Gemini: Code Clone Analysis Tool

Yasushi Ueda†, Yoshiki Higo†, Toshihiro Kamiya‡, Shinji Kusumoto† and Katsuro Inoue†

†Graduate School of Information Science and Technology, Osaka University,
Toyonaka, Osaka 560-8531, Japan, Phone:+81-6-6850-6571, Fax:+81-6-6850-6574
{y-ueda, kusumoto, inoue}@ist.osaka-u.ac.jp
‡PRESTO, Japan Science and Technology Corp.
kamiya@ist.osaka-u.ac.jp

Abstract

In this paper, we present a maintenance support environment based on code clone analysis, called Gemini.

1 Introduction

Maintaining software systems is getting more complex and difficult task, as the scale becomes larger. Code clone is one of the factors that make software maintenance difficult [1]. A code clone is a code portion in source files that is identical or similar to another. If some faults are found in a code clone, it is necessary to correct the faults in its all code clones. However, for large-scale software, it is very difficult to correct them completely. In order to detect the code clones effectively, various clone detection methods have been proposed.We have proposed and developed a code clone detection tool, CCFinder [2], that detects code clones from single program or multiples.

In this paper, we present a maintenance support environment, called Gemini [3], which visualizes the code clone information from CCFinder. Using Gemini, we can specify a set of distinctive code clones through GUIs, and refer the portions of source code corresponding to the clones, so that reconstruction or so can be carried out with high maintainability.

2 Preliminaries

2.1 Definition on clone and related terms

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. A code portion in a clone class of a program is called a code clone.

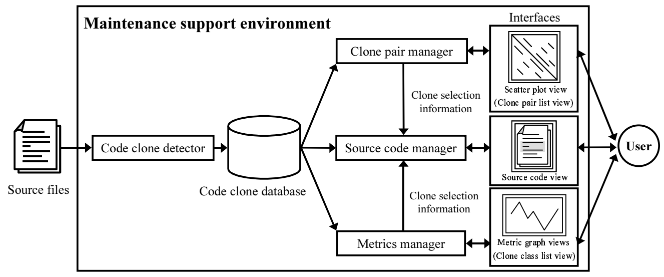


Figure 1. Architecture

2.2 CCFinder

CCFinder detects code clones from all the sub-strings of token sequence of source code and outputs the locations of the clone pairs on the source code. In clone detection of CCFinder, the token sequence of source code is transformed, i.e., tokens are added, removed, or changed based on the transformation rules that aims at regularization of identifiers and identification of structures. Then, each identifier related to types, variables, and constants is replaced with a special token. This replacement makes code portions with different variable names a clone pair. Details of CCFinder have been shown in [2].

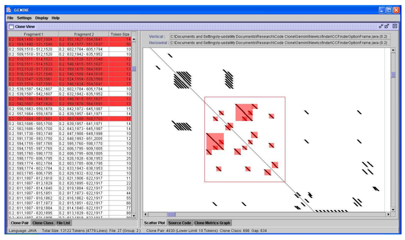
CCFinder has no GUI but it only generates character-based output. It is quite difficult for the person who analyzes the source code to investigate a code clone only from this information and source code, and to perform analysis of the source code and reconstruction of it.

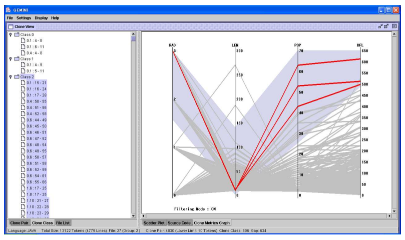
3 Maintenance support environment

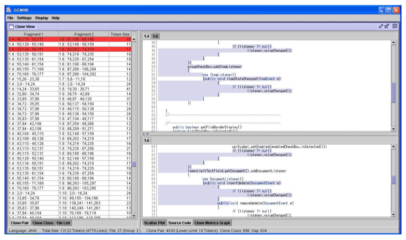
3.1 Design

Gemini invokes CCFinder internally and analyzes the outputs from CCFfinder. The architecture of Gemini is shown in Figure 1.

First of all, source files are input into code clone detector, CCFinder. Then the output of CCFinder is accumulated in this code clone database. Using the database, clone pair manager and clone class manager visualize the information





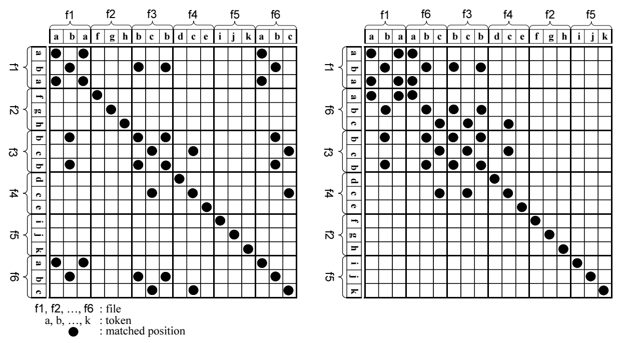


(a) Scatter plot view (right side) and clone pair list view (left side)

(b) Metric graph (right side) and clone class list view (left side)

(c) Source code view (right side)

Figure 3. Snapshots of Gemini



(a) Before sorting

(b) After sorting

Figure 2. A simple examples of scatter plot

of clone pairs or classes. On some interfaces, user can specify clone pairs or classes, he or she is interested in. By selecting them, he or she can refer to the actual source code through source code manager and its UI.

As principal interfaces to analyse code clones, there is scatter plot view and metric graph view in Gemini as follows:

Scatter plot view

A simple example of scatter plot is shown in Figure 2(a). In scatter plot, both the vertical and horizontal axes represent lines of source files. A black dot means that the corresponding tokens on the horizontal and the vertical axis are the same. So a clone pair is shown as a diagonal line segment. Naturally, a diagonal line from the upper left to the lower right is always drawn since such dot means comparison of token with itself. The dots are symmetrical with a diagonal line. In Figure 2(a), each file includes only three tokens in order to simplify the plot and files are sorted in alphabetical order of the file paths.

However, the distribution of dots is occasionally spread widely, depending on the file order. In such case, the cost for analysis is high especially in large-scale software. So we give this view a sorting function with respect to the file order not to distribute clone pairs all over the scatter plot as much as possible. As a basic idea of sorting, we put similar files as near as possible. Here the ratio of covered code range of a file by clones of the other file that is target for comparison is used as the criterion of similarity. Figure 2(a) is a scatter plot before sorting. By the sorting, the distribution becomes narrower in Figure 2(b).

Using this plot as user interface like in Figure 3(c)), user can easily identify the location of clone pairs. Then the corresponding source code can be referred through source code view (See Figure 3(c)).

Metric graph view

In this view, the values of several kinds of metrics for each clone class are shown as a graph, parallel coordinates. For an example of metrics, there is DFL [2]. It indicates an estimation of how many tokens would be removed from source files when the code portions in a given clone class are reconstructed. Based on the values of such metrics, we can focus on distinctive code clones that may be meanigful in maintenance. By setting the warning (interesting) range about the value of each metric, user can select clone classes whose metric values are in the range (See Figure 3(b)). Also the corresponding source code can be referred through source code view.

3.2 Implementation

Gemini has been implemented in Java and runs on the environment where JDK 1.3 VM can be executed. A example of GUI is shown in Figure 3.

References

[1] M. Fowler, *Refactoring: improving the design of existing code*, Addison-Wesley, 1999.

[2] 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, 2002.

[3] Y. Ueda, T. Kamiya, S. Kusumoto, K. Inoue, “Gemini: Maintenance Support Environment Based on Code Clone Analysis”, *Proc. of the 8th International Symposium on Software Metrics*, pp.67-76, 2002.