Scientific Journal of Impact Factor (SJIF): 3.134

E-ISSN (O): 2348-4470
P-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 12,December -2015

Supportive Advancement Tool for Refactoring Through Code Clone Analysis

Swati Giri[1], Shikhar Kapoor[2]

[1,2]*Information Technology, SRM University (India)*

ABSTRACT **–** Software clones are the perennial piece of technical development and this development is an enactment of Code Cloning. The cloned code is rendered into the required code by Refactoring. This provides a convenient way to achieve design objectives in a real time environment with minimum efforts. In this paper, a supportive advancement tool is proposed for refactoring of the cloned code.

Keywords: *Code Cloning; Refactoring; Real Time Environment; Advancement Tool; Software Clones*

I. INTRODUCTION

Software Clones refer to the code snippets that pose a similarity to each other. The approach to software development is a real time practice and the suitable choice for developers is the timesaver method of code cloning. Code Cloning is the copy paste method of reusing an existing code to take advantage of the assets that have already been implemented. In presence of code clones, the normal functioning of the system may not be affected, but without countermeasures by the maintenance team, further development of the software may become prohibitively expensive.

The cloned code is rendered into the required code by Refactoring. Code Refactoring refers to restructuring of existing computer code without changing its external behaviour. It helps in improving the non-functional attributes of the software. This method provides a convenient way to achieve design objectives in a real time environment with minimum efforts. The proposed system is a supportive advancement tool for refactoring of the cloned code.

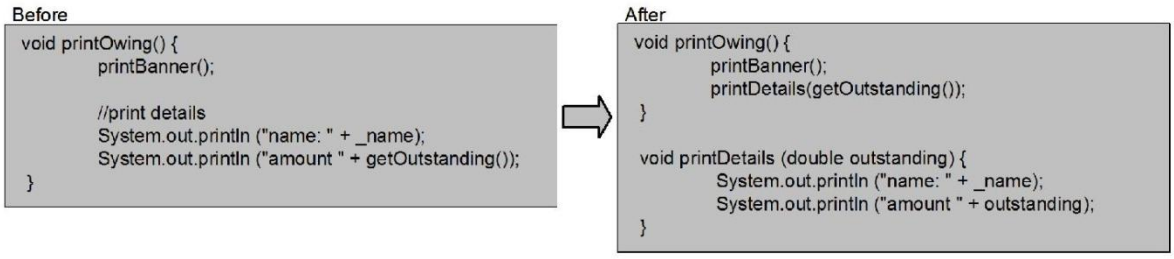

Fig 1: Code Cloning and Refactoring

II. PROPOSED SYSTEM

The idea of the proposed system is to provide a supportive advancement tool for Refactoring of cloned code. The Support tool conducts and manages several passive operations in refactoring.

1.1 Clone Detection.

The tool advances the refactoring process by identifying the clone pairs in the software code for a better assessment.

1.2 Clone Assessment.

The Support tool assesses the percentage of cloned code in the software code thus providing a faster and efficient way of clone assessment.

1.3 Systematic Updation.

It fastens the refactoring procedure as it detects the same clone repetitions in the software code and refactors all other replicas automatically, when one is systematically updated.

1.4 Reverse Engineering.

It not only fastens the refactoring process but also provides an easy rollback, if the refactored code is not adhered to the required output or changes are to be made, thus saving the developer from strenuous work.

For instance, in a large system a department analyses only function clones and reports that 6% - 7% code is cloned, another department estimated the normal industrial code and reports 12%-18% of duplicate code. This estimation of clones is inconsistent and would result in update anomalies. The Support tool eliminates these anomalies by detecting and assessing the potential clones and with systematic updation it fastens the refactoring in the recurring similar clones.

This Software Tool influences the Refactoring process by providing services and techniques that allow for more abstraction, are used for breaking code into more logical pieces and for improving names and location of code.

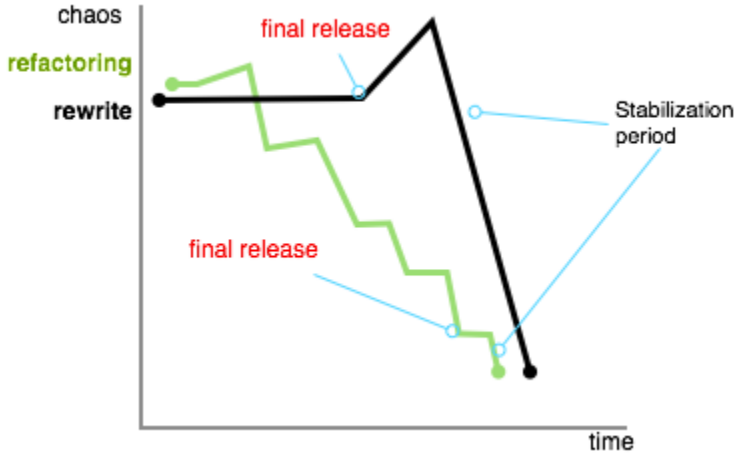

Fig 2: Refactoring versus Rewriting without the Support Tool

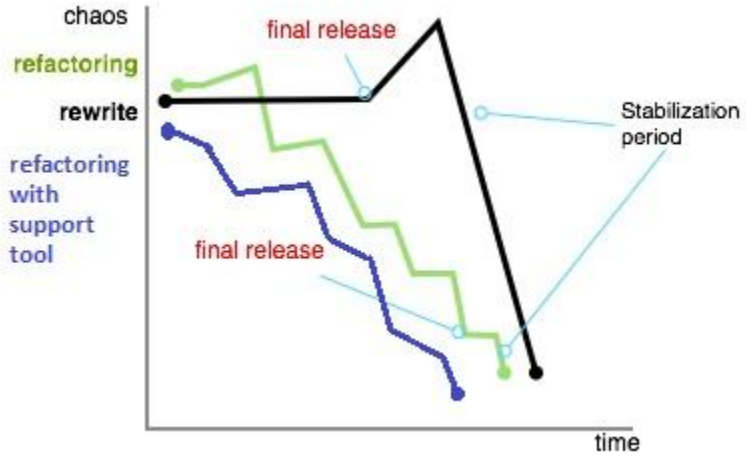

Fig 3: Refactoring versus Rewriting with the Support Tool

III. EXPERIMENTATION METHOD

The foremost step is the development of the support tool, its implementation and its integration with the existing refactoring systems as the proposed system is employed for use only after the tool's proper installation.

The following step is the initiation of the refactoring process with the assistance of the support tool. Once, the software code is assembled the support tool scans through the code to locate and highlight the clone pairs. Done with this, it presents the clones to the developers for refactoring. This is a rapid and effective process that saves time.

In the next step, the tool gathers the refactored code and then runs a match for the same clones recurring in the software code, and updates the matched clone with the refactored code throughout.

On completion of the above stated step, the tool compiles the refactored code with the software code and draws a comparison with the source code, presenting all the made changes. It provides a rollback method for reverse engineering when some changes are to be reverted.

Based on the developer's final refactored code, the Support tool compiles the code to present the software.

IV. ADVANTAGES AND APPLICATIONS

The proposed system provides the direct benefit of improving the quality of the source code by refactoring the cloned code and additional benefits such as:

- Detecting Library Candidates – It will provide the re-use potential of a code fragment that can be incorporated in a library.
- Helps in program understanding – The Support tool highlights the reused code and segregates it from the generic code, thus helping in clear understanding of the program.
- Find Usage Patterns – It states the reusability pattern of the clones.

V. DIRECT VS OVERHEAD COST

The initial cost of the setup would be marginally high as the Support tool is to be developed and integrated with the existing systems refactoring techniques so as to take benefit of the system.
Once implemented, the maintenance cost would be minimal. As the system would prove to be of great benefit to the developers, it will have high number of implementations, thus reducing the cost involved in the initial setup making a system with high financial feasibility.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present that putting the proposed system into practice can notably improve the development process of softwares providing scalability in the software life-cycle.
For future work, we plan to improve the existing systems and making them compliant to the needs of the proposed system. We plan to look into user-friendly techniques to speed up the setup and working of the system.

Finally, we intend to apply our system in the world of software development.

REFERENCES

[1] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone Detection Using Abstract Syntax Trees," Proc. of ICSM98, pp. 368-377, 1998.

[2] S. Ducasse , M. Rieger, and S. Demeyer, "A Language Independent Approach for Detecting Duplicated Code," Proc. of ICSM99, pp. 109-118, 1999.

[3] K. Prete, N. Rachatasumrit, N. Sudan, and M. Kim, "Template-based reconstruction of complex refactorings," 2010 IEEE International Conference on Software Maintenance, pages 1–10, Sept 2010

[4] Y. Dang, S. Ge, R. Huang, and D. Zhang, "Code clone detection experience at Microsoft," IWSC, pages 63–64, 2011.

[5] K. Kontogiannis, R. DeMori, E. Merlo, M. Galler, and M.Bernstein, "Pattern Matching Techniques for Clone Detection," Journal of Automated Software Engineering, Kluwer Academic Publishers, Vol. 3. pp.77-108, 1996.

[6] Jens Krinke, "Identifying similar code with program dependence graphs", Proc. of the 8th Working Conference on Reverse Engineering, pp. 562-584, 2001.

