DETECTION OF CROSS-PROJECT BENEFICIAL CLONES

Ms.Kavitha Esther Rajakumari
Computer Science and Engineering, Sathyabama University,
Chennai, Tamil Nadu, India
kavithaesther7@gmail.com

Dr.T.Jebarajan
Computer Science and Engineering, Rajalakshmi College of Engineering,
Chennai, Tamil Nadu, India
drtjebarajan@gmail.com

Abstract:

Duplicate codes are also known as code clones. They are considered as one of the main factors that deteriorate the quality of software. They are usually discarded by using automatic clone detection tools. In this paper the clones are detected using a data mining approach. The clones are well analyzed and the beneficial code clones are retained. These clones are maintained separately and are used in software maintenance. The beneficial clones will definitely help in reducing the overall time spent in maintenance phase.

Keywords: Code clones; beneficial clones; data mining.

1. Introduction

Duplicate codes or code clones are the repetitions of an already existing fragment or piece of code. Clones are of four types. They are named as Type-1, Type-2, Type-3 and Type-4 clones. Type-1 clones are exactly the same replica of another fragment. Type-2 clones are ones which have names changed in the fragments, example variable names. Type-3 clones is the modification of the original code fragments .i.e. either adding or deleting elements from the original fragment. Type-4 clones are functional clones.

They are mostly introduced by copy and paste work. Sometimes they are instituted accidently. If large numbers of clones are present in the software, maintenance phase becomes complicated. Time delay, finding the place of error, time spent in bug removal, etc are some of the difficulties faced during the maintenance period. So removal of harmful clones is very important [1].

Despite the fact that clones are harmful, there are clones which are beneficial. But they remain undetected and get discarded along with the harmful ones. One has to make careful analysis before eliminating the entire clones that are detected. But the problem with automatic clone detectors is that they ignore the beneficial clones.

In this paper a method is proposed to detect and to store the beneficial clones. The clones are detected based on data mining algorithm and stored separately in a table. These good clones are purported as a database and used efficiently.

2. Literature Survey

Most of the researchers have discussed about the negative aspects of clones. Below are the summary of techniques and views expressed about clones by various authors, in the field of code clones.

Pitts and Raoult and Guillemin describe the equivalence of programs in terms of operational semantics. Pits introduced a method proving contextual equivalence of ML functions. Result and Guillemin proves that two different ways of program equivalence are fixed-point semantics and operational semantics. They are discussed for recursive definitions. Ivanovo introduces a technique called program schemata. He has said that program transformation is to transform a program into a semantically equivalent one by applying only semantic-preserving transformations.

Fischer tells whether a component can be reused in a given context or not without any modification. The basic idea of the reuse approach is that a component satisfies a query with precondition and post condition. Podgurski and Pierce introduce a method called behavior sampling for the automated retrieval of components from a software component library for the purpose of reuse. The components are organized in a classification. The user is prompted with a choice between two different detections and its behavior. They are static and dynamic similarity detection.

There has been significant research dedicated to detecting syntactically similar code fragments. Program based clone detection can be found in the survey from Kosher Clone detectors, such as CCFinder, Clone Detective and Deckard. They are effective in finding clones created by copy and paste programming.

Rabin-Karp fingerprinting is used for calculating the length of n substrings of a text. First text-to-text transformation is performed for the dataset by eliminating uninterested characters. Then all whitespace characters are removed except line separators.

Table 1.Taxonomy for code clone Techniques.

Language Paradigm	Only Procedural
	Only Object-oriented
	Both procedural and OO
	Lisp-like languages
	Byte code
	Assembly code
	Extreme Programming
Clone Relation	Directly Clone Pair
	Directly Clone Class
	CC in post processing
Level of similarity	Textual
	Lexical
	Syntactical
	Semantical
	Hybrid
Clone granularity	Free
	Fixed
Clone similarity	Extract Match
	Parameterized match
	Near miss
	High level clone
	Design level strucutural clone
Comparison granularity	Line
	P-Line
	Substring
	Identifiers and comments
	Tokens
	Statements, Subtree, Sub graph
	Begin-End Blocks, Methods, Classes, Files

3. Algorithm Used

The FP-Growth algorithm allows frequent itemset discovery without candidate itemset generation. The algorithm scans the frequent patterns that appear in a dataset. This technique has two phases. The first phase is to derive the frequent items and the second phase is to construct an FP-Tree based on the results of the first phase. The technique works as follows: In the first scan of the database, frequent items (1-itemsets) and their support counts are derived. Then the itemsets are listed in descending order of support count. An FP-Tree is then constructed based on the support count of the itemsets. Mining of an FP-Tree is performed by calling FP-growth procedure (FP- tree, null) which is implemented as follows [4, 9, 10].

The following is the pseudo code of the FP-Growth algorithm.

Procedure FP_growth (Tree α)

```
(1)    If Tree contains a single path p then
(2)    For each combination (β) of the nodes in the path p.
(3)    Generate pattern b∪α with support = minimum support of nodes in β;
(4)    Else for each $a_i$ in the header of Tree {
(5)    Generate pattern β=$a_i$∪α with support = $a_i$ support;
(6)    Construct β, s conditional pattern base and then β's conditional FP_tree Tree β;
(7)    If $Tree_β$≠ф then
(8)    Call FP_growth ($Tree_β$, β) ;}
```

4. Proposed Method

The method uses the concept of a data mining algorithm known as the FP-Growth (Frequent Pattern Growth) algorithm. The algorithm originally scans the database and finds out the repeated patterns. Here the source code is scanned for repeated patterns. As a result the code clones will be detected. The harmful clones are discarded and beneficial clones are retained. Following are the steps carried out in this method:

(1) ***Extraction of functions:***

Initially a software project file is uploaded from the software projects database into the proposed system. Function retrieval is made possible through the frequent pattern growth algorithm. This algorithm uses the concept of frequent pattern mining. All functions used in the project are mined in this step.

(2) ***Removal of unwanted functions:***

After mining of function names, the functions are examined to know whether they are dead or active functions. Dead functions are those which are present in the program but remain inactive. These are used by some developers to make the source code of a program to appear complicated. These functions have no part in the execution they just increase the number of lines in a program. These are removed from the program once they are detected.

Active function plays a role in the program. The role played by these functions can be harmful or beneficial. Harmful functions are discarded and beneficial ones are retained.

(3) ***Creation of functional tables:***

The beneficial functions are divided into two categories, namely repeated and unrepeated functions. Repeated functions are known as functional clones. Functional clones belong to Type-4 clone category. These are put in a table format. Similarly unrepeated functions are put in a separate table. These tables are known as the functional tables.

(4) ***Normal functions to library functions*:**

The beneficial functional clones are assessed carefully by noticing the number of occurrences in the software project. A threshold value is assigned for the repeated functions. When this value is reached it is converted into a library function. By doing this procedure, normal functions are converted to library functions.

(5) ***Software Maintenance:***

The maintenance phase consists of four types of maintenances. They are adaptive, corrective, perfective and preventive maintenance. A sample software project is taken and all four maintenances are carried out. Here the functional tables play a major role while carrying out the maintenance task.

In adaptive maintenance, the software project is modified in order to get adapted with the environment. The modification is carried out to keep the software project usable in changed or changing circumstances. In corrective maintenance, the software project is diagnosed for errors. The detected errors are fixed by using the functional table. In perfective maintenance, it implements new or changed user requirements. In preventive maintenance, it performs increasing software maintainability or reliability to prevent future problems as far as concerned.

In all these types of maintenances the functional tables are used to enhance the projects. The functions present in the table are updated periodically and scanned for bugs. Time spent on maintenance is reduced considerably. The availability of reusable functional components makes the work straightforward.

The above framed steps are repeated for each software project in the database. Therefore the beneficial functional clones are used in a cost-effective manner. Below a flow diagram is presented that represents the flow of the proposed approach.

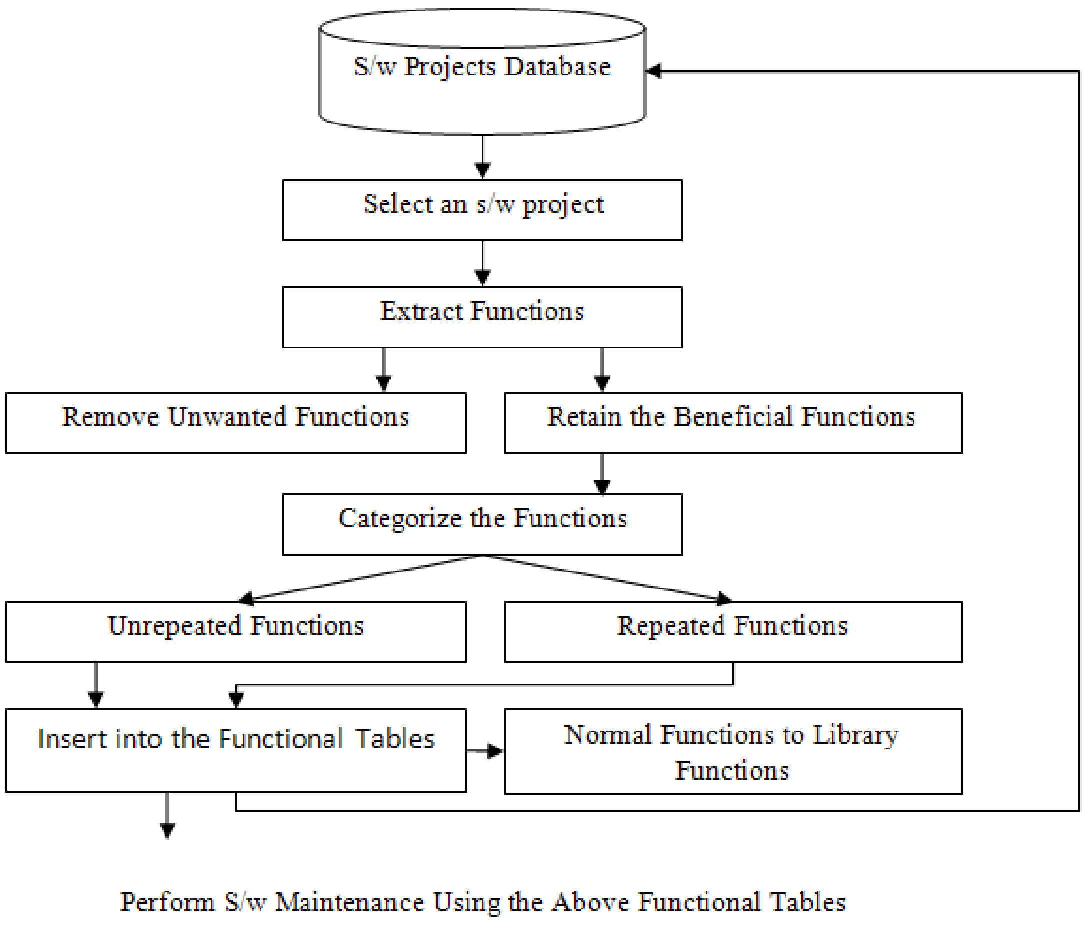

S/w Projects Database
Select an s/w project
Extract Functions
Remove Unwanted Functions
Retain the Beneficial Functions
Categorize the Functions
Unrepeated Functions
Repeated Functions
Insert into the Functional Tables
Normal Functions to Library Functions
Perform S/w Maintenance Using the Above Functional Tables

Fig.1: Flow diagram of the proposed approach.

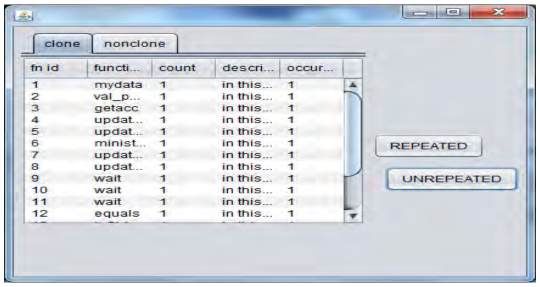
5. Results And Discussions

The database here is customized to a particular organization. So the proposed process is repeated till all projects in the database are scanned. By doing this, the organization can use the components of already developed successful projects. Below is the metrics used for the clone table.

Table 2. Functional Table Metrics

Code Clone Table		
S.No	**Code Analysis**	**Depth measures**
1.	Total no of projects	10
2.	Total no of lines of code	500
3.	Total no of methods retrieved	42
4.	Total no of similar methods retrieved	12
5.	Library file converted	2

Below is the functional table that was created using the beneficial functions. Initially, the onetime occurrences are displayed. Next two times, three times etc, occurrences are displayed. By using this table, software maintenance can be carried out efficiently. Time is reduced considerably.



fn id	functi...	count	descri...	occur...
1	mydata	1	in this...	1
2	val_p...	1	in this...	1
3	getacc	1	in this...	1
4	updat...	1	in this...	1
5	updat...	1	in this...	1
6	minist...	1	in this...	1
7	updat...	1	in this...	1
8	updat...	1	in this...	1
9	wait	1	in this...	1
10	wait	1	in this...	1
11	wait	1	in this...	1
12	equals	1	in this...	1

Fig.2: Functional table

6. Conclusion And Future Work

In this paper, the functional clones were detected by using the data-mining algorithm. Next, functional tables were created which consists of beneficial functional clones. Certain metrics were considered to detect clones. This proposed method projects the positive aspects of clones. The code clones usage is depicted in this paper. Time spent in carrying out the maintenance task is reduced noticeably.

Future work includes the application of these functional tables in the maintenance phase. Moreover certain steps described in the proposed method are done manually. In future, the entire process will be made automatic.

References

[1] Akito Monden, Daikai Nakae, Ken-ichi Toshihiro Kamiya, Matsumoto, Shin-ichiSato “Software Quality Analysis by Code Clones in Industrial Legacy Software”.

[2] Antonio Cuomo, Antonella Santone, Umberto Villano “A Novel Approach Based on Formal Methods for Clone Detection”.

[3] Hiroshi Igaki Keisuke Hotta, Tomoya Ishihara, Shinji Kusumoto, Yoshiki Higo: “*Inter-Project Function a Clone Detection toward Building Libraries* Survey”—An Empirical Study on 13,000 Projects in year, pp-17.

[4] JiaweiHan, JianPei, Micheline Kamber, ”Data mining concepts and techniques” published by Elsevier, a division of Reed Elsevier India Private Limited Chapter 6. Mining frequent patterns Associations and correlations, Basic concepts and methods, 6.2 Frequent Item set Mining Methods, 6.2.4 A Pattern Growth Approach for Mining Frequent Itemsets pp 243-245,257-259.

[5] Jonathan I. Maletic ,Michael L. Collard, Nouh Alhindawi Omar Meqdadi “Towards Understanding Large-Scale Adaptive Changes From Version Histories”in the year 2009,pp-1-4.

[6] Lars Heinemann “Effective and Efficient Reuse with Software Libraries”.

[7] Nelly Maneva “Software Quality Assurance and Maintenance for Outsourced Software Development”.

[8] Nicolas Battenberg, Weiyi Shang, Walid M. Ibrahim, Bram Adams, Ying Zou, Ahmed E. Hassan “An empirical study on inconsistent changes to Code clones at the Release level”.

[9] http://Data_Mining_Algorithms_In_R/Frequent_Pattern _Mining/The_FP-Growth_Algorithm.

[10] http://www.cs.uiuc.edu/~hanj/pdf/dami04_fptree. Pdf pp- 73-79.