A Methodology for Refactoring Legacy Code

S.A.M.Rizvi, Zeba Khanam
Department of Computer Science, Jamia Millia Islamia,
New Delhi, India
zebs_khan@yahoo.co.in
euphoria2.zeb@gmail.com

*Abstract—***Refactoring techniques have gained popularity due to their practical value in creating more agile code. Refactoring activities usually aim at improving the software quality, making them easier to understand, maintain, improvements on the software artifacts. Aspect-oriented programming (AOP) is receiving an increased attention due to its power in encapsulating crosscutting concerns. Refactoring allows reorganizing code while preserving the external behavior, while AOP facilitates modularizing crosscutting concerns in a system through use of a new unit of modularity called aspect. A combination of the two---aspect-oriented refactoring---helps in reorganizing code corresponding to crosscutting concerns to further improve modularization and get rid of the usual symptoms of crosscutting: code-tangling and code-scattering in legacy systems. The poorly designed procedural code when refactored with aspect orientation yields a better code. The aim of this paper is to establish a discipline for refactoring that will define the activities to be followed needed for refactoring.Our methodology is broken into two steps: first step is the preparation phase and the second is searching phase.This methodology is used by us in our research work [10].The set of activities described in this paper will help establish a proper refactoring procedure for the legacy code.**

Keywords: Refactoring methodology,legacy systems,aspect oriented programming.

I. INTRODUCTION

A methodology for refactoring proposed in ths paper is an organized set of interrelated activities which encapsulates a core concern of the process. This discipline containing the refactoring [4][2][3][5][6] activities for legacy software [7] can help the developers to organize the refactoring process in order to minimize the required efforts and maximize the effective results.The discipline is established to refactor the code making use of aspect oriented techniques [1].The complete refactoring procedure is shown in the diagram below. Section 2 focuses on the preparation phase of the refactoring procedure.Section 3 presents the main activities of the refactoring procedure in order to show how the activities are related and what is there recommended order. The developer chooses a refactoring which will be applied and uses our technique to increase confidence that the refactoring will not introduce behavioral changes in the code, which is passed as argument for our technique.Therefore ,this paper proposes a methodology that can be employed in order to apply the refactoring activities on the legacy system, employing the aspect oriented techniques.This methodology helps in establishing a proper refactoring process with predefined activities.The diagram below depicts the methodology

978-1-4244-8679-3/11/$26.00 ©2011 IEEE

II. PREPARATION

The activities of this phase are described in the following sections**.**

A. Refactoring Motive

The legacy code [8] is written in bad programming style, the code "works", but you need to refactor in order to put the code into a form you can understand and work with in order to extend its functionality. The main objective of refactoring is code clean up.

- **Extracting a reusable component**- Companies follow the business rules to conduct their businesses daily where the business rules are often implemented into the software systems. Over time, business rules evolve and the software that implemented the business rules is also modified to respond to the new business requirements. Unfortunately, timely updates on the documents corresponding to the software are usually ignored. The documents gradually become outdated and less useful for reference. A method for identifying and extracting business rules from legacy code by means of data identification and program slicing. Therefore one of the main objectives is extracting a reusable component.

- **Increasing loose coupling between components-** A loosely coupled code is more easily modifiable but the code. The more a code is loosely coupled the better it is. So one of the refactoring objectives is also to increase the loose coupling. This would be easily achieved through aspect oriented concept.

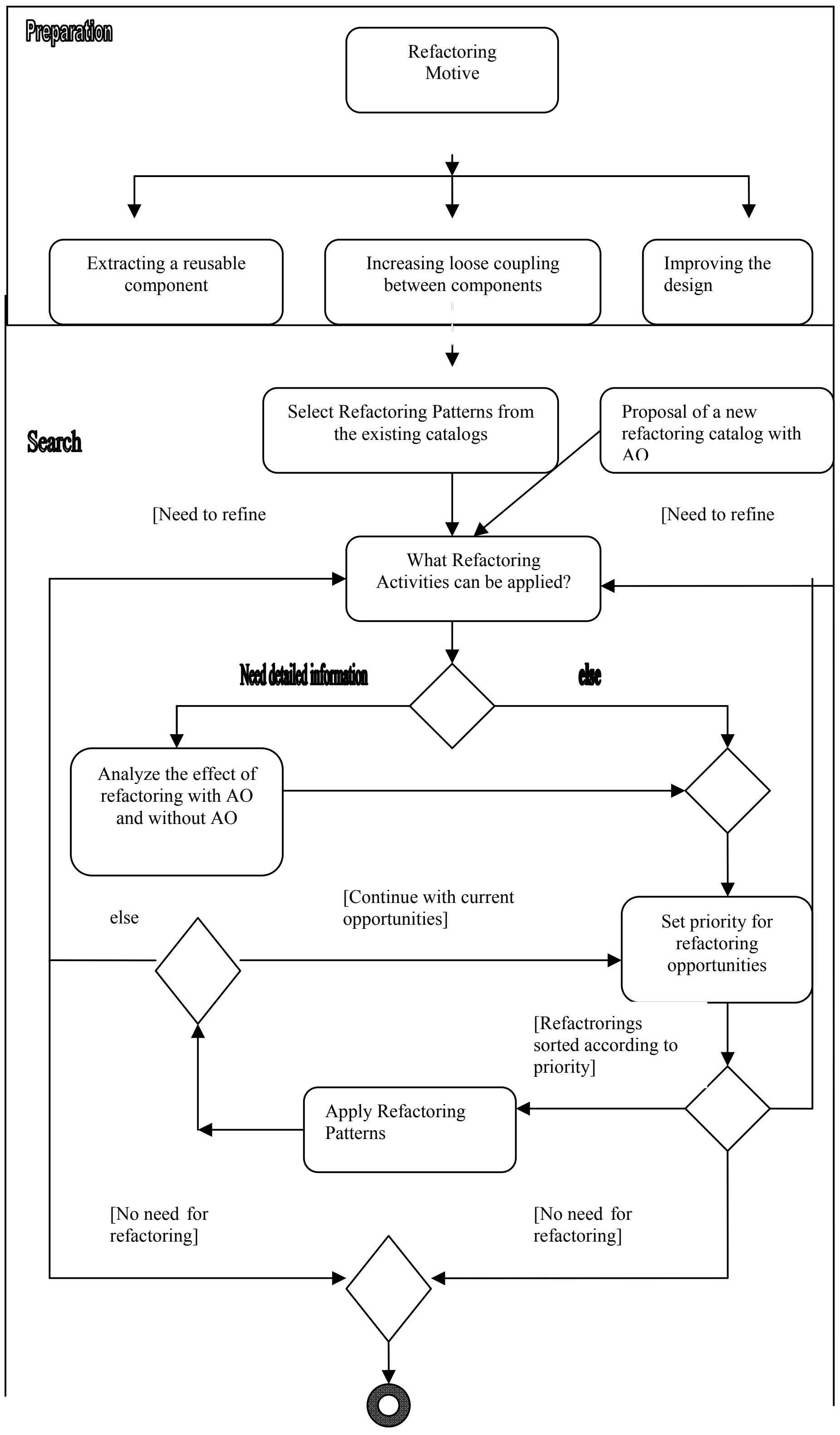
Preparation
Refactoring Motive
Extracting a reusable component
Increasing loose coupling between components
Improving the design
Search
Select Refactoring Patterns from the existing catalogs
Proposal of a new refactoring catalog with AO
[Need to refine
[Need to refine
What Refactoring Activities can be applied?
Need detailed information
else
Analyze the effect of refactoring with AO and without AO
[Continue with current opportunities]
else
Set priority for refactoring opportunities
[Refactrorings sorted according to priority]
Apply Refactoring Patterns
[No need for refactoring]
[No need for refactoring]

Figure 1:Main Activities:Methodology Overview

- **Improving the design-** Refactoring is an approach that facilitates the continuous change of source code, enabling it to evolve in line with changes in environments and requirements thus improving the design. The use of AOP further enhances the design.

III. SEARCH

The next phase is searching for refactoring opportunities in the software. After assuring the motive of refactoring the next phase is searching for places where to refactor the code. The legacy systems cannot also be refactored using proper tools as the tools are not applicable on these systems and tools using AOP are not yet available. So, here we deal with manual refactoring techniques using the concept of AOP.This phase describes the systematic approach to be followed for searching the refactoring opportunities. In the next section the various steps are described.

- **Select Refactoring Patterns-**-Several known refactorings exists, but selection from amongst the available refactoring catalogs must be such that it meets the refactoring motive that are defined in the flow chart.
- **Proposal of New AO refactorings—**Parallel to the selection of refactoring catalogs, we propose our own catalog of refactorings [9][10] with aspect oriented programming. The next step that follows is the selection of refactorings according to the situation and applicability.
- **What Refactoring Activities can be applied-**Once the selection of refactoring patterns are made out of the existing[11] and novel techniques ,the developer's job is to select the refactoring patterns and they should be applied according to their impact on the overall quality of the software.
- **Set priority for refactoring opportunities-**All the selected refactorings cannot be applied randomly, they have to applied according to their priority; some of them have to be applied in sequence also. The next step is to set the priorities for the refactoring patterns. One of the methodologies is to set the priority by ranking the patterns. The use of impact functions can also be done to compute the effect of refactoring.
- **Apply Refactoring Patterns-**After evaluating the refactoring patterns and ranking them on different parameters of software quality, the next step ahead is to apply the refactoring patterns on the software elements, re run the test cases and ensure that the refactoring patterns did not break anything. Their effect will be computed, the result will be analyzed. If needed the effect of the refactoring can be undone.
- **Analyze the effect of refactoring with AO and without AO-**One of the major contribution in our work is the usage of aspect orientation in refactoring the legacy code. So a major task in the process is also to compute the effect of aspect orientation in our work. It is to be analyzed as to how the usage of AO is affecting the software quality parameters and what are the drawbacks of using aspect orientation for the purpose.

IV. CONCLUSION

Usually there is a room for improvement in existing software projects. However the resources must be directed to those activities that bring more benefits to the project. Considering the refactoring activities that are more likely to improve the software design and quality, the developers should adopt an approach that would focus on a restricted set of refactoring patterns. Without focus, the developers could be losing time with refactoring opportunities that bring little to the overall quality of the software being developed. There are several activities involved in a refactoring process considering improvements on the software quality. In this paper a systematic approach to refactoring legacy software is adopted. First there is a need to define the refactoring motive to drive the entire process. After establishing the motive the refactoring patterns should be selected from the existing catalog and the new catalog (with AO).The next activity determines as to where and how these refactorings can be applied. Having known the refactoring patterns, the developer can then search for refactoring opportunities and evaluate the effects of refactoring.This methodology can be used by the developers to refactor the legacy code in a systematic manner and would help them in moving in the right direction rather than trying to apply every single refactoring existing in different catalogs.

REFERENCES

[1] Elrad T. (moderator) with panelists Aksit M., Kiczales G., Lieberherr K., Ossher H.,Discussing Aspects of AOP. Communications of the ACM, pp. 33-38, October 2001.

[2] Fowler M. (with contributions by Beck K., Opdyke W., Roberts D.). Refactoring –Improving the Design of Existing Code. Addison Wesley 1999.

[3] Griswold W. G., Program restructuring as an aid to software maintenance. PhD thesis, University of Washington, USA, 1991.

[4] Dijsktra E., A Discipline of Programming, Prentice Hall, 1976

[5] Opdyke W. F., Refactoring Object-Oriented Frameworks, Ph.D. Thesis, University of Illinois at Urbana-Champaign, USA, 1992

[6] Parnas D. L., On the criteria to be used in decomposing systems into modules. Communications of the ACM 15 (12), pp. 1053-1059, December 1972.

[7] Michael F., Working Effectively With Legacy Code, 2002

[8] Growth P., Refactroring Legacy Code, 2007

[9] Rizvi S.A.M and Khanam Z., Introduction of AOP techniques for refactoring legacy software , International Journal of Computer Application, 1(7):90-94, DOI, Published by Foundation of Computer Science, United States, ISSN: 0975-8887,25th Feb, 2010

[10] Rizvi S.A.M and Khanam Z., A systematic approach to refactoring legacy systems (In preparation)

[11] Cole L., Borba P., Deriving Refactorings for AspectJ. Proceedings of the 4th International Conference on Aspect-Oriented Software Development (AOSD 2005), ACM press, pp. 123-134, Chicago, USA, March 2005.