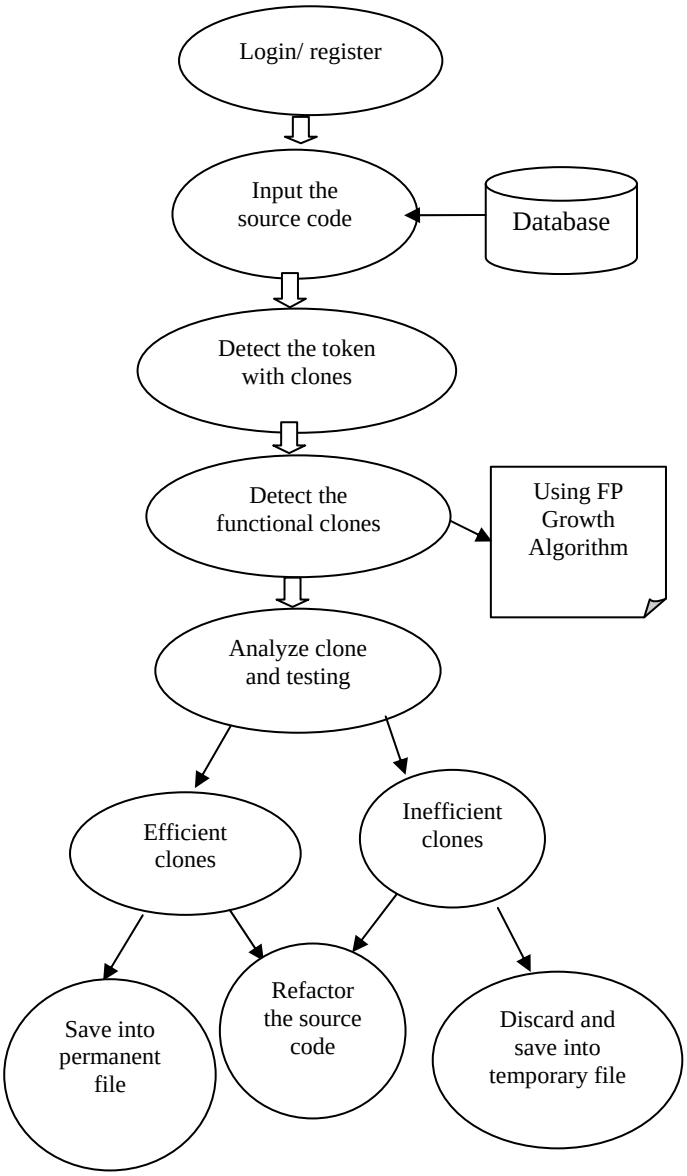

Fig3. Block Diagram

After the analysis of the source code given as input, the result has been displayed in the form of tables. Table 1 and Table 2 represent the unrepeated functions recognized in Hospital Management System and Library Management System respectively whereas the Table 3 and Table 4 represent the repeated functions in Hospital and Library Management System respectively.

Table 1.Unrepeated Functions- Hospital Management System

Fid	Fname	Task
1	Button1_Click(object sender, EventArgs e)	Go to doctor's page
2	appbtn_Click(object sender, EventArgs e)	To get appointment of the specialist doctor
3	pttyddl_SelectedIndexChanged(object sender, EventArgs e)	To view the outpatient details
4	Page_Load(object sender, EventArgs e)	To add the inpatient outpatient details
5	Button1_Click(object sender, EventArgs e)	To Insert the patient information
6	Calendar1_SelectionChanged(object sender, EventArgs e)	To select the appointment date
7	Button1_Click(object sender, EventArgs e)	To view particular patient details
8	canbtn_Click(object sender, EventArgs e)	To view Hospital employee login page
9	pidddl_SelectedIndexChanged(object sender, EventArgs e)	To get the medical test
10	pttypeddl_SelectedIndexChanged(object sender, EventArgs e)	To view the inpatient details
11	PID_SelectedIndexChanged(object sender, EventArgs e)	To allocate the patient admission date
12	Button1_Click(object sender, EventArgs e)	To view overall patient details.
13	Button3_Click(object sender, EventArgs e)	To view discharge patient details
14	subbtn_Click(object sender, EventArgs e)	employee signup
15	Resbtn_Click(object sender, EventArgs e)	employee registration

Table 2.Unrepeated Functions- Library Management System

Fid	Fname	Task
1	ImageButtonSearch_Click1(object sender, ImageClickEventArgs e)	It is used for searching the books
2	UpdateCategoryToDDL()	Updating the search usage.
3	flush()	to refresh the category of books
4	Gen(string emailid,string Message)	To send the message to user
5	ddlMemberID_SelectedIndexChanged(object sender, EventArgs e)	To update the member id
6	ddlBookName_SelectedIndexChanged(object sender, EventArgs e)	To Update the book name
7	GridViewMemberInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)	To delete the unwanted record
8	ddlDropBookPublisher_SelectedIndexChanged(object sender, EventArgs e)	To view the books dynamically
9	btnMakeEntry_Click(object sender, EventArgs e)	For Adding new member
10	AddOrUpdateBookInfo(GlobalAttributes _ga)	To add or update book information
11	DataTable GetBookConsumerList()	To get the consumer list by using data table.
12	DeleteMemberInfo(GlobalAttributes _ga)	To delete the member information
13	GridViewMemberInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)	To delete the unwanted record
14	AddBookLendEntry(GlobalAttributes _ga)	To insert the books to database

Table 3.Repeated Function- Hospital Management System

Fid	Fname	No. of times repeated	Task
1	Page_Load(object sender, EventArgs e)	8	To load asp.net page
2	Page_Load(object sender, EventArgs e)	2	To load page and retrieve patient information from database
3	pidddl_SelectedIndexChanged(object sender, EventArgs e)	2	To get the appointment from the doctor
4	Button2_Click(object sender, EventArgs e)	2	To redirect to the next page
5	void Page_Load(object sender, EventArgs e)	3	Doctor's signup process
6	backbtn_Click(object sender, EventArgs e)	2	Redirect to the next page

Table 4.Repeated Function- Library Management System

Fid	Fname	No. of times repeated	Task
1	Page_Load()	4	To load the page
2	Page_Load()	4	To load page with calling method
3	ddlBookEdition_SelectedIndexChanged()	2	To select the word going to changed

Table 5 Final Report- Repeated Functions

Fid	Fname	Task
1	Page_Load(object sender, EventArgs e)	To load asp.net page
2	Page_Load(object sender, EventArgs e)	To load page with calling method

Table 5 represents the functions repeated in both the sample applications i.e. Hospital and Library Management System.

IV. Conclusion And Future Work

The tool developed is easy to use and also can be used for multiple programming languages. In this tool, the clone detection is implemented using frequent pattern growth algorithm which includes tree data structure and array format which makes the clone detection easy and time saving. At the end a graph will be generated to evaluate the total time taken by the frequent pattern growth algorithm to complete its detection work. This algorithm is potentially faster and easier to use and work with as compared to the other pattern mining algorithm which makes it more efficient and cost effective.

This tool will be used to find beneficial clones. These beneficial clones will be used in the area of software engineering to provide support in developing quality software products.

References

[1] Girija Gupta and Indu Singh, "A Novel Approach Towards Code Clone Detection and Redesigning", International Journal Of Advanced Research in Computer Science and Software Detection, September 2013, Volume 3.

[2] M Suman, T Anuradha, K Gowtham, A Ramakrishna, "Frequent Pattern Mining Algorithm Based on FP Tree Structure and Apriori Algorithm", International Journal of Engineering Research and Application, January 2012 Volume 2.

[3] Florian Verhein, "Frequent Pattern Growth (FP Growth) Algorithm", January 2008.

[4] Chanchal Kumar Roy and James R. Cordy "A Survey on Software Clone Detection Research", September 2007.

[5] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan, "Frequent Pattern Mining: current status and future directions", January 2007.

[6] Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant Anna, Lorraine Beir, "Clone Detection Using Abstract Syntax Tress".

[7] Lingxiao Jiang, Ghassan Mishergi, Zhendong Su, Stephane Glondu, "Scalable and Accurate Tree-based Detection of Code Clones".

[8] Bjorn Bringmann, Siegfried Nijssen and Albrecht Zimmermann, "Pattern-Based Classification: A Unifying Perspective".

[9] Matthias Rieger and Stephane Ducasse, "Visual Detection of Duplicated Code".

[10] Shihab Rahman, Dolon Chanpa, "FP Growth Algorithm For Mining Frequent Pattern".

[11] Depti pawar, Mrs.Sankriti Shiravale "http://www.slideshare.net/deepti92pawar/the-comparative-study-of-apriori-and-fpgrowth-algorithm#", March 2013.