Software Maintainability Modeling using Fuzzy Systems: Early Stage Perspective

Syed Wajahat Abbas Rizvi
Department of Computer Science and Engineering,
Amity University, Lucknow

ABSTRACT
Maintainability has been a big challenge for the information technology industry. Every stakeholder in the context of software application needs a maintainable software. The basis of this concern is the cost that the software maintenance consumes. In continuation with this crucial issue, this paper has developed a Maintainability prediction model that quantifies the software Maintainability through fuzzy techniques in the early phase of software development life cycle. The focus of the paper is the Maintainability quantification prior to the coding phase so that the personnel involved in developing the software should be able to take suitable and timely measure. If they get any input before the start of coding, then definitely they will do the correction in a cost-effective manner. This study identified product based object-oriented design measure and integrated them with fuzzy inference system. The developed model has also validated, along with appropriate predictive accuracy results.

Keywords
Software Maintainability, Early Stage Prediction, Fuzzy Logic, Software Defects, Software Metrics, Software Maintainability Model, Object-Oriented Design, UML Class Diagrams.

1. INTRODUCTION
Literature has defined the software maintainability as *"the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment"* [1, 2]. As it could be observed that the world is considered as a global village only because of the exponential growth of information technology. Every sector of society whether it is retail, defense, transportation, telecommunication, aircrafts, banking, home appliances, entertainment, education, e-governance, and so on, are highly influenced by computer software [3]. This level of dependence and trust on day to day applications is forcing the IT professionals to develop number of easy to use as well as maintainable software.

As a result, this inherent pressure, up to some extent, has been increasing the probability of committing errors and making the software less maintainable [4]. As it is a well-accepted fact in the software industry that software maintenance is an expensive and challenging phase in the life cycle of the software applications, developer has not been managing it properly and in some cases, it is often ignored. There may be various reasons for it, but one is improper management as well as the absence of appropriate metrics for software maintainability [5]. As class diagrams are the key elements of any object-oriented design documents therefore early estimation of their maintainability may help designers to incorporate required enhancements and corrections in order to improve their maintainability and consequently the maintainability of the final software to be delivered in future. Software maintainability has been considered as a critical factor therefore its prediction is of great significance. A precise quantification of this software quality attribute can be achieved through software maintainability models only in the last stages of development life cycle. However, with the objective of cost-effectiveness and timely management of resources its prediction in the early phases of software development is one of key area of concern [6]. In this paper, the researcher has developed fuzzy based maintainability modeling that has used the fuzzy inference system to quantify the maintainability of the developing software at the end of design stage [7]. The rest of the paper is organized as follows; section 2 briefly describes some of the related research works on maintainability prediction. Section 3 presents the proposed maintainability prediction model, the input and output variables involved in the fuzzy inference, their fuzzy profiles and rule base, while the develop model validated in the fourth section. Statistical validation followed by predictive accuracy measure for the model are presented in Section 5, and finally the paper concludes with possible future directions in section 6.

2. RELATED WORK
Researcher has scanned the literature of last two and half decades and found significant number of software maintainability quantification as well as estimation efforts. One key observation that is being noticed during the state of the art is that most of the models have quantified the maintainability during coding and testing phases of development life cycle, while few authors had suggested its quantification in the early phases like requirements or design [8]. Antonellis et al. [9], performed an effort where the researcher had mapped object-oriented coding-based measures onto lower level properties of maintainability highlighted in ISO 9126. In [10], Oman and Hagemeister developed the Maintainability Index that has also quantifies maintainability of software application through the metrics derived from the code of developing software.

In a study [11] authors measured the software maintainability using lines of comments, cyclomatic complexity and total numbers of lines of code. In [12], researcher had proposed a model that quantifies adaptive software maintenance effort in terms of DLOC (difference lines of code). Here hayes et al., considered number of added, deleted and updated lines in the application code. Polo et al. [13], used number of modification requests, mean effort per modification request and type of correction to examine maintainability. In another study Hayes and Zhao [14], proposed a maintainability model that separates modules of applications as 'easy to maintain' and 'not easy to maintain'. This effort assists software developers to know the modules those are not easy to maintain, before integrating them. Survey of the literature further revealed that various efforts has been done to develop several maintainability models, but mostly, utilizes the measures from and after the coding phase. Because of this, maintainability predictions are made in the latter stages of

SDLC, and it become very difficult to improve the maintainability at that stage. Muthanna et al. [15], developed a maintainability model using polynomial linear regressions. Genero et al. [16], developed four models that relate size and structural complexity metrics of UML class diagrams with maintainability measures like understandability time, modifiability correctness and modifiability completeness. But none of the four models quantify the maintainability of class diagrams itself.

In another study Kiewkanya et al. [17], had used the concept of weighted-sum method and developed a maintainability model. In that study the weighted values of understandability and modifiability of a class diagram are summed up, to get the maintainability value. One important finding is that multiple regressions might be used in place of weighted-sum. That would be more suitable than weighted-sum. That will reflect the impact for understandability and modifiability on maintainability. In a situation where actual maintenance data is not available specially at the requirements and design phases, maintainability of the developing software application can be measured through those software metrics that are maintainability relevant [18]. The concept of fuzzy sets was introduced by Zadeh [19] to represent vagueness in linguistics as a mathematical way. Literature has been witnessing a significant number of contributions [20, 21, 22, 23, 24] in the area of maintainability prediction. But, the power of fuzzy inference system has not been used to predict the maintainability in the design stage of object-oriented software applications. Fuzzy Logic techniques are emerging as robust optimization techniques that can solve highly complex, nonlinear, correlated and discontinuous problems [25]. As it can be easily noticed that software metrics available in the early phases are not very comprehensible and sometime reflect complex dependency among them. Consequently, fuzzy techniques have been found very helpful as far as handling and processing of subjective information is concern. On the basis of afore said critical findings it can be easily inferred that the fuzzy techniques have justified their utility to process information that is not objective in nature specially in the early phases of software development [26]. The key issue is how it is applied in making the software product more maintainable.

3. MAINTAINABILITY PREDICTION MODEL

After recognizing the criticality of early stage maintainability quantification of object-oriented software applications it is required to develop a model that will use the measures from the design phase. Therefore, this paper has focused on the identification of maintainability-relevant software measures for early prediction. For this, a comprehensive model has developed as shown in figure 1. The model shown in the diagram integrates object-oriented design metrics as input to the fuzzy inference system in order to quantify the maintainability of the developing software after its design stage completes, that is prior to the coding stage. The model is referred as Design Stage Maintainability Prediction Model (DSMPM) and based on the assumption, that not only maintainability but also quality of a software application is adversely impacted by the weaknesses of early stage constructs. Therefore, the model focuses on the most crucial early stage that is design phase. The model developed in this study predicting the maintainability using Fuzzy Inference System (FIS), therefore researcher has used fuzzy logic toolbox of MATLAB for implementing the model. The elementary steps of the model implementation include input/output variables identification, development of fuzzy profile and membership functions of the identified input/output variables and development of fuzzy rule base, followed by fuzzification and defuzzification. Following sections are focusing on these steps of model implementation.

3.1 Identify Software Metrics

In the past two and half decade several research works in the object-oriented metrics arena were produced [27]. However, as far as the aim or objective of this research is concern the focus is on the metrices those can measure the design documents in an effective manner.

Table 1. Size and Structural Complexity Metrics For UML Class-Diagrams [1]

Metric Name	Metric Definition
Number of Classes (NC)	The total number of Classes.
Number of Attributes (NA)	The total number of Attributes.
Number of Methods (NM)	The total number of Methods.
Number of Associations (NAssoc)	The total number of Associations.
Number of Aggregations (NAgg)	The total number of Aggregation relationships within a class diagram.
Number of Dependencies (NDep)	The total number of Dependency relationships.
Number of Generalizations (NGen)	The total number of Generalization relationships within a class diagram.
Number of Aggregation Hierarchies (NAggH)	The total number of Aggregation Hierarchies in a class diagram.
Number of Generalizations Hierarchies (NGenH)	The total number of Generalization Hierarchies in a class diagram.
Maximum DIT (MaxDIT)	It is the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalization hierarchy is the longest path from the class to the root of the hierarchy.
Maximum HAgg (MaxHagg)	It is the maximum between the HAgg value for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from class to leaves.

After a thorough review of the literature the metrics shown in the table are identified as a set of suitable design measures for class diagrams. Adaptation of object-oriented metrics in numerous application domains should only take place if the metrics are valid, in the sense that they accurately measure the attributes of software for which they were designed to

measure and have been validated empirically [28]. The metrics identified in the table has also empirically validated, also having good level of correlation with maintainability of class-diagrams [29].

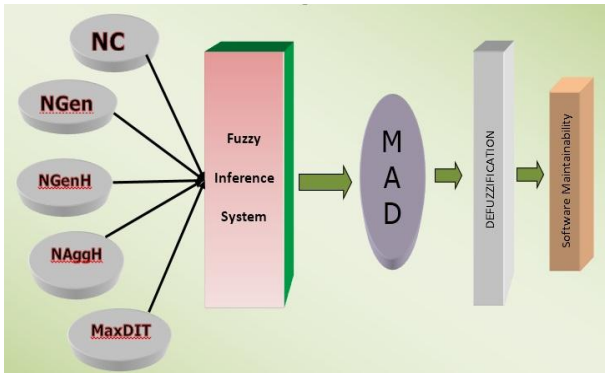

Fig 1: Design Stage Maintainability Prediction Model (DSMPM)

3.2 Selecting Input and Output Variables

To identify metrics those are effectively contributing in the prediction of maintainability, the technique of backward stepwise multiple regression has been used. Consequently, the most significant metrics obtained through backward stepwise multiple regression, has used as input variables for fuzzy inference system. The researcher has performed the Backward Stepwise Regression on the eleven metrics shown in the table 1. This procedure starts with a model, which includes all the metrics as independent variables, and gradually eliminates those, one after another, which does not explain much of the variation in dependent variable, until it ends with an optimal set of independent variables.

From the figure 2, it can be noticed that the backward stepwise multiple regression starts with all the eleven metrics as independent variable in the first model, and gradually eliminates those one after another, which do not explain much of the variation in the dependent variable (mantainability). Therefore, 'NAgg' (corresponds to highest value i.e. 0.775, in the sig. column) has removed from the first model and is not participated in the second model. Similarly, 'MaxHagg' is removed from the second model. This elimination process continues until it ends with an optimal mix of independent variables, in the form of model 7. All the metrics (independent variables) participating in model 7, have significant value less than 0.05. It indicates that these metrics are significant at a confidence level of 95%.

As it can be noticed from the above table that Seventh Model is comprised of the metrics those are statistically significant. The last column of table has values lesser than 0.05. It indicates that each of the five metrics is statistically significant at a significance level of 0.05 (equivalent to a confidence level of 95%). Therefore NC, NGen, NGenH, NAggH and MaxDIT have been selected as input variables for the fuzzy based maintainability prediction model. Apart from that, one output variables MAD (Maintainability After Design) is also taken as the output for the model. MAD represent the level of Maintainability at the end of design stage.

Fig 2: Stepwise Backward Regression

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.*
		B	Std. Error	Beta		
1	(Constant)	0.359	0.539		0.667	0.522
	NC	0.461	0.255	2.274	1.812	0.103
	NA	0.092	0.073	0.878	1.263	0.238
	NM	-0.049	0.030	-0.860	-1.657	0.132
	NAssoc	0.178	0.095	0.437	1.862	0.096
	NAgg	-0.059	0.201	-0.100	-0.295	0.775
	NDep	-0.295	0.277	-0.192	-1.062	0.316
	NGen	-0.218	0.131	-1.043	-1.660	0.131
	NAggH	-0.758	0.543	-0.450	-1.397	0.196
	NGenH	-1.032	0.415	-0.883	-2.486	0.035
	MaxHagg	0.102	0.182	0.096	0.561	0.588
	MaxDIT	0.868	0.325	0.790	2.666	0.026
2	(Constant)	0.487	0.305		1.596	0.142
	NC	0.406	0.165	2.002	2.457	0.034
	NA	0.097	0.068	0.920	1.419	0.186
	NM	-0.049	0.028	-0.869	-1.760	0.109
	NAssoc	0.179	0.091	0.439	1.965	0.078
	NDep	-0.236	0.186	-0.154	-1.273	0.232
	NGen	-0.186	0.072	-0.892	-2.577	0.028
	NAggH	-0.827	0.467	-0.491	-1.771	0.107
	NGenH	-0.947	0.284	-0.810	-3.333	0.008
	MaxHagg	0.102	0.173	0.096	0.587	0.570
	MaxDIT	0.812	0.254	0.740	3.199	0.010
3	(Constant)	0.532	0.286		1.859	0.090
	NC	0.400	0.160	1.971	2.500	0.030
	NA	0.092	0.066	0.880	1.407	0.187
	NM	-0.045	0.026	-0.798	-1.719	0.114
	NAssoc	0.175	0.088	0.429	1.987	0.072
	NDep	-0.290	0.157	-0.189	-1.844	0.092
	NGen	-0.175	0.068	-0.838	-2.590	0.025
	NAggH	-0.636	0.324	-0.378	-1.962	0.076
	NGenH	-0.958	0.275	-0.819	-3.484	0.005
	MaxDIT	0.762	0.231	0.693	3.290	0.007
4	(Constant)	0.590	0.295		2.003	0.068
	NC	0.513	0.144	2.530	3.572	0.004
	NM	-0.017	0.017	-0.294	-0.957	0.357
	NAssoc	0.091	0.067	0.223	1.352	0.201
	NDep	-0.164	0.134	-0.107	-1.220	0.246
	NGen	-0.191	0.069	-0.914	-2.756	0.017
	NAggH	-0.453	0.309	-0.269	-1.467	0.168
	NGenH	-0.914	0.284	-0.782	-3.218	0.007
	MaxDIT	0.574	0.197	0.523	2.917	0.013
5	(Constant)	0.668	0.282		2.369	0.034
	NC	0.434	0.117	2.138	3.711	0.003
	NAssoc	0.074	0.065	0.182	1.144	0.273
	NDep	-0.132	0.130	-0.086	-1.019	0.327
	NGen	-0.157	0.059	-0.750	-2.649	0.020
	NAggH	-0.518	0.300	-0.307	-1.723	0.109
	NGenH	-0.763	0.235	-0.653	-3.241	0.006
	MaxDIT	0.418	0.110	0.380	3.811	0.002
6	(Constant)	0.735	0.275		2.672	0.018
	NC	0.412	0.115	2.033	3.581	0.003
	NAssoc	0.037	0.054	0.091	0.692	0.500
	NGen	-0.150	0.059	-0.719	-2.551	0.023
	NAggH	-0.497	0.300	-0.295	-1.656	0.120
	NGenH	-0.648	0.207	-0.555	-3.132	0.007
	MaxDIT	0.416	0.110	0.379	3.790	0.002
7	(Constant)	0.629	0.224		2.802	0.013
	NC	0.471	0.077	2.322	6.143	0.000
	NGen	-0.173	0.048	-0.830	-3.639	0.002
	NAggH	-0.616	0.241	-0.366	-2.555	0.022
	NGenH	-0.696	0.192	-0.596	-3.634	0.002
	MaxDIT	0.396	0.104	0.360	3.808	0.002

3.3 Develop Fuzzy Profiles and Rule Base

After identifying the required input-output variables for fuzzy inference system, the next step to systematically incorporate expert knowledge into the developing system [30]. As the above Input/output variables are fuzzy in nature, therefore should be expressed by defining appropriate fuzzy profiles. The current work has used the expert opinion to define the membership functions for the design level measure. The shape of the membership function selected in this research is triangular. But in the literature researchers has been using a variety of shapes like polygonal, trapezoidal, triangular, and so on [31]. Because of the convenient in representing the expert knowledge as well as easiness in the computation the paper has selected the triangular membership functions. Fuzzy Profile for the five variables (NC, NGen, NGenH, NAggH, MaxDIT and MAD) are shown in figure 3.

Fig 3: Fuzzy Profiles

Value	NC (0-1)	NGen (0-1)	NAggH (0-1)	NGenH (0-1)	MaxDIT (0-1)	MAD (0-1)
Very low						(0;0;0.3)
Low	(0;0;0.35)	(0;0;0.45)	(0;0;0.4)	(0;0;0.35)	(0;0;0.45)	(0.2;0.35;0.45)
Medium	(0.3;0.6;0.75)	(0.3;0.5;0.65)	(0.25;0.45;0.7)	(0.25;0.5;0.75)	(0.35;0.5;0.75)	(0.4;0.55;0.75)
High	(0.65;0.8;1)	(0.6;1;1)	(0.6;1;1)	(0.7;1;1)	(0.7;1;1)	(0.6;0.75;0.85)
Very high						(0.8;1;1)

The range for the values of all input and output variables has been taken from 0 to 1. As described above that the proposed maintainability model, has five input variables at the design phase, and each has three linguistic states *i.e.*, low (L), medium (M) and high (H). Therefore, total number of rules is 243, and defined as shown in the figure 4.

Fig 4: Rule Base

Rule	NC	NGen	NAggH	NGenH	MaxDIT	MAD
1	LOW	LOW	LOW	LOW	LOW	LOW
2	LOW	LOW	LOW	LOW	MEDIUM	VERY LOW
3	LOW	LOW	LOW	LOW	HIGH	VERY LOW
-	-	-	-	-	-	-
-	-	-	-	-	-	-
99	MEDIUM	LOW	MEDIUM	HIGH	HIGH	VERY LOW
100	MEDIUM	LOW	HIGH	LOW	LOW	MEDIUM
101	MEDIUM	LOW	HIGH	LOW	MEDIUM	LOW
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
189	HIGH	LOW	HIGH	HIGH	HIGH	LOW
190	HIGH	MEDIUM	LOW	LOW	LOW	MEDIUM
191	HIGH	MEDIUM	LOW	LOW	MEDIUM	LOW
-	-	-	-	-	-	-
-	-	-	-	-	-	-
243	HIGH	HIGH	HIGH	HIGH	HIGH	MEDIUM

4. EMPIRICAL VALIDATION

Empirical validation is an important phase of any model development effort. Although the paper comprehensively presented prediction accuracy results of the developed maintainability prediction model (DSMPM), this section presenting the maintainability prediction consistency along with the prediction range for a variety of input values.

Table 2. Maintainability Prediction Range

	NC	NGen	NAggH	NGenH	Max DIT	MAD
Worst Case	0	0.9	0.9	0.9	0	**0.082**
Avg. Case	0.5	0.5	0.5	0.5	0.5	**0.61**
Bast Case	1	0.1	0.1	0.1	1	**0.893**

The above table 2, presents the values of MAD (Maintainability After Design) by the developed model (Design Stage Maintainability Prediction Model (DSMPM)) for the best, average and worst-case input values of participating input metrics (NC, NGen, NAggH, NGenH and MaxDIT). These values of MAD signifying the lower and upper bounds of prediction range at the design phase. It is quiet noticeable from the table that the value of the MAD is 0.082 in the worst case, because the values of corresponding design level measure are at their worst. As long as the range is concern it is also quite satisfactory ranging from 0.082 to 0.893. The model also helps developers to make them aware about the influence of a specific design stage measure on the MAD, through sensitivity analysis. Therefore, once the impact of any input measure on maintainability has been identified, the more cost effectively it can be controlled to improve the overall maintainability and quality of the developing product in its early stage of development.

5. PREDICTIVE ACCURACY

As it is a well-known fact that developing a model is not a big effort until the predictive accuracy of the model has not been verified. This portion of the paper quantitatively ensure that how effectively the Design Stage Maintainability Prediction Model (DSMPM) is predicting the maintainability of the developing software at its design stage. For validating the quantifying ability of DSMPM the researcher has contacted the software developing industries and collected the relevant data for design stage of 12 software projects. In order to statistically validate it, study has calculated the Pearson's correlation coefficient between the actual maintainability values (already known) and the defuzzified values (obtained through MATLAB tool) of Maintainability after Design (MAD).

Table 3. Pearson's correlation coefficient

	Maintainability Predicted	Actual Maintainability
Maintainability Predicted	1	0.921
Actual Maintainability	0.921	1

The above table 3 shows the values of correlation coefficient between the predicted maintainability and the actual maintainability (already known) of the corresponding software project. The well-known software SPSS has used for computing Pearson's correlation coefficient between the predicted and actual maintainability values, and its value is 0.921. This value indicates that there exists a high positive correlation between the already known values (obtained from industry) and the computed values of developed model

(DSMPM). After ensuring it statistical validity the next key requirement is computing the values of various predictive accuracy measures, this is the aspect that should not be ignored. Accurate modeling can assist in scheduling resources and evaluating risk factors. Any improvement in the accuracy of reliability prediction can significantly impact the quality of the developing software application [32]. Predictive accuracy measures that are commonly used includes Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Balanced MMRE (BMMRE), Median Magnitude of Relative Error (MdMRE), Mean Absolute Percentage Error (MAPE) and Prediction at level n (Pred(n)). The values needed to compute these measures are the actual and the predicted maintainability values [33]. The next job is to compute the Magnitude of Relative Error (MRE)s, and subsequently the Mean of these MRE values i.e. MMRE (Mean Magnitude of Relative Error).

$$\text{Sum of } MRE_1, MRE_2, \ldots\ldots MRE_{12} = 1.537$$

$$MMRE = 1.537/12 = 0.12808$$

As it is evident that the value of MMRE falls well below the acceptance threshold value of 0.25. Because, Conte [34] suggests that if MMRE $\leq$ 0.25 then it is considered quite acceptable prediction accuracy of any prediction model. As MMRE suffers from extreme values therefore after computing the MMRE, the researcher will compute next important accuracy measure that is Balanced Mean Magnitude of Relative Error (BMMRE, as it overcomes the limitations of MMRE) and Mean Absolute Percentage Error MAPE as shown below.

$$\text{Sum of } BMRE_1, BMRE_2, \ldots\ldots BMRE_{12} = 2.193$$

$$\text{Balanced MMRE (BMMRE)} = 2.193/12 = 0.18275$$

$$\text{Sum of percentage errors} = 167.478$$

$$\text{Mean Absolute Percentage Error (MAPE)} = 13.9565$$

The computed values of BMMRE and MAPE are quite encouraging and ensures the goodness of the model. One more important predictive measure is the 'Pred(n)', that will tell us the percentage of estimates with an MRE less than or equal to 0.25:

$$Pred(0.25) = 0.89 \ (89\%)$$

The value shown above is suggesting that 89% of the values quantified by the Design Stage Maintainability Prediction Model (DSMPM) have MREs less than or equal to 0.25, this is also a very good result that further strengthen the predictive quality of the model.

Table 4. Summary of Predictive Accuracy Measures

S.No.	Name of Measure	Value
1	MMRE	0.12808
2	BMMRE	0.18275
3	MAPE	13.9565
4	PRED(n)	0.89 (89%)

The above table 4 summarized all the computed results. After reviewing the values, it can be claimed that the fuzzy based maintainability model developed in this paper is quantifying the maintainability of the object-oriented developing application quite well after the design phase is completed or prior to the start of the most crucial coding phase in the development life cycle.

6. CONCLUSION AND FUTURE WORK

Researchers are continuously doing their efforts to quantify various quality attributes of software applications at different stages of software life cycle. This paper has also performed a similar contribution in this domain by developing Fuzzy based Maintainability Estimation Model, in order to quantify the software maintainability before the coding phase starts. the paper quantifies the maintainability of class-diagram using design level measure like size and structural complexity metrics of class diagrams. The result produced in the paper will definitely assist application developers to review the design stage documents and perform needed corrections to reduce future maintenance cost that may pop-up in the latter stages of SDLC. The incorporation of fuzzy inference system in this model gives it an edge over existing similar models in the domain of maintainability estimation. Further it is empirically validated along with predictive accuracy measures. Overall looking at the results it can be concluded that the model developed will help the concerned stakeholders in the development domain where maintainability of software has been a big concern for a long period. In future also better and generalized models can be produced by conducting a larger scale study with a variety of industrial based projects across diverse domains.

7. REFERENCES

[1] Rizvi, S. W. A., and Khan, R. A. 2010. Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD). Journal of Computing, 2(4), 26-32.

[2] Mamdani, E. H. 1977. Applications of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Transaction on Computers, 26(12), 1182–1191.

[3] Li W., and Henry, S. 1993. Object-Oriented Metrics that Predict Maintainability. Journal of Systems and Software, 23(2), 111 – 122.

[4] Schneidewind, N. F. 1992. Methodology for Validating Software Metrics. IEEE Trans. on Software Engineering, 18(5), 410 - 422

[5] Rizvi, S. W. A., and Khan, R. A. 2009. A Critical Review on Software Maintainability Models. Proceedings of the Conference on Cutting Edge Computer and Electronics Technologies, 144-148.

[6] Bansiya, J., and Devis, C. 1997. Automated Metrics for Object-Oriented Development. Dr. Dobb's Journal, 272(12), 42-48.

[7] Bansiya, J., and Devis, C. 2002. A Hierarchical Model for Object-Oriented Design Quality Assessment. IEEE Transactions on Software Engineering, 28(1), 4-17.

[8] Rizvi, S.W.A. and Khan, R.A. 2013. Improving Software Requirements through Formal Methods. International Journal of Information and Computation Technology, 3(11), 1217-1223.

[9] Antonellis, P., Antoniou, D., Kanellopoulos, Y., Makris, C., Theodoridis, E., Tjortjis, C., and Tsirakis, N. 2007. A Data Mining Methodology for Evaluating Maintainability According to ISO/IEC-9126 Software Engineering Product Quality Standard. Proc. 11th IEEE Conference on Software Maintenance and Reengineering (CSMR2007), 21 – 23 Mar. 2007, Amsterdam, Netherlands.

[10] Oman, P. W. and Hagemeister, J. R. 1994. Construction and Testing of Polynomials Predicting Software

Maintainability. Journal of Systems and Software, 24(3), 251 – 266.

[11] Welker, K. D. and Oman, P. W. 1995. Software Maintainability Metrics Models in Practice. Journal of Defense Software Engineering, 8(11), 19 – 23.

[12] Hayes, J. H., Patel, S. C. and Zhao, L. 2004. A Metrics-Based Software Maintenance Effort Model. Proc. 8th European Conference on Software Maintenance and Reengineering (CSMR'04) IEEE Socity, 24 – 26 Mar. 2004, 254 – 258.

[13] Polo, M., Piattini, M., and Ruiz, F. 2001. Using Code Metrics to Predict Maintenance of Legacy Programs: A Case Study. Proc. of International Conference on Software Maintenance, IEEE Computer Society, Florence Italy, ICSM 2001, 202-208.

[14] Hayes, J. H., and Zhao, L. 2005. Maintainability Prediction: A Regression Analysis of Measures of Evolving Systems. Proc. 21st IEEE International Conference on Software Maintenance, 26 - 29 Sept. 2005, 601 - 604.

[15] Muthanna, S., Kontogiannis, K., Ponnambalam, K., and Stacey, B. 2000. A Maintainability Model for Industrial Software Systems Using Design Level Metrics. Proc. 7th Working Conference on Reverse Engineering (WCRE'00), 23 - 25 Nov., 2000, Brisbane, Australia, 248 – 256.

[16] Genero, M., Manso, E., and Cantone, G. 2003. Building UML Class Diagram Maintainability Prediction Models Based on Early Metrics. Proc. 9th International Symposium on Software Metrics (METRICS'03), 3 - 5 Sept., 2003, 263 – 275.

[17] Kiewkanya, M., Jindasawat, N., and Muenchaisri, P. 2004. A Methodology for Constructing Maintainability Model of Object-Oriented Design. Proc. 4th International Conference on Quality Software, IEEE Computer Society 8 - 9 Sept., 2004, 206 - 213.

[18] Rizvi, S.W.A., Singh, V.K. and Khan, R.A. 2016. Software Reliability Prediction using Fuzzy Inference System: Early Stage Perspective, International Journal of Computer Applications, 145(10), 16-23.

[19] Zadeh, L.A. 1989. Knowledge representation in fuzzy logic. IEEE Transactions on Knowledge and Data Engineering, 1(1), 89–100.

[20] Hitz, M., and Montazeria, B. 1996. Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective. IEEE Transactions on Software Engineering, 22(4), 267 – 271.

[21] Rizvi, S.W.A., Singh, V. K., and Khan, R. A. 2016. Fuzzy Logic based Software Reliability Quantification Framework: Early Stage Perspective (^{FL}SRQF), Elsevier Procedia-Computer Science, 89, 359-368

[22] Abreu B. F., and Carapuca, R. 1993. Candidate Metrics for Object-Oriented Software within a Taxonomy Framework. Journal of Systems and Software, Elsevier-Science, 23(1), 87 – 96.

[23] Genero, M., Manso, E., Visaggio, A., and Piattini, M. 2007. Building Measure-Based Prediction Models for UML Class Diagram Maintainability. Journal of Empirical Software Engineering, 12(5), 517 – 549.

[24] Rizvi, S. W. A., Singh, V. K., and Khan, R. A. 2016. The State of the Art in Software Reliability Prediction: Software Metrics and Fuzzy Logic Perspective. Advances in Intelligent Systems and Computing, Springer, 433, 629-637.

[25] Elish, M. O., and Elish, K. O. 2009. Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study. Proc. of European Conference on Software Maintenance and Reengineering (CSMR'09), 24 - 27 Mar., 2009, 69 - 78.

[26] Rizvi, S.W.A., Khan, R.A., and Singh, V.K. (2017). Early Stage Software Reliability Modeling using Requirements and Object-Oriented Design Metrics: Fuzzy Logic Perspective. *International Journal of Computer Applications*, 162(2), 44-59.

[27] Bruntink M., and Deursen, A. 2004. Predicting Class Testability using Object-Oriented Metrics. Proc. 4th IEEE International Workshop on Source Code Analysis and Manipulation SCAM'04, 15 - 16 Sept., 2004, 136 – 145.

[28] Olague, H.M., Etzkorn, L. H., Messimer, S.L., and Delugach, H.S. 2008. An Empirical Validation of Object-Oriented Class Complexity Metrics and their Ability to Predict Error-prone Classes in Highly Iterative, or Agile Software: a Case Study," Journal of Software Maintenance, vol. 20(3), 171 – 197.

[29] S. W. A. Rizvi, V. K. Singh, R. A. Khan: "Revisiting Software Reliability Engineering with Fuzzy Techniques", IEEE Xplore, 10th INDIACom, organized by BVICAM, New Delhi, March, 2016. ISBN:978-93-80544-19-9, ISSN:0973- 7529, pp. 1037-1042.

[30] Ross, T. J. 2010. Fuzzy Logic with Engineering Applications. 3rd Edition, John Wiley and sons.

[31] Dromey, R.G. 1995. A Model for Software Product Quality. IEEE Transactions on Software Engineering, 21(2), 146-162

[32] S. W. A. Rizvi, V. K. Singh, R. A. Khan,: "Application of Fuzzy Logic in Early Stage Software Reliability Prediction", International Journal of Information Processing, Vol. 10, Issue 3, November 2016, pp. 61-77.

[33] Walkerden, F., and Jeffery, R. 1999. Analogy, Regression and Other Methods for Estimating Effort and Software Quality Attributes. Proceeding of European Conference Optimizing Software Development and Maintenance, 37-46

[34] Conte, S. D., Dunsmore, H. F., and Shen, V. Y. 1986. Software Engineering Metrics and Models. ISBN: 0805321624, Benjamin Cummings Publishing Co., Inc., Redwood city, CA, USA.