A Fuzzy Model for Evaluating the Maintainability of Object Oriented System Using MOOD Metrics

Gurpreet Kaur[1], Mehak Aggarwal[2]
[1]Research Scholar, [2]Associate Professor
[1,2]Department of Computer Science and Engineering, Lala Lajpat Rai Institute of Engineering & Technology, Moga, (Punjab) INDIA
[1]gurpreet_7288@yahoo.co.in, [2]aggarwal.mehak@gmail.com

***Abstract*: The research on object-oriented metrics has a great contribution in improving the quality of object-oriented systems. Many software metrics have been proposed by different researchers such as Chidamber and Kemerer (CK), Lorenz & Kidd, Rosenberg, Abreu etc. These metrics measure the OO design mechanisms such as polymorphism, inheritance, Encapsulation, Information hiding, Cohesion, Coupling etc which greatly influence the various quality characteristics of the software's such as reusability, reliability, maintainability and so on. Fernando Brito e Abreu has proposed a set of six object-oriented metrics namely MOOD Metrics and presented the correlation between these metrics and the Normalized Rework (maintainability). But the results given by Abreu are not quantitative and vague in nature. In this research work, a decision making system based on fuzzy inference is proposed that quantifies the maintainability of object-oriented system using the MOOD Metrics. This system is finally validated using the empirical results given by Abreu.**

***Keywords*- MOOD Metrics, Fuzzy Logic, Maintainability, Object Oriented System.**

I. INTRODUCTION

The main goal of software engineering is to produce better software with high quality. However, control of quality is impossible unless it becomes quantifiable. Thus, fitting the software quality on a measurable basis is an important work on which many studies have been performed but is yet an open ended subject. In the International Standard ISO/IEC 9126 [8], a software quality model is established based on the Factor-Criteria-Metrics (FCM) Quality Model [12]. According to the standard, quality is affected by factors and factors are assessed via criteria. The main factors determining software quality are given as functionality, reliability, usability, efficiency, maintainability and portability. In the standard, mapping of these factors to criteria is described clearly but still it is ambiguous which criterion is mapped with which metrics. The factors defined are dependent on hardware used, technology involved, design of software, etc. The most design-dependent factor among all is maintainability. This makes this factor one of the favorites since most of the metrics are also design-dependent. When it comes to designing software, the most popular design technique is OO design. OO programming approach is more maintainable than the procedural ones [7] and it introduces the object concept as well as features like inheritance, encapsulation and polymorphism. OO design increases modularity compared to functional design. Modularity increases understandability thus maintaining the code becomes easier. As OO paradigm has introduced new concepts and features; new metrics [13] [14] like CK, Morris's and MOOD for measuring inheritance, polymorphism, coupling and data hiding. So today's requirement is to established the clear (exact) relationship between quality attribute and metrics which is quantifiable as well as not vague in nature. Fuzzy Logic is one of the new concepts that can be used to remove the vagueness. The outline of the paper is following: Section II presents the object-oriented features that affect the quality characteristics of object-oriented systems. Section III elaborates the MOOD metrics suite for OO design. Section IV portrays the correlation between MOOD Metrics and the quality measure Normalized rework (maintainability) given by Abreu [1][2][3]. Section V represents the purposed Fuzzy Model and Section VI concludes with the proposed validation.

II. FEATURES OF OBJECT ORIENTED SYSTEM

Object Oriented Paradigm has become quite popular over recent years and has completely replaced the Procedural Oriented Paradigm. Object-Oriented Programming (OOP) encourages software reuse by providing design and language constructs for modularity, encapsulation, inheritance and polymorphism.

A. Encapsulation

Encapsulation implies that the non-essential details of an object are hidden from the user and an access is provided to its essential details. Therefore, encapsulation is also called information hiding. It provides security to the data and the methods of a class.

B. Inheritance

In object-oriented methodology, inheritance enables you to extend the functionality of an existing class. You create a class that inherits the attributes and behavior of another class.

C. Polymorphism

Polymorphism is derived from two Latin words-Poly, which means many, and morph, which means forms. Anything that exists in more than one form is known as a polymorph. It enables you to assign a different meaning or usage to an entity in different contexts.

D. Coupling

Coupling is the degree to which each program module relies on each one of the other modules. It is interdependency

between different components or functions. It is measure of interconnections among the modules.

III. MOOD METRICS USED FOR CALCULATING ABOVE FEATURES

We have used object oriented metrics suite proposed by Abreu [1][2][3] for measuring the use of object oriented design mechanisms. Following are the metrics-

A. Method Hiding Factor (MHF)
The MHF numerator is the sum of the invisibilities of all methods defined in all classes. The invisibility of a method is the percentage of the total classes from which this method is not visible. The MHF denominator is the total number of methods defined in the system under consideration.

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)}$$

B. Attribute Hiding Factor (AHF)
The AHF numerator is the sum of the invisibilities of all attributes defined in all classes. The invisibility of an attribute is the percentage of the total classes from which this attribute is not visible. The AHF denominator is the total number of attributes defined in the system under consideration.

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$

C. Method Inheritance Factor (MIF)
The MIF numerator is the sum of inherited methods in all classes of the system under consideration. The MIF denominator is the total number of available methods (locally defined plus inherited) for all classes.

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}.$$

D. Attribute Inheritance Factor (AIF)
The AIF numerator is the sum of inherited attributes in all classes of the system under consideration. The AIF denominator is the total number of available attributes (locally defined plus inherited) for all classes.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

E. Polymorphism Factor (POF)
The POF numerator represents the actual number of possible different polymorphic situations. Indeed, a given message sent to class **Ci** can be bound, statically or dynamically, to a named method implementation. The latter can have as many shapes ("morphos" in ancient Greek) as the number of times this same method is overridden (in **Ci** descendants).
The POF denominator represents the maximum number of possible distinct polymorphic situations for class **Ci**. This would be the case where all new methods defined in **Ci** would be overridden in all of their derived classes.

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

F. Coupling Factor (COF)
The COF denominator stands for the maximum possible number of couplings in a system with TC classes. The client-supplier relation (represented by **Cc** =>**Cs**) means that **Cc** (client class) contains at least one non inheritance reference to a feature (method or attribute) of class **Cs** (supplier class). The COF numerator then represents the actual number of couplings not imputable to inheritance.

$$COF = \frac{\sum_{i=1}^{TC} [\sum_{j-1}^{TC} is\_client(C_i, C_j)]}{TC^2 - TC}$$

where

$$is\_client(C_c, C_s) = \begin{cases} 1 & iff\ C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 & otherwise \end{cases}$$

Each of these metrics refers to a basic structural mechanism of the object-oriented paradigm as encapsulation (MHF and AHF), inheritance (MIF and AIF), polymorphism (POF) and coupling (COF).

IV. CORRELATION BETWEEN MOOD METRICS AND MAINTANABILITY

The MOOD metrics are believed to quantify independent aspects of the design and therefore their effect on quality can be assessed individually. To provide some evidence about the relationship between OO design and software project quality, the correlations between the MOOD metrics and the quality measures of normalized rework (Maintainability) are determined [1]. The resulting coefficients of correlation are shown in Table 1.

Table 1: **C**orrelation coefficients of MOOD and quality measures

	MHF	**AHF**	**MIF**	**AIF**	**POF**	**COF**
NR	-0.569	-0.143	-0.780	-0.549	-0.707	0.907

Based on the data provided in Table 1, the following conclusions have been drawn:

- MHF has a moderate negative correlation with rework (a ***maintainability*** measure). This means that once MHF increases, the effort spent to fix defects

will be expected to decrease which increases the maintainability.

- AHF did not show any significant correlation.
- MIF has a high negative correlation with normalized rework measure. This means that once MIF increases the effort spent to fix defects will be expected to decrease which increases the maintainability. Very high values of MIF (above the 70% to 80% range) are believed to reverse this beneficial effect.
- AIF has a moderate negative correlation with normalized rework measure. **This result does not allow any strongly supported conclusions to be drawn.**
- POF has a moderate to high negative correlation with rework. This means that an appropriate use of polymorphism in OO project designs should decrease the rework. However, very high values of POF are expected to reduce these benefits.
- COF has a very high positive correlation with all quality measures. Therefore, as coupling among classes increases, the normalized rework is also expected to increase. This result shows that coupling in software systems has a strong negative impact on software quality and therefore should be kept to the minimum required during design. It is desirable that classes communicate with as few others as possible because coupling relations increase complexity, reduce encapsulation and potential reuse, and limit understandability and maintainability.

Here, we are not taking AHF and AIF factors because according to the interpretations given by Abreu, these metrics did not show any significant correlation.

V. FUZZY APPROACH FOR MAINTAINABILITY EVALUATION

A. Proposed Model

There are various methods to measure the maintainability but none of them was correct approach. Thus we propose a fuzzy model approach for maintainability measurement of an object oriented system. Fuzzy logic considers the fuzzy value instead of binary values [5][9][10]. The benefit of using fuzzy logic is that the fuzzy logic models can be built even with little or no data. In this paper, fuzzy model measure reworks (a ***maintainability*** measure) that that takes into consideration the MHF, MIF, POF and COF metrics.

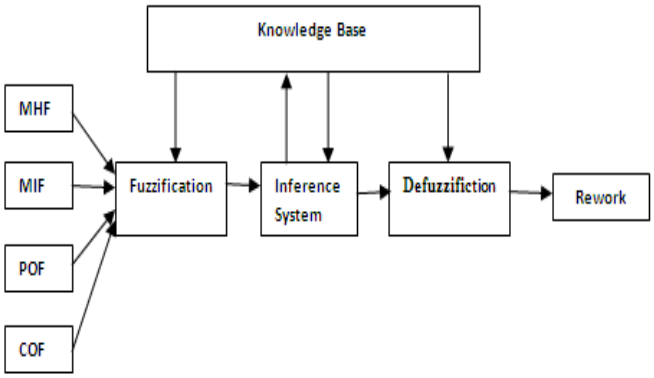

Figure 1: Block diagram of Fuzzy Model

B. Working of the model

In this model we have taken four inputs as MHF, MIF, POF, and COF to provide a crisp value of rework (a ***maintainability*** measure) using rule base. Fuzzy inference system (FIS) uses fuzzy logic to map the input to output. Mamdani fuzzy inference method is used. After the fuzzification process is completed, we take the fuzzy sets for output variables that require Defuzzification. For Defuzzification the input will be fuzzy set and output will be a singleton value. The centroid method which gives centre of area under curve is most commonly used for Defuzzification. There are many types of membership functions but for simplicity we have used triangular membership function.

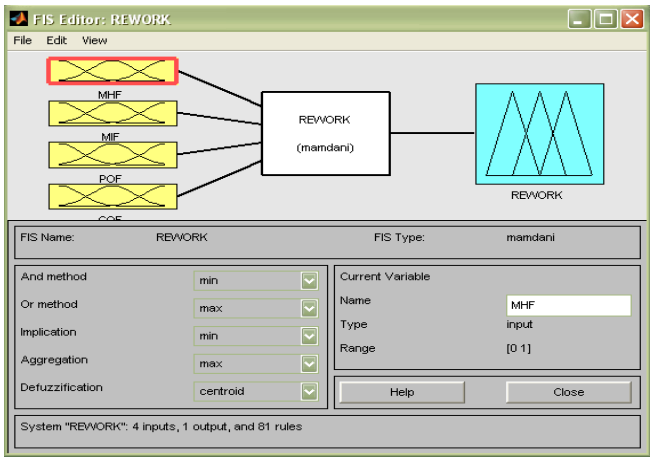

Figure 2: Inputs and Outputs of Fuzzy Model

C. Membership Function for Inputs and Outputs

For measuring rework (a ***maintainability*** measure) of an object oriented system we have considered four inputs-MHF, MIF, POF, and COF. These are shown in figure 3, 4, 5, 6. We have taken three membership functions- low, medium and high for each input. These inputs are taken on an interval of [0, 1].

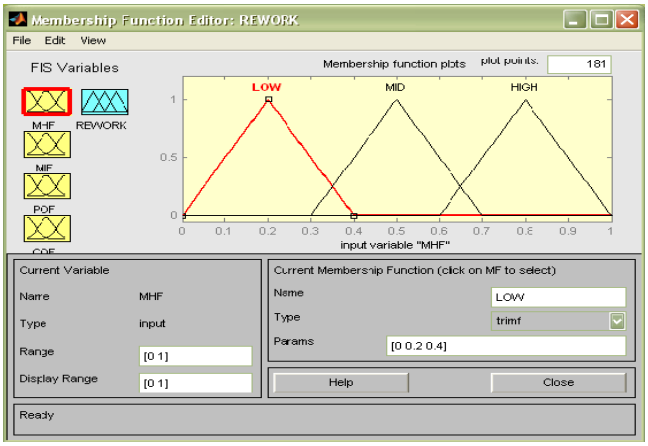

Figure 3: Membership function for MHF

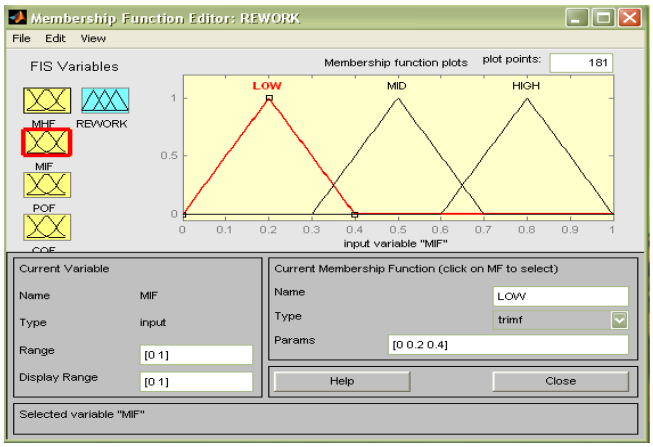

Figure 4: Membership function for MIF

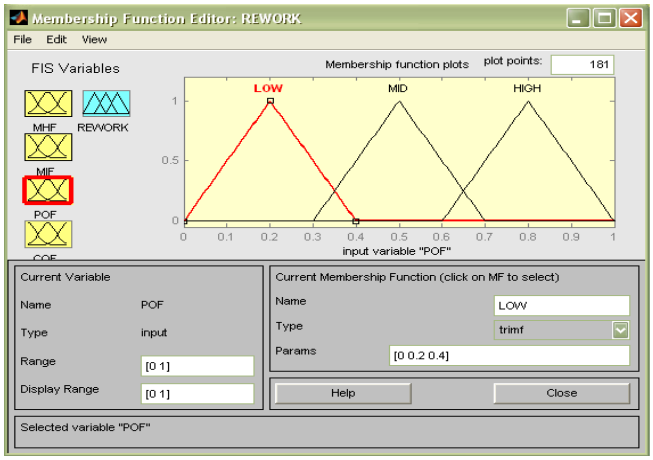

Figure 5: Membership function for POF

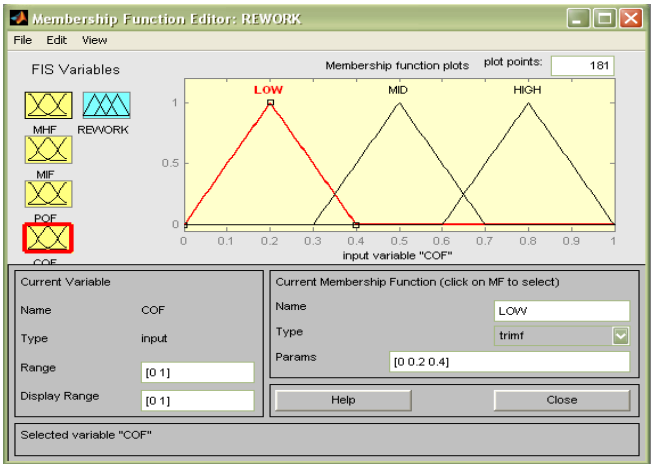

Figure 6: Membership function for COF

For the output (rework) we have taken five membership functions-very low (VL), low (L), medium (M), high (H) and very high (VH). The range for this is taken from [0,100]. This is shown in the Figure 7.

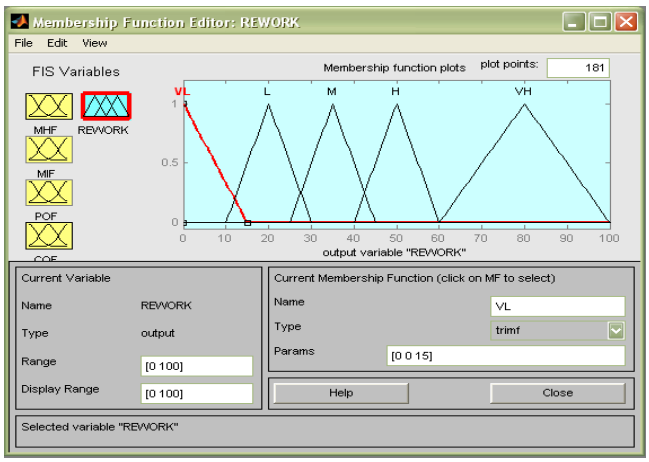

Figure 7: Membership function for Rework

D. Knowledge Base and Evaluation Process

In order to measure maintainability (rework) of a software system, all the four inputs (MHF, MIF, POF, COF) are integrated with the help of fuzzy model. Each of these inputs contains three terms- Low, Medium and High. Thus by integrating and forming different combinations for all the inputs we get 81 rules. In general terms if there are x inputs with y terms each then total number of rules R formed will be y*y*y…..x times. Thus R= y^x.

In our model we have 4 inputs and 3 terms. Hence our total number of rules will be 3^4= 81. For all 81 combinations rework (a ***maintainability*** measure) can be classified as very low, low, medium, high or very high. A survey is taken from n experts including project mangers, software developers, research scholars, maintainability experts to finalize the set of rules are found.

Table 2: Rules for Fuzzy Model

Rework Evaluation Using Features					
S. No	MHF	MIF	POF	COF	REWORK
1.	L	L	L	L	VL
2.	L	L	L	M	M
3.	L	L	L	H	H
.	.	.	.	.	.
10.	L	M	L	L	VL
11.	L	M	L	M	M
.	.	.	.	.	.
20.	L	H	L	M	M
21.	L	H	L	H	H
.	.	.	.	.	.
37.	M	M	L	L	L
38.	M	M	L	M	H
39.	M	M	L	H	H
40.	M	M	M	L	VL
.	.	.	.	.	.
61.	H	L	H	L	VL
62.	H	L	H	M	VL
.	.	.	.	.	.
81	H	H	H	H	VL

E. Metric Values

To find the values of all the features of object oriented system we need metrics. For this purpose we have chosen MOOD Metrics. The feature Encapsulation is related with (MHF), Inheritance is related with (MIF), Polymorphism is related with (POF) and Coupling is related with (COF). We have taken different values for these metrics. Metric values are given as input to the fuzzy model and the crisp value of rework is obtained using MATLAB rule viewer.

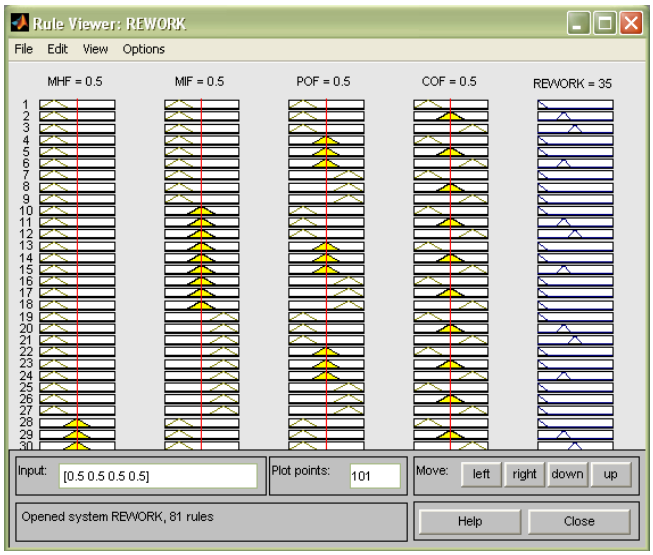

Figure10: Value of Rework obtained using Matlab

VI. VALIDATION OF PROPOSED MODEL

The proposed model is validated against correlations between the MOOD metrics and the quality measures of normalized rework (NR) given in Section IV. Table 3 shows the correlation between MHF and rework, Table 4 shows the correlation between MIF and rework, Table 5 shows the correlation between POF and rework, and Table 6 shows the correlation between COF and rework.

Table 3: Correlation of MHF with REWORK

S.NO	MHF	MIF	POF	COF	Experimental Results
1.	0.01	0.1	0.1	0.1	7
2.	0.02	0.1	0.1	0.1	6.84
3.	0.03	0.1	0.1	0.1	6.68
4.	0.04	0.1	0.1	0.1	6.52
5.	0.05	0.1	0.1	0.1	6.34
6.	0.06	0.1	0.1	0.1	6.18
7.	0.07	0.1	0.1	0.1	6.02
8.	0.08	0.1	0.1	0.1	5.87
9.	0.09	0.1	0.1	0.1	5.71
10.	0.10	0.1	0.1	0.1	5.57
11.	0.11	0.1	0.1	0.1	5.43
12.	0.12	0.1	0.1	0.1	5.3
13.	0.13	0.1	0.1	0.1	5.18
14.	0.14	0.1	0.1	0.1	5.07
15.	0.15	0.1	0.1	0.1	4.97

Table 4: Correlation of MIF with REWORK

S.NO	MHF	MIF	POF	COF	Experimental Results
1.	0.15	0.01	0.15	0.15	7
2.	0.15	0.03	0.15	0.15	6.68
3.	0.15	0.05	0.15	0.15	6.34
4.	0.15	0.06	0.15	0.15	6.18
5.	0.15	0.08	0.15	0.15	5.87
6.	0.15	0.09	0.15	0.15	5.71
7.	0.15	0.11	0.15	0.15	5.43
8.	0.15	0.12	0.15	0.15	5.3
9.	0.15	0.14	0.15	0.15	5.07
10.	0.15	0.2	0.15	0.15	4.97
11.	0.15	0.85	0.15	0.15	4.97
12.	0.15	0.86	0.15	0.15	5.07
13.	0.15	0.88	0.15	0.15	5.3
14.	0.15	0.91	0.15	0.15	5.71
15.	0.15	0.95	0.15	0.15	6.34
16.	0.15	0.97	0.15	0.15	6.68
17.	0.15	0.99	0.15	0.15	7

Table 5: Correlation of POF with REWORK

S.NO	MHF	MIF	POF	COF	Experimental Results
1.	0.4	0.4	0.1	0.4	50
2.	0.4	0.4	0.4	0.4	35
3.	0.4	0.4	0.6	0.4	35
4.	0.4	0.4	0.61	0.4	32.3
5.	0.4	0.4	0.62	0.4	29.7
6.	0.4	0.4	0.63	0.4	27.3
7.	0.4	0.4	0.64	0.4	24.9

8.	**0.4**	**0.4**	**0.65**	**0.4**	**22.4**
9.	**0.4**	**0.4**	**0.66**	**0.4**	**19.9**
10.	**0.4**	**0.4**	**0.67**	**0.4**	**17**
11.	**0.4**	**0.4**	**0.68**	**0.4**	**13.9**
12.	**0.4**	**0.4**	**0.69**	**0.4**	**10.1**
13.	**0.4**	**0.4**	**0.7**	**0.4**	**5.57**
14.	**0.4**	**0.4**	**0.99**	**0.4**	**7**
15.	**0.4**	**0.4**	**1**	**0.4**	**50**

Table 6: Correlation of COF with REWORK

S.NO	MHF	MIF	POF	COF	Experimental Results
1.	**0.2**	**0.2**	**0.2**	**0.2**	**4.67**
2.	**0.2**	**0.2**	**0.2**	**0.25**	**4.97**
3.	**0.2**	**0.2**	**0.2**	**0.3**	**5.47**
4.	**0.2**	**0.2**	**0.2**	**0.35**	**22.4**
5.	**0.2**	**0.2**	**0.2**	**0.4**	**35**
6.	**0.2**	**0.2**	**0.2**	**0.61**	**36.7**
7.	**0.2**	**0.2**	**0.2**	**0.62**	**38.2**
8.	**0.2**	**0.2**	**0.2**	**0.63**	**39.7**
9.	**0.2**	**0.2**	**0.2**	**0.64**	**41.1**
10.	**0.2**	**0.2**	**0.2**	**0.65**	**42.5**
11.	**0.2**	**0.2**	**0.2**	**0.66**	**43.9**
12.	**0.2**	**0.2**	**0.2**	**0.67**	**45.3**
13.	**0.2**	**0.2**	**0.2**	**0.68**	**46.8**
14.	**0.2**	**0.2**	**0.2**	**0.69**	**48.3**
15.	**0.2**	**0.2**	**0.2**	**0.7**	**50**

VII. CONCLUSION

This paper proposes a fuzzy model to evaluate maintainability of object-oriented software system. The inputs for the proposed model are MHF, MIF, POF, and COF, on which maintainability depends. These inputs were determined based on study and using extensive survey. Based on expert's knowledge rule base is generated with 81 rules for evaluating maintainability of object oriented software systems. Results are validated against correlations between the MOOD metrics and the quality measures of normalized rework (NR). This model will help software developers and researchers to analyze the maintainability of object oriented software system early in the design phase when various alternatives are presented before them. In future the model will be more refined by taking into consideration of other object oriented metrics.

REFERENCES

[1] Abreu, Fernando Brito e. and Melo, Walcelio. "Evaluating the impact of Object Oriented Design on Software Quality." Proceedings of the 3rd International, March 1996.

[2] Abreu, Fernando Brito e.; Esteves, Rita. and Goulão, Miguel. "The Design of Eiffel Programs: Quantitative Evaluation Using the MOOD Metrics" Originally published in Proceedings of TOOLS'96 USA,Santa Barbara, California, July 1996.

[3] Abreu, F. B.; Goulão, M. and Esteve, R. "Toward the Design Quality Evaluation of Object-Oriented Softwarec Systems". Proceedings of the 5th International Conference on Software Quality, Austin, Texas, USA,October 1995.

[4] Arora, Deepak.; Khanna, Pooja.; Tripath, Alpika i.; Sharma, Shipra. and Shukla, Sanchika. "Software Quality Estimation through Object Oriented Design Metrics". IJCSNS International Journal 100 of Computer Science and Network Security, VOL.11 No.4, pp. 100-104, April 2011

[5] Desai, Ankit. and Ganatra, Amit. "Fuzzy Based Refactoring Cost Resembler (FRCR) Model for Object Oriented Systems". International Journal of Computer Theory and Engineering Vol. 4, No. 2, pp. 251-258, April 2012.

[6] Ghosh, Soumi., Kumar Dubey, Sanjay., Prof. (Dr.) Rana, Ajay. "Comparative Study of the Factors that Affect Maintainability". International Journal on Computer Science and Engineering (IJCSE) Vol. 3 No. 12, pp. 3763-3769, Dec 2011

[7] Henry, S., Humphrey, M., Lewis, J., "Evaluation of the Maintainability of Object-Oriented Software", IEEE Region 10 Conference on Computer and Communication Systems, 1990.

[8] International Standard ISO/IEC 9126-1:2001, "Software Engineering -Product Quality - Part 1: Quality Model".

[9] Khalsa, Sunint K. "A Fuzzified Approach for the Prediction of Fault Proneness and Defect Density". Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, London, U.K, July 2009.

[10] Mago, Jagmohan.; Kaur, Sarabjit. and Kumar Saurabh.(2012) "Fuzzy Model to Analyze and Interpret Object Oriented Software Design" International Journal of Electrical, Electronics and Computer Engineering 1(1) pp. 41-46.

[11] Mayer, T. and Hall, T. "Measuring OO Systems: A Critical Analysis of the MOOD Metrics", Technology of Object-Oriented Languages and Systems, 1999.

[12] McCall, J., Richards, P., Walters, G., "Factors in Software Quality,Volume I", NTIS Springfield, 1977.

[13] Rodriguez, Daniel. and Harrison, Rachel. "An overview of Object Oriented Design Metrics" RUCS/2001/TR/A March 2001.

[14] Sarker, Muktamyee. "An overview of Object Oriented Design Metrics", pp. 1-53, June 2005.

[15] Turk, Tuna. "The Effect Of Software Design Patterns On OO Software Quality And Maintainability", pp. 1-68, September 2009.