A Method for Tracking and Managing Code Clones

E. Kodhai, Dr. S. Kanmani

1Department Of Information Technology, Sri Manakula Vinayagar Engineering College, Puducherry, India. Email: kodhaije@yahoo.co.in
2Department Of Information Technology, Pondicherry Engineering College, Puducherry, India. Email: kanmani@pec.edu

Abstract—Clones are usually a bad smell in software maintenance. A number of clone detection techniques are available which detects efficiently. However, they all use refactoring technique for implementation. But number of analysis in code clones proved that it is not always possible. Our method is a technique which let the clones for modification in a separate alternate window and provides support for updating. Our method also provides support for software evolution by documenting the clones, and provides the note of changes made by the developers.

Index Terms—Software Maintenance, Code Clone, Refactoring.

I. INTRODUCTION

The size and the complexity of software increase, maintenance tasks become more difficult and burdensome. Arthur [3] states that only one-fourth or one-third of all life-cycle costs are attributed to software development. Also, Yip et al. [12] says that some 67% of life-cycle costs are expended in the operation-maintenance phase of the life cycle.

A code clone is defined as a code fragment occurring more than once in identical or similar form into a software system. The presence of code clones is the factor making software maintenance more difficult [11]. For example, if a code fragment is modified, it has to be determined whether or not to modify each of its code clones. Unfortunately, we often overlook some of the code clones that should be modified simultaneously.

One technique that helps reducing the number of code clones is Refactoring. Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior [8]. By making refactoring efforts on a set of code clones, they can be merged into a method [6], a component [9].

But refactoring efforts may not always be a good solution to the clone problem. An empirical study of Kim et al. [10] revealed two points: first one is that some code clones are short-lived, and merging them wouldn’t improve the maintainability; second one is that most of long-living code clones are not suited to be refactored because there is no abstraction function of the programming language. Also, Balazinska et al. [5] reported that differences between code clones tend to hinder merging them, which indicates that it requires counter measures to modifying all code clones sharing the identical or similar form without overlooking.

In this paper we describe a technique which let the clones to be modified in a separate window which will not disturb the original source code. If the modified code fits then it can be altered in source code by simply saving the modified code. It also notes that which user is modifying the code and when he is doing by maintaining a separate database. It also stores the detected clones as a document for further software evolution. To implement this technique we use a tool called simscan as a eclipse plug-in. This tool helps us to detect the clones.

This paper is organized as follows. Section II describes the clone detection tool simscan. Section III and IV describe our proposed method and its implementation respectively. Section V describes related work. Section VI concludes our study.

II. CLONE DETECTION

The code clones in java source code are detected using the tool SIMSCAN.

A. SIMSCAN

SimScan (Similarity Scanner) is a utility to find duplicated or similar pieces of code in large amounts of Java sources. The utility is distributed as a plugin for Borland JBuilder and Eclipse Java IDEs and as a stand-alone command line tool.

To find similarities in code, SimScan relies on structural comparison of code which gives it a distinct advantage over simpler alternatives. For example, renaming a variable or changing the code formatting will not affect the results. Furthermore, it allows finding duplicate code fragments even when there are inserted or deleted lines of code or other differences, as long as these differences are relatively small to the size of the match. Not all matches are always suitable for refactoring, you have the option to tune SimScan search for stricter matches.

B. Search Results

The results will be grouped in groups of similarity. Each group represents a connected component of similar items. Not all items in the group are necessarily similar to each other. The groups of similarity will be sorted according to the defined sort criterion.

Each match in the group can itself be expanded to show all direct matches to this specific match (Because a group is just a connected component, not necessarily all other matches in he group will be direct matches to the specific match). The expanded view enables you to look at the specific similarities between two matches.
III. APPROACH

To implement our method we thus use the simscan results for the detection of clones. After detecting the clones, the following modules are:

**Clone Documentation**: clones are documented for future use in software evolution.

**Clone Notification**: The user is let to modify the clones in a separate alternate window without disturbing the original source code. If the alteration works properly then it can be saved in the original source code itself. Then a database is created to notify that which user have altered and by what time, etc.

Thus our method lets the system to be passed to simscan and then document the clones by clone documentation and let to alter the clones by change notification. Refer the flow in the System Architecture in figure 1.

![System Architecture](image)

IV. IMPLEMENTATION

To implement our method the first step is to plug-in the SIMSCAN tool in the eclipse plug-in. Then implement the two modules as follows

A. **Clone Documentation**

The Clone Documentation will document the clone groups. The documented clone groups are nothing but the group of clones that occur after the successful clone detection process. These documented clones will be used for future purpose.

Clone documentation saves the output of the detected clones. After the search finishes a folder will be created with a detailed report on the search results and the parameters of the search. The output consists of the following files and folders:

- **Search_parameters.txt**: Contains information on the general parameters of the search, such as search duration, scanned files and folders, scan options.
- **Duplicates.txt**: The source excerpts of all found duplicates are listed in this file, grouped by similarity group. It could be very useful for a quick overview of the results, selecting which similarity groups should be reviewed more closely, etc.
- **Duplicates_summary.txt**: A summary version of duplicates.txt with information for the duplicates without listing any source code.
- **Numbered directories (0001, 0002, etc.):** Each of the numbered directories contain a single group of similar items. These directories provide the same information as duplicates.txt, but in a more convenient form to browse and review.
- **Similarity_graph.txt**: Since similarity groups are just connected components, and not necessarily all nodes in a similarity group are similar to each other, this file is necessary if you want to know exactly which are the duplicates to a specific match. One would rarely need to look at this file, however it provides for thoroughness.
- **Warning.txt**: This file is created only when the search has been cancelled and indicates that the results might be incomplete.
- **Log_simscan.txt**: (only when running from the command-line) Contains a log of the search progress.

B. **Clone Notification**

The clone groups are modified and saved i.e. the source codes are altered using the Source code alteration application designed using VB.NET.(refer figure 2.) The altered files are saved to the fixed location i.e. to the folder altered codes.

![Source alteration window](image)

After the source codes are altered it is saved to altered files folder. If the alteration does not change the functionality of the source code the original code could be replaced or a new copy of code could be saved without disturbing the original code.(refer figure 3.)

A database is maintained in-order to see which user has altered which code and time of alteration shows the time of the source code is been altered. The database report also shows which user altered which source code.(refer figure 4.)
V. RELATED WORK

Balazinska et al. [5] reported what kinds of differences between code fragments tend to hinder applying reengineering actions to them. The result was quite natural, that strictly identical or superficially different code clones are easier to reengineer than code clones including other kinds of differences. In other words, code clones including some gaps are difficult to be refactored. However, CCFinder can detect only identical or renamed code clones, that is, it cannot detect gapped code clones. It may be more effective to use another clone detection technique that can detect gapped code clones in the context of simultaneous modification.

Some of the existing techniques/tools are metrics-based detection [4] and CP-Miner [2]. Toomim et al. [7] has proposed a synchronous modification method on code clones included in the same clone set. In their method, there is a database of code clone information in the backend of the editor program. And when a code fragment included in a clone set is modified, other code fragments in the clone set are also simultaneously modified. Duala-Ekoko at al. [1] also has proposed a simultaneous editing method. The method identifies corresponding lines in code fragments being similar to each other based on the Levenshtein distance[6] after the code fragments are detected.

VI. CONCLUSION

In our proposed system, after clone detection the system does two functions. In the clone documentation phase, after successful clone detection, the detected clones are documented. The purpose of documentation is to refer the detected clones for future purpose. In the clone notification phase, alterations are made to source code and system is made clone free. Thus in our method we have managed the clones after the detection of clones by using the tool for software maintenance.

REFERENCES