Advances in Parallel Computing Technologies and Applications
D.J. Hemanth et al. (Eds.)

doi:10.3233/APC210154

Model for Refactoring a Software Using Feature Oriented Dependency (FOD)

Malathi S [a,1]

[a] *Lecturer in Computer Engineering (Deputed from Annamalai University), Srinivasa Subbaraya Govt Polytechnic College-Puttur*

Abstract. Refactoring is the process of improving the code of the software without affecting the external behavior of the code only by reconstruct the internal structure . It makes code cleaner, clearer, simpler or in other words, clean up the code. It also improves the quality of code then it became more reliable and easy to maintain through lifecycle of software. Refactoring has become renowned concept in software development process. The IDE (Integrated Development Environment) highly prefer this technique. Researches on refactoring technique have improved now a day. Beyond that , this particular technique is used to improve different functions of application software. It mainly speed up the function and helps to get the output much faster. In this proposed work Feature Oriented Dependency (FOD) tool is created used for refactoring process established on a chemical reaction optimization meta heuristic approach to discover the appropriate refactoring resolutions.

Keywords. Refactoring, Restructuring, Re-engineering, Reverse engineering.

1. Introduction

FOD methodology utilized using refactoring as re-engineering technique, the main concept of the FOD is to mold a software system according to the structures it provides i.e divide and run them independently, the objective is to build well-structured software that can be personalized to the requirements of the user and the application.

From a set of features, various different software systems can be produced, to share common features and also to add different features, in any software development, the domain analysis method can be used, a problem is analyzed and the solution for the problem is also defined.

Outsized and composite software systems require a strong understanding of desired system features, capabilities of the software are mandatory to implement those structures, software refactoring, long promised enhancements will become feasible only when the features and aptitudes are common to systems, the systematic exploration of software systems to be defined cohesion is called domain analysis.

[1] Malathi S, Department of Computer Engineering, Srinivasa Subbaraya Govt Polytechnic College, Puttur.
E-Mail: malathisivasamy1980@gmail.com.

2. Literature review

Kyo Kang [1] presented a domain analysis in which one technique that can be applied to meet this condition. By observing a class of related software systems and the common underlying theory of those systems, domain analysis can offer a reference model for describing the class. It can provide a basis for understanding and communication about the problem space addressed by software in the domain. Domain can also propose a set of architectural approaches for the execution of new systems.

John MCGregor [2] evaluated domain analysis, domain engineering, domain specific language and many other wildcard matches applies to the term domain. Domain may be thought of in several different ways. Many of the technical explanations view a domain as the subject for a family of programs. For example, telephone call switching systems. Domain based approach provides a context for software development that creates synergy with other business activities of the company generating strategically significant results.

Sukhdeep Kaur and Raman Maini [3] proposed that refactoring is a crucial process to improve the quality of software. Refactoring is a part of software engineering that expands more readability of program and maintainability of the software. Refactoring is a mostly used technique that gives the code simpler, cleaner, reusable, extendable, maintainable or other features by transforming a program. In programming language bad smell or code smell is a code or design problem that makes the software that specifies a problem, may be desired in it refactoring of code. In this paper some refactoring techniques discussed that are used to eradicate code smells from code or program.

Javier Perez et al. [4] presented a case study to evaluate the fitness of graph conversion tools for program refactoring. Adding quality for this purpose, a graph transformation system must be able to import a graph based on the models of java programs. Case study of this aims to enable comparison of various features of graph conversion tools, such as their expressiveness and their ability to interact with the user. The model of java programs is presented and some examples for translating JAVA source code into the model are provided.

Sangeetha [5] proposed refactoring id typically done in small stages. After every small steps we left with a working system that is functionally unchanged. So refactoring does not preclude changing functionality, it just says that it's a different activity from reorganizing code. It implies equivalence, the beginning and elegant or in other words. Refactoring is a well defined process that improves the quality of systems and allows developers to renovate the code that is becoming hard to maintain, without throwing away the existing source code and starting again. By careful application of refactoring the system behavior will remain the same, but return to a well structured design. FODA tools makes it more likely that the developer will perform the necessary refactoring, since the tools are much quicker and reduce the chance of introducing bugs.

U.Devi et al. [6] said refactoring is effectively known to remove the problem of code clones. Replicated code fragment in source code, usually known as code clones. Refactoring is a general and capable technique to remove the problem of code clones. Refactoring is the sequence of code changes which improve the quality of design (internal structure) without changing the behavior of software (external structure). Refactoring is usually a small change to the software. Code clones are categorized and detected based on certain established approach the tools which have been given are able to categorize

code clones based on the approaches. Refactoring is one remedial measure to tackle with the problem of code clones.

William [7] presented software refactoring is the systematic practice of improving application code's structure without altering it behavior. Refactoring is somewhat that he as an knowledgeable developer naturally did to his code, without deliberately thinking about it. It's a core element of agile approaches, and most professional IDE's include refactoring tool. Although the invention of refactoring and it is implementation into professional practice were practically destined. Refactoring tools and software processes such as agile development, a project team can now more freely choose to invest in software design. Designers are no more omniscient that before, but refactoring investing early only in design that will surely pay off down. API will be used many subsystems while delaying other design investment until the issues become clears.

Ramalakshmi and Gayathri Devi [8] evaluated Refactoring is done to develop the quality of a software system's structure which tends to reduce as the system evolves. While manually determining useful refactoring is a challenging task search based techniques can automatically discover suitable refactoring. Refactoring approach uses the concept of pare to optimality which naturally applies to search based refactoring. Before refactoring is done, the test case should be generated. A formal written test case is considered by a known input and by an expected output, which is worked out before the test is performed.

Woo- Chang Shin and Jungkyn Rho [9] characterized a refactoring tool that can modify the internal structure of software to a more easily understandable and modifiable structure that enormously affects software maintenance productivity. This paper proposed a code model to support software maintenance tool developers to easily access and handle software source codes. Also it displayed the implementation method of the software refactoring operation.

John Grundy and John Hosking [10] proposed several tools that have been established to support automation in both narrow and broad domain ranging across Artificial Intelligence (AI) tool kits such as theorem proves and model checkers requirements, design coding and testing support tools. They verified various configuration management process enhancement and project management support tools and code generators, code analysis, visualization, refactoring and reverse engineering tools.

3. Software Refactoring

Process of altering a software system in such a way that it does not modify the external behaviour of the code , it develop the internal structure, such that it increases code quality, reliability and maintainability throughout life cycle, makes software easier to understand and improves maintainability

3.1. Feature modeling

Software consists of features such as A, B, C, D, E and F As per the need of customers,these needed features can be given to them through refactoring with the help of the FOD tool. FOD tool could help to divide and run the features as per the need by the customers

4. Metrics of FOD

Table 1. Average Performance for FOD before and after Refactoring

Tools	Before Refactoring with KLOC	After Refactoring with KLOC
SPS	10	5
KAPTUR	15	10
CTA	20	10
DESIRE	6	5
FOD	5	3

Table 2. Average Performance of CBO and LSCC

TOOLS	CBO	LSCC
SPS	7.9	5.45
KAPTUR	8.8	3.8
CTA	3.5	3.02
DESIRE	3.5	1.7
FOD	10.5	9.5

The first, metric CBO (Coupling Between Object) classes symbolizes the number of classes combined to a given class. The coupling is achieved by the following factors such as method calls, field accesses, inheritance , arguments return types, and exceptions.

The second metric Low Level Class Cohesion (LLCC). Metric denotes the classes, Meaningful class coupling and cohesion metric helps object-oriented software developers identify class design weaknesses and refactoring classes consequently. The results show that LLCC is better than CBO metric.

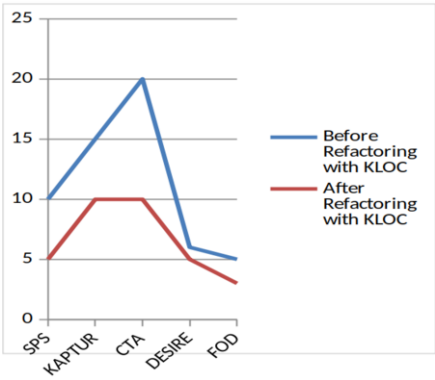

Figure 1. KLOC Vs Tools with Before and After Refactoring

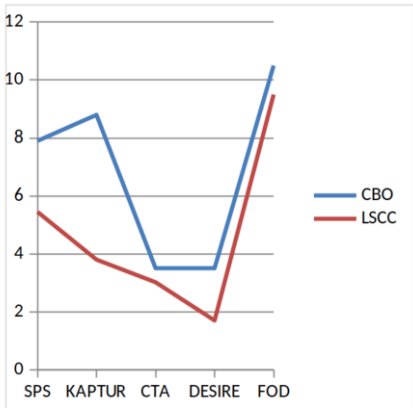

Figure 2. KLOC Vs Tools with CBO and LSCC

The advantage of the proposed system is to improve the code readability. Computational cost or Complexity is reduced by using the propose methods. It also improves the performance of the system. In Table 1 shows the average performance of FOD before

and after refactoring. Some of the automobile spare parts project, comparing five different tools with FOD, The thousands of (Kilo) Lines Of Code (KLOC) is decreased in our proposed work as shown in the table. Table 2 shows average performance of FOD tool is compared with other tools. The FOD tool gives better results than other tools. Figure 1 shows the accuracy performance for FOD before and after refactoring, number of lines are reduced after refactoring. FOD tool provides better results compare to other tools. Figure 2 shows the accuracy performance for FOD before and after refactoring, number of lines are reduced after refactoring. FOD tool provides better results compare to other tools such as SPS, KAPTUR, CTA, DESIRE. It shows FOD tool gives better results than other tools, based on clarity of the code and speed of the execution.

5. Conclusion and Future work

This research work proposed to FOD Tool which is established and executed successfully. By this tool any application software is refactoring. Any application software is developed through FOD. Each modules separately given to the users and run independently. Our experimental results displays the efficiency of our approach compared with existing approaches and different others meta heuristic approaches. We plan to study and implement this model in several number of other mission critical applications, their features and different languages which could be possibly applied by FOD tool so as to create new applications, this extension would require the possibly of adding new features in the FOD tool, for example, economical decisions over the extended feature model and checking a product which cost less than others.

References

[1] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. 1990 Nov 1:01-94.
[2] John McGregor D. Journal of object technology. Chair of Software Engineering. 2004;3(7):71-81.
[3] Kaur S, Maini R. A comprehensive review of refactoring techniques. International Journal of Latest Technology in Engineering, Management & Applied Science. 2015 Oct;4(10):78-83.
[4] Pérez J, Crespo Y, Hoffmann B, Mens T. A case study to evaluate the suitability of graph transformation tools for program refactoring. International Journal on Software Tools for Technology Transfer. 2010 Jul;12(3):183-99.
[5] Sangeetha V, Sangeetha M. Fascinating Perspective of Code Refactoring. International journal of Advanced Research in Computer Science and Software Engineering. 2016 Jan;16:164-8.
[6] Devi U, Sharma A, Kesswani N. A Study on the Nature of Code Clone Occurrence Predominantly in Feature Oriented Programming and the Prospects of Refactoring. International Journal of Computer Applications. 2016;141(8):39-44.
[7] Griswold WG, Opdyke WF. The birth of refactoring: A retrospective on the nature of high-impact software engineering research. IEEE Software. 2015 Sep 23;32(6):30-8.
[8] Ramalakshmi B, Devi DG. An Efficient Sdmpc Metric Based Approach For Refactoring Software Code. International journal of Engineering and computer science. 2015 May;14:11733-42.
[9] Shin WC, Rho J. Implementation of software refactoring operation using a code Model. International journal of Software Engineering and its Applications. 2014;8(6):17-30.
[10] Grundy J, Hosking J. Guest editors introduction: special issue on innovative automated software engineering tools. Automated Software Engineering. 2013 Jun;20(2):137-9.