SHINOBI: A Tool for Automatic Code Clone Detection in the IDE

Shinji Kawaguchi*, Takanobu Yamashina†, Hidetake Uwano‡, Kyhohei Fushida*,
Yasutaka Kamei*, Masataka Nagura*§ and Hajimu Iida*
*Graduate School of Information Science,
Nara Institute of Science and Technology, Ikoma-shi, Nara, Japan
Email: {kawaguti, kyohei-f, yasuta-k, nag}@is.naist.jp, iida@itc.naist.jp
†Nihon Unisys, Tokyo, Japan
Email: takanobu.yamashina@unisys.co.jp
‡Department of Information Engineering,
Nara National College of Technology, Nara, Japan
Email: hideta-u@is.naist.jp
§Presently with Systems Development Laboratory, Hitachi, Ltd.

***Abstract*—Recent research has acknowledged that code clones decrease the maintainability and reliability of software programs, thus it is being regarded as one of the major factors to increase development/maintenance cost. We introduce SHINOBI, a novel code clone detection/modification tool that is designed to aid in recognizing and highlighting code clones during software maintenance tasks. SHINOBI is implemented as an add-in of Microsoft Visual Studio that automatically reports clones of modified snippets in real time.**

I. Introduction

Code clones are one of the major obstacles to software maintenance. If a defect is contained in a code portion that has been copied many times, we have to investigate not only that code portion, but also all copied codes. This is very time consuming especially for large scale software.

Presently, many code clone detection tools have been proposed, with some of these tools highly integrated with the development environment [1], [2], [3]. This is very important as programmers change code clones using some sort of development environment, not using clone detection tools.

These clone detection environments require the running code clone detection to be manually switched on. However, such manual detection cannot be run so frequently because code clone detection from the whole source code takes more than five or ten minutes for large scale software. CloneTracker [1] reduces detection tool invocation by using a Clone Regional Description (CRD). CRD is robust for source code changes, and it enables tracking of code clones if source codes are modified from when the code clone detection was applied. However, CloneTracker needs to re-run the code clone detection tool to detect newly created code clones.

We believe that it is important to detect newly created code clones so as to react to code clones in their early stages. This is because it is practically recommended to refactor code clones that appear too frequently as it is most likely not part of the original design. In such a situation, the code clone detection results should be shown in real time, reducing the spread of this code clone and reducing refactoring.

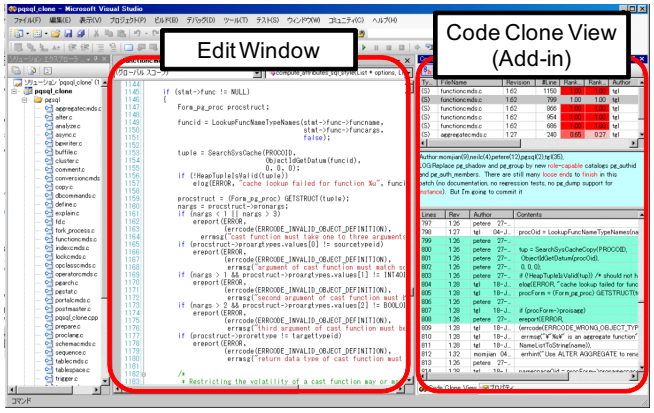

Figure 1. SHINOBI Client Screenshot

This paper introduces SHINOBI, a tool for automatic code clone detection. The main features of SHINOBI are as follows.

- SHINOBI is highly integrated with Microsoft Visual Studio. For instance, it is implemented as an add-in of Visual Studio. A programmer can easily check and edit detected code clones.
- SHINOBI automatically detects code clones with source code being edited. The detection process is automatic, implicit, and quick. A programmer can get a list of code clones without noticeable time penalty whenever he develops with the IDE.
- SHINOBI highlights code clones to help recognize code clones during software maintenance tasks.

II. SHINOBI

Figure 1 shows a screenshot of SHINOBI. SHINOBI automatically detects code clones being edited in Visual

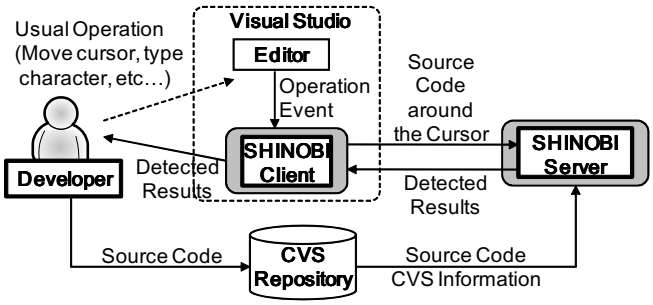

Figure 2. An Architecture of SHINOBI

Studio from the whole source code. The list of detected code clones are shown in the right pane. In the right pane, we can confirm the number of code clones, location of each code clone, and source code of each code clone.

SHINOBI has a token-based clone detection engine. This engine is nearly unaffected by the change of a variable identifier like CCFinder [4]. In practice, we implemented clone detection engine referring to their paper.

SHINOBI is implemented as SHINOBI server and SHINOBI client (Figure 2.) The SHINOBI server automatically acquires and parses source code from a specified directory or CVS repository. If you specify CVS repository, you can register all file revisions, or newest files revisions only. Next the SHINOBI server prepares the Suffix Array Index to detect code clones quickly. Once the SHINOBI server created the Suffix Array Index, it can search similar token sequences in $O(m \log(n))$ where m is token length of the source code sent from the SHINOBI client, and n is the number of all stored tokens. The SHINOBI server updates the Suffix Array Index whenever new code is committed to CVS repository. It requires very short time because the SHINOBI server analyzes newly added or deleted source code only.

The SHINOBI client always observes the cursor movement in Visual Studio. The client sends source code to the SHINOBI server if movement is detected. You can tweak how log tokens are sent to the SHINOBI server. The default is 50 tokens. The SHINOBI server then attempts to detect code clones using Suffix Array Index. Finally detected code clones are sent to the SHINOBI client which display the results to developers in Code Clone View (the right pane in Figure 1).

So as not to frustrate users, The SHINOBI server must detect code clones very quickly. In our experimentation, the SHINOBI server detects code clones less than 0.5 second for 4.5 MLOC software using 3.6GHz Pentium IV Windows XP machine. We think it shows that SHINOBI can be used for large scale software.

III. Usage Scenario

SHINOBI tells you how many code clones exist in the whole source code as soon as a copy of a code portion is made. This feature helps us notice code clones and which code needs refactoring at an early stage. Such automatic detection is especially essential if you are unaware of other developers creating clones. Using SHINOBI, you can see how much code has been copied by other developers as well as yourself.

SHINOBI is also used for software maintenance. When fixing source code, SHINOBI informs of the code clones in the source code which could be candidates of code to be fixed. SHINOBI automatically detects code clones without explicit code clone detection invocation. Without SHINOBI, more attention is needed to identify code clones when source code is fixed.

IV. Conclusion

In this paper, we introduced SHINOBI, a clone-aware software development environment, highly integrated with Microsoft Visual Studio. SHINOBI automatically and implicitly detects code clones and shows them to help recognize them. Developers easily understand how many code clones exist in the whole source code. As stated in Section III, SHINOBI will be useful when developers have to edit code clone in software maintenance tasks. SHINOBI is available at http://sdlab.naist.jp/prj_shinobi.html

Acknowledgement

We would like to thank employees at Nihon Unisys Ltd. for their cooperation and valuable advice. This work was supported by StagE Project, the Development of Next Generation IT Infrastructure, supported by MEXT.

References

[1] E. Duala-Ekoko and M. P. Robillard, "Tracking code clones in evolving software," in *Proceedings of the 29th international conference on Software Engineering (ICSE2007)*, 2007, pp. 158–167.

[2] P. Jablonski and D. Hou, "CReN: a tool for tracking copy-and-paste code clones and renaming identifiers consistently in the ide," in *Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, 2007, pp. 16–20.

[3] R. Tairas, J. Gray, and I. D. Baxter, "Visualizing clone detection results," in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE2007)*, 2007, pp. 549–550.

[4] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: A Multilinguistic Token-based Code Clone Detection System for Large Scale Source Code," *IEEE Trans. Software Engineering*, vol. 28, no. 7, pp. 654–670, 2002.