Impact of Refactoring on Software Quality

Prabhjot Kaur
Computer Science and Engg. Department
Punjab Technical University
Jalandhar, India

Puneet Mittal
Computer Science and Engg. Department
Baba Banda Singh Bahadur Engineering College
Fatehgarh Sahib, India

***Abstract*— Refactoring is the process of transformation of software. It changes its internal structure of software without affecting its external behavior. Bad smells means there are potential problems in the code, which we have to refactor. In this paper, we use two tools for bad smells detection on object oriented open source software are PMD and JDeodrant. Then after refactoring bad smells, we analyze the impact of refactoring on external quality attribute of software.**

***Keywords*— Refactoring, Metrics, Quality of Software**

I. INTRODUCTION

A. Refactoring

Refactoring code is the process of reorganizing code with the intent of simplifying both design and structure while not changing the functionality of the code [4]. By refactoring, we makes the code clean. It enhances the quality of software by removing potential problem in code. It minimizes the chances of introducing of bugs in the code.

B. Bad Smells

Bad smells are the defect in design of the software. Code smells are called as Bad smell. It is an indication of flaws in code which to be removed by applying appropriate refactoring technique. Fowler [16] given 22 bad smells and their respective 72 refactoring techniques to remove these bad smells. There are various tools to detect the bad smells. We detect 6 bad smells in the code. JDeodrant tool will detect 4 bad smells are- God class, Long method, Feature envy and Type Checking. Bad smell detected by PMD are- Dead code and Long parameters list.

- God Class- means a large class. Too many functions in one class so it's difficult to understand functionality of class.
- Long Method-means very long method in a class.
- Feature envy-means a class that is more interested to use function or methods of another class.
- Type Checking-it is switch statement bad smell. It has more duplication of code. So it is best to use polymorphism instead of switch statements.
- Dead Code- means variable, methods and classes that does not perform any functionality in software.
- Long Parameter List-means too many parameters are passed in parameter List.

C. Refactoring Techniques

Technique used for refactor the bad smells are called as Refactoring Techniques. These are the set of procedures to remove bad smell or clean the code. There are some refactoring techniques [16] used are-

- Extract Method - means extracting the set of statements into a new method.
- Extract Class – means extracting the set of methods and statements into a new class from the old class.
- Move Method – means methods from one class to another more relevant class.
- Remove or Delete – means delete the unused imports, local variables, unused private methods.
- Replace Conditional with Polymorphism- means replace switch conditional statements with polymorphism.
- Introduce Parameter Object–means replace parameter with an object.
- Replace Parameter with Method Call – means replace parameter passing function with value getting code inside the class.

D. Tools Used

- Eclipse- It is an Java based open source Integrated Development platform. It is designed in such a way that it can be extensible using plugins. It supports various languages C, C++, JAVA, PHP and COBOL. We can integrate bad smell detection plugins and refactoring tools into eclipse for refactoring the code.
- JDeodrant Plugin- is an eclipse plugin which are used for detecting and removing bad smells in the code. It can detect four bad smells are –Feature envy, God Class, Type Checking and Long Method. It detects bad smells in java based code.
- PMD Plugin-is a static java source analyze. It can be integrated into eclipse as a plugin. It can detect bad smells are- Dead code, Long Parameter List and Duplication Code.
- Metrics 1.3.6 Plugin- is a quality calculation tool. It can calculates No. of attributes, Lines of code, No. of classes, Weighted methods per Class, Cohesion and Coupling.

E. Quality Attributes

Software Quality Attributes are the characteristics of software by which quality is described and evaluated. It is divided into two groups- Internal Quality Attributes and External Quality Attributes. Internal Quality is measured directly from the

code. External quality attributes are measured with the help of internal quality attributes.

Internal Quality Attributes are-

- Lack of Cohesion
- Coupling
- Depth of Inheritance
- Number of Classes
- Lines of Codes
- Weighted Method per Class
- Abstractness

External Quality Attributes are-

- Understandability
- Reusability
- Functionality
- Effectiveness
- Flexibility
- Extendibility

II. LITERATURE SURVEY

Fowler et al. [16] describes the 22 bad smells and their 72 respective techniques to refactor bad smells. They mention duplicate code as a serious kind of bad smell. It increases maintenance cost of software. Due to increasing use of open source software and its variants, there is also increased use of code reuse. Due to Code reuse, it results in duplication of code.

Bansiya and Davis [10] presented a QMOOD (Quality Model for Object Oriented Designed) that access quality attributes like reusability, functionality, extendibility, flexibility, understandability, effectiveness. QMOOD relates low level design properties such as encapsulation, coupling and cohesion to high level quality attributes. It weighted quality attributes accordance to their influence and importance in the system.

Alshayed [1] investigate the effect of refactoring on software quality attributes. He focused on quality attributes like adaptability, maintainability, understandability, reusability and testability. They apply refactoring on three open source software- UML tool, terpPaint, Rabtpad. But after refactoring, he concludes that it does not necessary that refactoring improve the quality attributes of software.

Kannangara and Wijayanaka [21] investigate the impact of refactoring on internal quality attributes are maintainability, DIT, LOC, coupling. They compare quality attributes of nonrefactored code with refactored code. After study, they get result that only maintainability is improve, other attributes does not show any positive effect.

Tsantalis et al. [15] presented a tool, JDeodrant which is implemented as a plugin in Eclipse that automatically indentifies God class. They remove these smells by extract class refactoring. They also indentify the application of extract class refactoring in bad smells removal.

Kaur and Kaur [14] provide a review on bad smell detection tool PMD and JDeodrant using eclipse. They discuss and compare the bad smell detected by PMD and JDeodrant and their refactoring. They apply refactoring techniques on Online Exam System which is written in java. They refactor 6 bad smells using tools.

III. PROBLEM FORMULATION

To maintain the poorly design system is difficult and tough work. Software goes through various evolutionary development lifecycle, and then its quality attributes degrade. So it becomes difficult for developer to maintain the understandability, extensibility and reusability of the software. So refactoring is a way to maintain overall functionality and behavior of the system. So we proposed to detect the bad smells in object oriented open source java software and remove these by refactoring techniques. Then after refactoring, we analyze the impact of refactoring on external quality attributes of the software.

- To find different bad smells in an open source softwares.
- To analyze various refactoring techniques.
- To clean code by removing these bad smells through refactoring.
- To analyze the impact of refactoring on software quality before and after refactoring.

IV. RESEARCH METHOLOGY

A. *Methodology*

JDeodrant and PMD are bad smells detector. Object oriented open source software are JChart 2D (3.2.1),GhantProject(9.11) and RabtPad(0.1).

1. Measure internal quality attributes of the software.
2. Detect code bad smell by using PMD and JDeodrant tools in the software.
   a. Indentify the type of bad smell detected.
   b. Indentify location of bad smell detected.
   c. Refactoring techniques to be applied to it to refactor bad smell.
   d. Check, is there any error occur during refactoring, if not then move to next step to find next bad smell, otherwise rollback refactoring and applied another refactoring technique to refactor bad smell.
3. Then after removing bad smells in the codes, measure the internal quality attributes of software.
4. Calculates the external quality attributes of software with the help of internal quality attributes. Compare the external quality attributes of software before and after refactoring to analyze the impact of refactoring.

B. *External Quality Attributes.and Internal Qualiy Attributes*

TABLE I. Shows the external quality attributes given by bansiya[12]

External QA	Formula Used for Calculation
Reusability	-0.25*Coupling+0.25*Cohesion+0.5* Messaging+ 0.5*Design Size.
Flexibility	0.25*Encapsulation - 0.25*Coupling + 0.5*Composition + 0.5* Polymorphism.
Understand-ability	-0.33*Abstraction+0.33*Encapsulation-0.33*Coupling+0.33* Cohesion-0.33*Polymorphism-0.33*Complexity-0.33*Design Size.
Functionality	0.12*Cohesion + 0.22*Polymorphism + 0.22*Messaging + 0.22*Design Size +0.22*Hierarchies.
Extendibility	0.5*Abstraction - 0.5*Coupling + 0.5*Inheritance +0.5* Polymorphism.
Effectiveness	0.2*Abstraction + 0.2*Encapsulation + 0.2*Composition+ 0.2* Inheritance+ 0.2*Polymorphism.

TABLE III. Shows the Intenal Quality Formula

Design Property	Metrics used by Bansiya	Metrics we Used
Design Size	Design Size in Classes (DSC)	Number of Classes
Hierarchies	Number of Hierarchies (NOH)	Depth of Inheritance Tree
Abstraction	Average Number of Ancestors (ANA)	Abstractness
Encapsulation	Data Access Metrics (DAM)	(Total no. of attributes –static Attributes) / (Total no. of attributes + static Attributes)
Coupling	Direct Class Coupling (DCC)	Instability
Cohesion	Cohesion Among Methods of Classes(CAM)	1/Lack of Cohesion of Methods
Composition	Measure of Aggregation (MOA)	Number of Overridden Methods
Inheritance	Measure of Functional Abstraction (MFA)	No. of Overridden Methods /Number of Methods
Polymorphism	Measure of Polymorphic Methods (NOP)	Number of Overridden Methods
Messaging	Class Interface Size (CIS)	Number of Methods
Complexity	Number of Methods (NOM)	Weighted Methods per Class

V. RESULTS

Number of bad smells detected is shown in table IV. Table V and table VI shows the internal quality attributes before and after refactoring respectively. Then table V and table VI shows the external quality attributes values before and after refactoring respectively.

TABLE IV. Show the number of bad smell detected in software

Bad smells	RabtPad	JChart2D	GhanttProject
God Class	10	8	33
Feature envy	5	9	34
Long method	27	16	150
Type Checking	6	-	6
Dead Code	31	5	70
Long Parameter List	-	2	3

TABLE V. Shows Internal Quality Attributes of Software Before Refactoring

Softwares / Metrics	RabtPad	JChart2D	Ghantt Project
Coupling	0.252	0.406	0.397
Cohesion	2.551	2.463	0.2595
Messaging	7.581	3.991	6.189
Design Size	1.824	9.727	9.211
Encapsulation	0.4819	0.1797	0.5288
Composition	8.581	1.411	4.709
Polymorphism	0.258	0.467	0.417
Abstraction	0.044	0.0851	0.206
McCabe Complexity	18.871	8.533	15.034
Inheritance	0.966	0.882	0.937
Hierarchies	2.744	3.636	2.166

TABLE VI. Shows Internal Quality Attributes of Software After Refactoring

Softwares / Metrics	RabtPad	JChart2D	Ghantt Project
Coupling	0.296	0.623	0.397
Cohesion	3.164	2.898	0.2595
Messaging	10.12	4.236	6.189
Design Size	2.941	12.727	9.211
Encapsulation	0.4903	0.337	0.5288
Composition	5.38	1.543	4.709
Polymorphism	0.34	0.350	0.417
Abstraction	0.077	0.0865	0.206
McCabe Complexity	17.44	7.786	15.034
Inheritance	0.967	0.9174	0.937
Hierarchies	2.18	3.043	2.166

TABLE VII. Shows External Quality Attributes of Software Before Refactoring

External Quality Attributes	RabtPad	JChart2D	GhanttProject
Reusability	5.277	7.373	7.665
Flexibility	4.476	0.882	2.595
Understandability	-6.011	-5.469	-8.077
Functionality	3.042	4.216	3.987
Extendibility	0.508	0.514	0.581
Effectiveness	2.066	0.604	1.359

TABLE VIII. Shows External Quality Attributes of Software After Refactoring

External Quality Attribute	RabtPad	JChart2D	GhanttProject
Reusability	7.247(↑)	9.050(↑)	11.431(↑)
Flexibility	2.908(↓)	0.875(↓)	1.731(↓)
Understandability	-5.755(↑)	-6.051(↓	-7.469(↑)
Functionality	3.807(↑)	4.826(↑)	5.710(↑)
Extendibility	0.544(↑)	0.365(↓)	0.662(↑)
Effectiveness	1.450(↓)	0.646(↑)	1.000(↓)

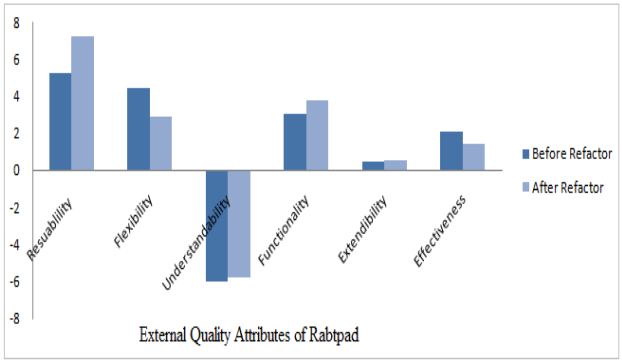

Fig. 1. Shows External Quality Attributes of RabtPad before Refactoring and after Refactoring.

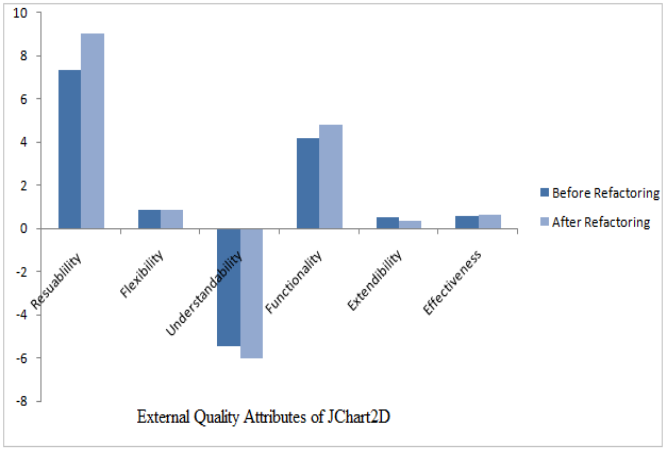

Fig. 2. Shows External Quality Attributes of JChart2D before Refactoring and after Refactoring.

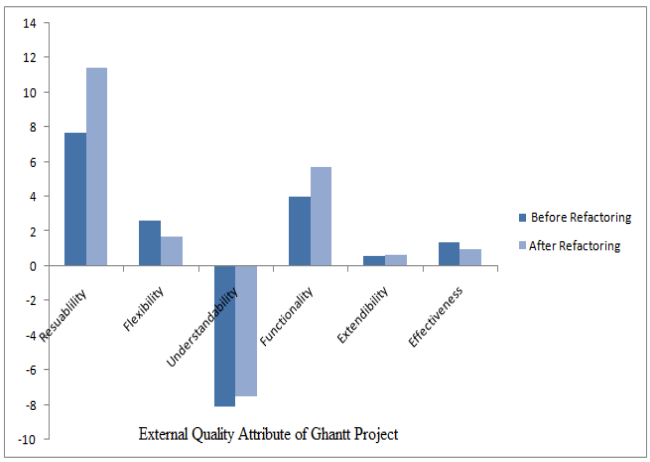

Fig. 3. Shows External Quality Attributes of GhanttProject before Refactoring and after Refactoring.

VI. CONCLUSION

In this paper, we notice the impact of refactoring of refactoring on external and internal quality attributes. It shows that reusability and functionality of all three shows an postive impact. But other external attributes like extendibility, understandability and flexibility shows negative impact on software after refactoring. So conclude from the experiment that group of refactoring techniques have postive and negative impact on software.

In future work, we can detect more bad smells and then apply more refactoring techniques to software to refactor these smells in a code. Then, after refactoring analyze the impact on quality.We can also check impact of refactoring on other quality attributes like testability.

REFERENCES

[1] M. Alshayeb, “Empirical Investigation of Refactoring Effect on Software Quality”, Information and Software Technology, ELSEVIER, 2009.

[2] W. Opdyke, “Refactoring Object-Oriented Frameworks”, PhD thesis, University of Illinois at Urbana-Champaign, 1992.

[3] A. Rani and H. Kaur, “Refactoring Methods and Tools”, International Journal of Advanced Research in Computer Science and Software Engineering, vol.2, No. 12, 2012.

[4] D. Jonsson, “Detecting Code Smells in Educational Software: an in-depth study by David Jonsson”, March 2013.

[5] D. Roberts, “Practical Analysis for Refactoring”, PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1999.

[6] S. Kaur and S. Singh, “Spotting and Elimination Type Checking Code Smells using Eclipse Plug-in: JDeodorant”, International Journal of Computer Science and Communication Engineering, v.5, No.1, 2016.

[7] EMF Metrics Plugin, URL – Retrieved from http://sourceforge.net/projects/metrics/.

[8] PMD, URL--Retrieved from http://pmd.sourceforge.net/eclipse/.

[9] ISO/IEC9126,”Software product evaluation- Quality characteristics and guidelines for their use”, 9126 Standard, Information technology, 1991.

[10] J. Bansiya and C.G. Davis, “A Hierarchical Model for Object-Oriented Design Quality Assessment,” IEEE Transactions on Software Engineering, vol. 28, no. 1, pp. 4-17, 2002.

[11] JDeodorant, URL - Retrieved from https://marketplace.eclipse.org/content/jdeodorant.

[12] K.O. Elish and M. Alshayeb, ” A Classification of Refactoring Methods Based on Software Quality Attributes”, Arabian Journal for Science and Engineering, Springer, 2011.

[13] K.O. Elishand M. Alshayeb, “Investigating the Effect of Refactoring on Software Testing Effort”, 16th Asia-Pacific Software Engineering Conference,IEEE, 2009.

[14] S. Kaur and S. Kaur, “Review on Identification and Refactoring of Bad Smells using Eclipse”, International Journal for Technological Research in engineering, v.2, Issue 7, March 2015.

[15] N. Tsantalis, M. Fokaefs and E. Stroulia, “JDeodorant: Identification and Application of Extract Class Refactoring”, ICSE ‘11, USA, 2011.

[16] M. Fowler, K. Back, J. Brant, W.Opdyke and D.Roberts. “Refactoring: Improving the Design of Existing Code”, Addison-Wesley, New York, 1999.

[17] M. Mantyla,“Bad Smells in Software – a Taxonomy and an Empirical Study”, Master Thesis, Department of Computer Science, Helsinki University of Technology, 2003.

[18] A. Chatzigeorgiouand A. Manakos, “Investigating the Evolution of Bad Smells in Object-Oriented Code”, International Conference on the Quality of Information and Communications Technology, IEEE, 2010.

[19] N. Tsantalis and A.Chatzigeorgiou, “Identification of move method refactoring opportunities”, IEEE Transactions on Software Engineering, pp.347–367, 2009.

[20] N. Kumari and A. Saha, ”Effect of Refactoring on Software Quality “, Academy and Industry Research Collaboration Center, v.4, pp.37-46, 2014.

[21] S.H. Kannangara and W.H.J.I Wijayanaka, “An Empirical Evaluation of Impact of Refactoring on Internal and External Measures of Code Quality”, International Journal of Software Engineering and Applications, v.6, No.1, 2015

[22] A. Sharma and S.K. Dubey, “Comparison of Software Quality Metrics for Object-Oriented System”, International Journal of Computer Science & Management Studies (IJCSMS), ISSN (Online): 2231 –5268, vol. 12, 2012.