International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 8 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT183833

Impact of Clone Refactoring on External Quality Attributes of Open Source Softwares

Er. Himanshi Vashisht, Sanjay Bharadwaj, Sushma Sharma
[1]Computer Science and Engineering Department, Haryana Engineering College, Jagadhri, Haryana, India
[2,3]Computer Science and Engineering Department, DAV College for girls, Yamunanagar, Haryana, India

ABSTRACT

Code refactoring is a "Process of restructuring an existing source code.". It also helps in improving the internal structure of the code without really affecting its external behaviour". It changes a source code in such a way that it does not alter the external behaviour yet still it improves its internal structure. It is a way to clean up code that minimizes the chances of introducing bugs. Refactoring is a change made to the internal structure of a software component to make it easier to understand and cheaper to modify, without changing the observable behaviour of that software component. Bad smells indicate that there is something wrong in the code that have to refactor. There are different tools that are available to identify and emove these bad smells. A software has two types of quality attributes- Internal and external. In this paper we will study the effect of clone refactoring on software quality attributes.

Keywords : Clone refactoring, Metrices, External Quality Attributes

I. INTRODUCTION

Refactoring is basically the behavior preserving process. Code duplication is a serious problem with software. Due to code reuse, it leads to duplicate code in software. Roy et al.[2] discussed various clone detection tools and techniques.If a code segment is copied with some changes like addition or deletion of statements and alters its variables name, functions and type, then it comes under type-3 or near miss clones [4].

Software refactoring is the super-set of software restructuring. Martin Fowler [1] book "Improving the Design of Existing Code" describes different 22 bad smells in code and techniques to remove these bad smells. Refactoring is the method of altering the software system in such a way that its external behavior does not change but its internal structure is enhanced. Refactoring only modifies the internal structure of software so that it will be easy to maintain the software in the future. Refactoring reduces the complexity of software and make it easy to understand for user.

Refactoring Techniques

The technique that is used to remove clones is called as Refactoring Techniques. These are set of measures and steps to keep software clean. There are some basic techniques for clone proposed by Fowler [1]:

- **Extract Method-** is applied when the clone segment are to be found in methods that belong to the same class. In this condition, extract unified code in a new private method within the same class [5].
- **Extract and Pull up Method-** is applied when the clone segments are to be found in methods that

belong to different sub classes of the same super class. In this situation, unified code is placed in a new protected method in the super class [5].

- **Introduce Template Method**- is a unique case of the refactoring techniques. If clones do not belong to previously clone types but have same return type and identical signature. Then we create an abstract method with same signature in super class where unified code is pulled up [5].
- **Introduce Utility Method**- is applied when the clone segment is to be found in methods of dissimilar classes and the segments do not access any instance method or variables. In this situation, we extract a unified code into a static method placed within a utility class [5].

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. Metrics calculation tool will calculate internal quality attributes. External quality attributes are measured with the help of internal quality attributes.

Internal Quality Attributes are [6] -

- Lack of Cohesion
- Coupling
- Number of Classes
- Abstractness
- Depth of Inheritance
- Lines of Codes
- Weighted Method per Class
- Complexity
- Hierarchies
- Design Size
- Polymorphism
- Encapsulation

External Quality Attributes are [6] -

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

II. PROPOSED ALGORITHM

- ✓ Fowler et al. [1] described the 22 bad smells and their 72 respective techniques to refactor bad smells. They also associated refactoring rules with these bad smells, suggesting how to resolve these bad smells. They declared duplicate code as a serious kind of bad smell. It increased 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.
- ✓ The International Organization for Standardization (ISO/IEC9126) et al. [2] published a set of international standards related to the software engineering, such as ISO 12207 and ISO 9126. However, there is a set of cross-references between the two standards. The ISO 9126 on software product quality and ISO 12207 on software life cycle processes had been analyzed to investigate the relationships between them and to make a mapping from the ISO 9126 quality characteristics to the ISO 12207 activities and vers versa. This study presented a set of comments and suggestions to improve the ISO 9126. The weaknesses of the cross references between the two ISO standards had been highlighted. In addition, this study provided a number of comments and suggestions to be taken into account on the next version of the ISO 9126 international standard.
- ✓ Kamiya et al. [3] proposed a clone detection tool CCFinder (Code Clone Finder). This tool incorporates the use of a lexical analyzer which removes the white spaces, comments from source code and generate token sequence of code, Then after, token sequence is transformed

using certain rules. This transformation regularizes the identifiers by partially removing the context information. A special token replaces the identifiers so that code portions with different variable names could be returned as clone pairs by the matching algorithm.

- ✓ Garg and Tekchandani [4] introduce an approach to refactor the clones on the basis of their essentiality. The approach measures the maintenance overhead in terms of repetitiveness, size of clones and complexity. They find clones using CCFinder clone detection tool. After detection of clones, calculate efforts required in maintaining clones. They arrange clones according to their value of maintenance overhead. The clones which having high value should be refactor first.
- ✓ Tstanalis et al. [5] propose an approach to check the refactorability of clones. They defined pre-condition which are checked during refactorability. If these pre-condition are satisfied, then we can remove clones easily. If these are violated, then refactorability of that clone is not possible. They used four clone detector tools- CCFinder, Deckard, CloneDR, Nicad.They found that clone with a close distance tends to be more refactorable than more distant. Type 1 clones are more refactorable than other types of clones.
- ✓ Bansiya and Davis et al. [4] 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. They weighted quality attributes accordance to their influence and importance in the system.
- ✓ Fontana et al. [7] investigates the impact of clone refactoring on quality attributes internal quality attributes like complexity, coupling and cohesion. They used three clone detection tools PMD, Bahumas and CodePro on two open source software– Ant and GhanttProject. Intellij IDEA tool is used for refactoring. They analyze that, after refactoring there is improvement in cohesion, decrement in coupling, complexity and lines of code.
- ✓ Alshayed et al. [8] investigates the effect of refactoring on software quality attributes. He focused on quality attributes like adaptability, maintainability, reusability, understandability and testability. They apply refactoring on three open source software- terpPaint, UML tool and Rabtpad. But after refactoring, he concludes that it does not necessary that after refactoring there is increase in quality of software.

III. PROPOSED STRATERGY

CCFinder [19] is used as bad clone detector tool. Jdeodrant [15] is a refactoring plugin which is used to refactor the clones according to their respective techniques. Eclipse metrics plugin [27] used to calculates the internal quality attributes of source code. Object oriented open source software are JChart 2D 3.2.1 [14], apache-ant 1.7.0 [2], JMeter2 2.3.2 [17] and JEdit 4.2 [16].

Steps used to re-factor clones and to calculate the quality attributes are as following-

1. Before applying any single refactoring, calculate the internal quality metrics (Object Oriented Metrics) of software.
2. Detect Clones in software using Clone detection tool CCFinder.
3. Then import result file of clone detection in Jdeodrant plugin.
   a. Identify where the software should be refactor.

b. Make a small change i.e. a single refactoring without changing the outer behavior of the software.
c. Test Refactor code, if works and move on to the next refactoring.
d. If fails, rollback the last smaller change and repeat the refactoring in some different way.

4. After applying all the refactoring techniques, calculate the internal quality metrics of software (Object Oriented Metrics) to determine the impact of refactoring.
5. Calculate the external quality attributes by using internal quality metrics.
6. Compare external quality attributes of software to predict the impact on software quality.

IV. IMPLEMENTATION FRAMEWORK

.

The external quality attributes are dependent on the internal quality attributes. Therefore, attributes can be calculated by using these formulas given by Bansiya and Davis [6].

Table 1- External Quality attributes Formula

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.
Understandability	-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.

Internal Quality attributes

Internal Quality attributes are calculated by Eclipse Metrics [14] plug-in .We interpret these values to calculate metrics used by Bansiya [6].

Table 2- Internal Quality Attributes formula

Design Property	Metrics we Used	Formulas
Design Size[6]	Number of Classes	$DS = \sum_{i=1}^{p} NOC$ where, NOC = Total number of classes in a package, p = number of packages.
Hierarchies [6]	Depth of Inheritance Tree	$Hierarchies = DIT$ DIT = Depth of inheritance tree.
Abstraction [6]	Abstractness	$Abs = \frac{\sum_{i=1}^{n} NoI}{n}$ Where NoI = total number of interfaces in a package n=total number of classes in a package.
Encapsulation [6]	(Total no. of attributes –static Attributes) / (Total no. of attributes + static Attributes)	$Enc = \frac{a(P)}{a}$ Where, a(P) = number of private attributes in a class, a = total number of attributes in a class.
Cohesion [6]	1/Lack of Cohesion of Methods	$LCOM = \frac{\left(\frac{1}{a}\sum_{i=1}^{n} m(A)\right) - m}{1 - m}$ here, m(A)= number of methods accessing an attribute A, then Calculate the average of m(A) for all attributes, m = total numbers of methods for all classes, a = total number of attributes in a class

		n= number of classes.
Composition [6]	Number of Overridden Methods	$Composition = \sum_{i=1}^{n} NOA$ where, NOA = Total number of Attributes in a class, n = number of classes.
Inheritance [6]	No. of Overridden Methods /Number of Methods	$Inheritance = \left(\sum_{i=1}^{n} \frac{NORM}{NOM}\right) \times 100$ where, NORM= number of overridden method in a class,
Polymorphism [6]	Number of Overridden Methods	$Poly = \sum_{i=1}^{n} NORM$ where, NORM = number of overridden methods in a class, n = number of classes.
Messaging [6]	Number of Methods	$Messaging = \sum_{i=1}^{n} NOM$ where, NOM = the total number of public methods in a class, n = number of classes.
Complexity [6]	Weighted Methods per Class	$WMC = \sum_{i=1}^{m} Ci$ Ci= complexity of method i in a class, m= number of methods.
Coupling [6]	Instability	$CBO = Ce$ Where Ce= efferent coupling .

V. RESULTS

In this section, impact of clones refactoring on quality of softwares is analyzed by comparing various quality attributes.

Number of clones Detected in Software

In research work, three types of clones have been detected on four different open source softwares JChart 2D (3.2.1), apache-ant (1.7.0), JMeter (2.3.2), JEdit (4.2) using CCFinder. Table III provides information about the number of clones detected in the open source softwares using CCFinder tool.

Softwares	**JChart2D**	**Apache-ant**	**JMeter**	**JEdit**
TLOC	6693	115744	81307	81004
Clones	248	2798	2018	969

Table 3- number of clones smell detected in software

Refactoring Impact on Internal Quality Attributes of Software

To To find the impact of clones refactoring, first calculate internal quality attributes of software without applying any refactoring technique. After removal of clones, calculate internal quality attributes. Internal quality attributes values before refactoring and after refactoring is shown in Table IV and Table V respectively.

Table 4 - Internal Quality Attributes of Software Before Refactoring

Softwares / Metrics	JChart2D	Apache-ant	JMeter	JEdit
Design Size	9.727	11.361	5.406	21.471
Hierarchies	3.636	2.689	2.914	2.382
Abstraction	0.0851	0.086	0.111	0.078
Encapsulation	0.8403	0.405	0.035	0.467
Coupling	6.818	7.205	4.383	6.882
Cohesion	2.463	2.890	2.336	3.731
Composition	1.411	2.597	2.424	2.985
Inheritance	0.882	0.123	0.123	0.158
Polymorphism	0.467	1.024	1.091	0.901
Messaging	3.991	8.266	8.85	5.685
Complexity	8.533	18.15	17.896	21.126
TLOC	6693	115744	81307	81004

Table 5 - Internal Quality Attributes of Software After Refactoring.

Softwares / Metrics	JChart2D	Apache-ant	JMeter	JEdit
Design Size	10.091	11.62	5.584	21.706
Hierarchies	3.73	2.748	2.957	2.385
Abstraction	0.0865	0.091	0.114	0.081
Encapsulation	0.8324	0.402	0.036	0.464
Coupling	7.182	7.388	4.497	7.059
Cohesion	1.855	2.92	2.415	3.759
Composition	1.36	2.538	2.368	2.951
Inheritance	0.891	0.128	0.124	0.156
Polymorphism	0.45	1.065	1.094	0.892
Messaging	4.153	8.293	8.79	5.707
Complexity	8.198	17.869	17.434	20.854
TLOC	6651	115300	81213	80065

Refactoring Impact on Complexity of Software

Table VI, Shows the refactoring impact on complexity of software. From Figure 1 and Figure 2, it is clear at after refactoring weighted method per class and MCcabe cyclomatic complexity of all the software is reduced. So refactoring shows positive impact on complexity.

Table 6 - Impact of Refactoring On Complexity of Software

Complexity	**McCabe Cyclomatic Complexity**		**Weighted methods per Class**	
Softwares	Before Refactoring	After Refactoring	Before Refactoring	After Refactoring
JChart2D	2.015	1.876(↓)	8.533	8.198 (↓)
Apache-ant	2.109	2.0699(↓)	18.150	17.869(↓)
JMeter	1.864	1.822(↓)	17.896	17.434(↓)
JEdit	3.161	3.095(↓)	21.126	20.854 (↓)

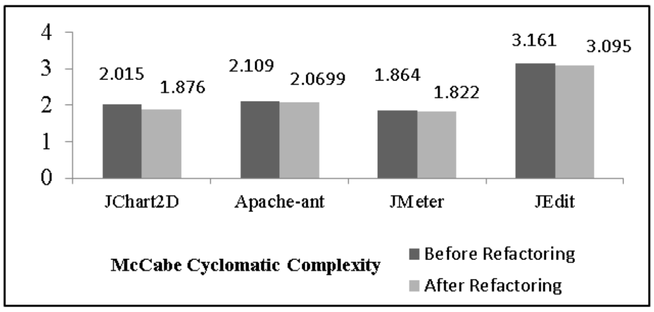

Figure 1. Impact of clones refactoring on McCabe Cyclomatic Complexity of software

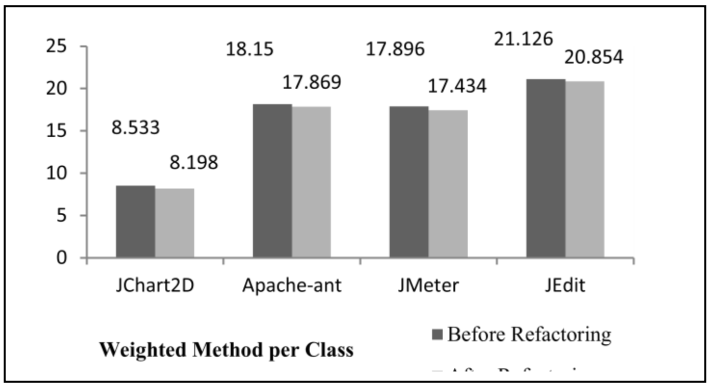

Figure 2. Impact of Clones refactoring on weighted method per class

External QA	JChart2D	Apache-ant	JMeter	JEdit
Reusability	5.790↑)	8.839(↑)	6.665(↑)	12.881(↑)
Flexibility	-0.682(↓)	0.055(↓)	0.615(↓)	1.302 (↑)
Understandability	-7.695(↓)	-11.456(↓)	-8.669(↑)	-15.981(↓)
Functionality	4.275(↑)	5.570(↑)	4.343(↑)	7.655(↑)
Extendibility	-2.877(↓)	-3.052(↓)	-1.582(↓)	-1.935(↑)
Effectiveness	0.723(↓)	0.844(↓)	0.747(↓)	1.3206(↑)

Refactoring *Impact on External Quality Attributes*

The external quality attributes are calculated by using formulas given by Bansiya and Davis [6]. According to the formula given above, the values of external quality attributes are shown in Table VII and Table VIII.

Table 8- External Quality Attributes Values Before Rfactoring

External Quality Attributes	**JChart2D**	**Apache-ant**	**JMeter**	**JEdit**
Reusability	5.770	8.734	6.616	12.790
Flexibility	-0.555	0.110	0.684	0.339
Understandability	-7.367	-11.395	-8.750	-15.265
Functionality	4.216	5.481	4.297	7.144
Extendibility	-2.691	-2.986	-1.529	-2.872
Effectiveness	0.737	0.847	0.764	0.917

Table 9 - External Quality Attributes Values After Refactoring

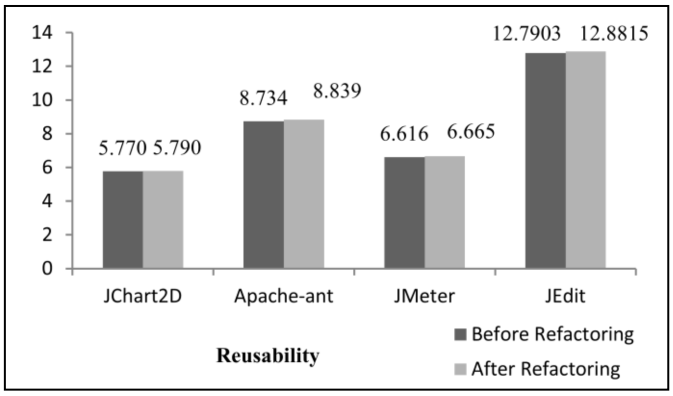

Figure 3. Impact of Refactoring on Reusability of software

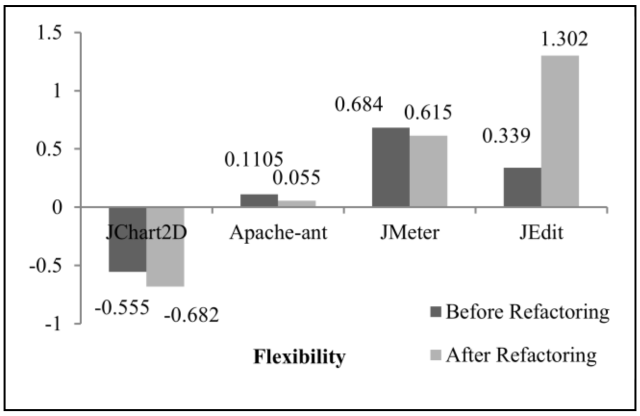

Figure 4. Impact of Refactoring on Flexibility of software

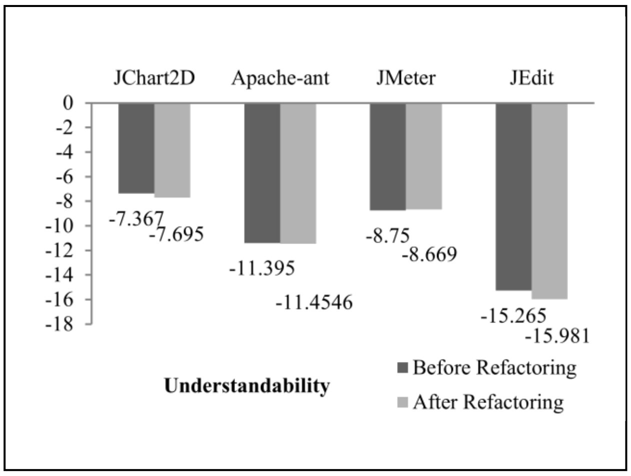

Figure 5. Impact of Refactoring on Understandability of software

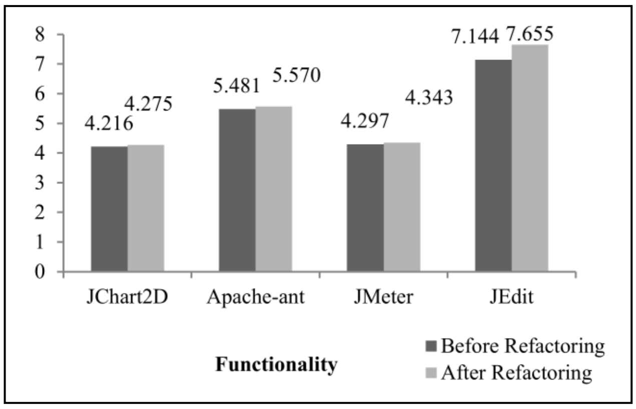

Figure 6. Impact of Refactoring on Functionality of software

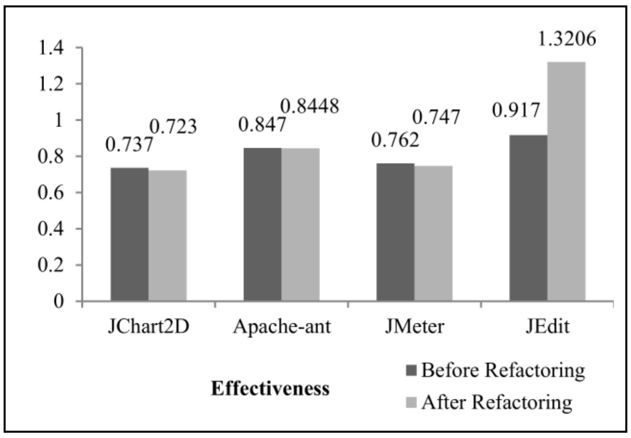

Figure 7. Impact of Refactoring on Effectiveness of software

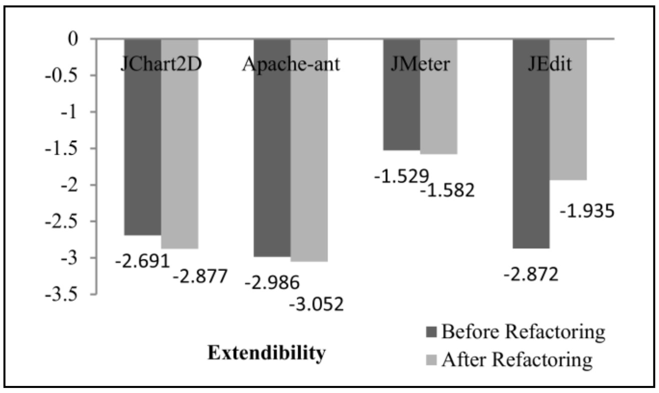

Figure 8. Impact of Refactoring on Extendibility of software

As shown in Figures 3 and Figure 5, Reusability and functionality of all the four open source softwares increased when refactoring is applied. In Figure 4, flexibility of JChart2D, Apache-ant, JMeter is decrease, but only JEdit flexibility is increased. In Figure 5, understandability of JChart2D, Apache-ant, JEdit is decrease, but only JMeter show slight improvement. In Figure 7 and Figure 8, effectiveness and extendibility of JChart2D, apache-ant, JMeter decreased, only JEdit attributes values increased a huge amount.

VI. CONCLUSION AND FUTURE WORK

Refactoring makes code easy to use. In this work four, different softwares are used to analyze the impact of clones' refactoring on quality of softwares. From experimental results, conclusion comes out that the complexity of the softwares is reduced using refactoring. By applying refactoring on softwares, reusability and functionality of all the softwares is increased and other quality attributes like flexibility, understandability, effectiveness, extendibility is decreased. Some refactoring techniques improved the quality of softwares and some refactoring techniques shows negative effect on quality Result shows that refactoring techniques may also have inverse effect on software quality attributes.

VII. REFERENCES

[1]. M. Fowler, K. Back, J. Brant, W. Opdyke and D.B. Roberts, "Refactoring: improving the design of existing code," Addison-Wesley, New York, 1992.

[2]. C.K. Roy, J.R. Cordy and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of Computer Programming *ELSEVIER,* vol. 74, pp. 470–495, 2009.

[3]. T. Kamiya, S. Kusumotoand K. Inoue, "CCFinder: a multilinguistic token based code clone detection system for large scale source code," IEEE Transaction on Software Engineering, vol. 28, no. 7, pp. 654-670, 2002.

[4]. R. Gargand and R. Tekchandani, "Enhancing code clone management by prioritizing code clones," Master's Thesis, Thapar University, Patiala, 2014.

[5]. N. Tsantalis, M. Mazinanian and G.P. Krishnan, "Assessing the refactorability of software clones, " IEEE Transactions on Software Engineering, vol. 41, No. 11, 2016.

[6]. 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.

[7]. F.A. Fontana, M. Zanoni, A. Ranchetti and D. Ranchetti, “Software Clone Detection and Refactoring,” ISRN Software Engineering, 2013.

[8]. M. Alshayeb, “Empirical investigation of refactoring effect on software quality, ” Information and Software Technology, ELSEVIER, 2009.

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

[10]. JChart2D Retrieved from https://sourceforge.net/projects/jchart2d/.

[11]. Apache-ant Retrieved from http://ant.apache.org/.

[12]. JMeter Retrieved from http://jmeter.apache.org.

[13]. JEdit Retrieved from http://www.jedit.org. Metrics Plugin, URL – Retrieved from http://sourceforge.net/projects/metrics