Software Reliability Prediction using Fuzzy Inference System: Early Stage Perspective

Syed Wajahat A. Rizvi
Department of Computer Science, BBD University, Lucknow

Raees Ahmad Khan
Department of Information Technology, Dr. Bhimrao Ambedkar University, Lucknow

Vivek Kumar Singh
Department of Information Technology, BBDNITM, Lucknow

ABSTRACT
The paper presents a reliability prediction model that predicts the reliability of the developing software using fuzzy inference system. The focus of the study is on the reliability prediction prior to the coding phase so that the developers use this information for optimally performing resource planning and quality assessment of the software under development. Requirements and object-oriented design level product measures have participated for early reliability prediction. The paper has also utilized the strengths of fuzzy logic to deal with the uncertainties and vagueness involved in the early stage measures. The model has also been statistically validated through the data set obtained through twenty real software projects. The values of the Pearson's correlation coefficient along with the predictive accuracy measures are quite encouraging, and support that the developed model is a better and improved reliability prediction model.

Keywords
Software Reliability, Early Stage Prediction, Fuzzy Logic, Software Defects, Software Metrics, Software Reliability Model.

1. INTRODUCTION
Literature has defined the software reliability as the probability that a given software system has operated for some duration without being fail on the machine for which it was designed, given that it is used within design limits [1, 2]. As in the current scenario internet has been reducing the distances between geographical boundaries all over the globe and every sector of society whether it is transportation, military, telecommunication, aircrafts, shopping, home appliances, entertainment, education, e-governance, and so on, are highly influenced by computer software [3]. Such dependence as well as the trust on software is forcing the industry to provide more user friendly software within limited development time and resources. Consequently this pressure, up to some extent, has been increasing the probability of committing errors and making the software less reliable [4].

The literature exposes several unpleasant happenings related to the failures of software in a variety of domains [2, 4, 5, 6, 7] due to that many people lost their lives. Lee L. in 1992 described in his book that many patients lost their lives in 1985 and 1986 due to software error in the Therac-25 radiation therapy machine [4]. One incident was reported in 1993 by South West Thames regional health authority on the computer aided dispatch system that was broken after its installation, consequently London ambulance service was unable to handle more than 5000 daily request for carrying patients in emergency situations [5]. Due to incompatible software responses to the pilots, aviation industry had also faced lots of airline crashes and abnormal flight conditions [6]. On January 28, 1986, space shuttle challenger broke apart 73 seconds into its flight, leading to the death of its seven crew members [2]. Due to the change in the three lines of code in a single program in 1991, the telephone system collapsed in California and eastern parts [5]. On February 25, 1991, during the Iraq war, the chopping error that missed the 0.000000095 second in precision in every 10th of a second made the patriot missile fail to intercept a scud missile [6]. One of the major causes behind all these unfortunate events is the presence of unreliable software.

As reliability has become a critical factor in software systems, its prediction is of great importance. An accurate estimate of reliability can be obtained through software reliability models only in the later phase of development. 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 [8, 9, 10]. In this paper, the author has proposed reliability modeling that has used the fuzzy inference system to predict the reliability of the developing software before the coding start. The rest of the paper is organized as follows; section 2 describes the state-of-art on reliability prediction models. Section 3 presents the proposed reliability prediction model, while the proposed model has implemented in section 4. Statistical validation and the predictive accuracy of the model are presented in Section 5, while the paper concludes in section 6.

2. RELATED WORK
During last twenty to twenty five years a significant number of software reliability prediction or estimation models have been proposed in the literature. These models have quantified the reliability during various stages of software development, most of them had predicted the reliability in later stages, while some had advocated its prediction in the early stages like requirements or design. According to Pham [11], reliability models can be classified in two major categories as deterministic and probabilistic models. Probabilistic models can be further categorized in four classes as (a) Failure rate model, (b) Fault count model, (c) Error seeding model, and (d) Software Reliability Growth Models (SRGMs). Failure rate models focus on the failure rate, rather on the count of failures. While the failures count models have interest in the number of failures during a specified duration rather than the duration gap between failures [12]. Failure seeding models estimate the number of faults on the basis of known number of seeded defects in the software [13], and a software reliability growth model is applicable in the last phases of software development and quantifies the software reliability in terms of estimated residual defects. They are called SRGMs, because with the progress of software testing residual defects decreases, while the reliability of the software grows [12]. In the absence of fault related data SRGMs are not applicable. In this situation researchers have an option to rely on fault data obtained from the requirements or design

stages. In the absence of failure data at the early stages of development life cycle reliability can be predicted in the light of those software metrics that are reliability relevant, maturity level of the developer and expert opinions. It can be noticed from some of the studies that software metrics along with the process maturity play a prominent role in early defect prediction, when failure data is unavailable [14].

The concept of fuzzy sets was introduced by Zadeh [15] to represent vagueness in linguistics as a mathematical way. It can be considered as generalization of classical set theory. Many researchers had contributed [12, 16, 17, 18, 19] in the area of reliability prediction using fuzzy logic. Yadav et al. [19] present a model that predicts the number of residual faults before testing stage. The study had used the software metrics along with fuzzy logic to predict the remaining defects in the software that are expected during testing or when the software would be actually used. A new approach was introduced by Aljahdali et al. [20] using Fuzzy Logic and Normalized Root of Mean of the Square of Error (NRMSE) for software reliability prediction. This design of the fuzzy model was based on the Takagi-Sugeno (TS) fuzzy model. Another similar study [21] introduced a methodology that starts with the analysis of the UML model of software architecture followed by the bayesian framework for reliability prediction. Three different types of UML diagrams Use Case, Sequence and Deployment diagrams were utilized. While, another promising study [22] introduced two new estimation methods to overcome the limitations of existing statistical methods in estimating the defect content after a review. In [14, 23] authors proposed fuzzy logic based defect prediction models. The relevant metrics are judged as per linguistic terms and fuzzy techniques were applied in order to develop the model. The predicted defects of twenty software projects, by the proposed model, were found very near to the actual defects found in the testing phase.

Fuzzy Logic techniques are emerging as robust optimization techniques that can solve highly complex, nonlinear, correlated and discontinuous problems [12]. As most of the early stage software metrics are not very comprehensible and involve high and complex dependency among them. That's why fuzzy logic inference systems have found usefulness in capturing and processing subjective information in terms of software metrics in the early phase of software development. On the basis of above paragraphs it is evident that the Fuzzy Logic has proved its usefulness in capturing and processing subjective information in the early stages of software development [24]. The key issue is how it is applied in making the software product more reliable.

3. PROPOSED RELIABILITY PREDICTION MODEL

After recognizing the criticality of requirements and design stage for early prediction of software reliability, it is needed to consider the suitable and appropriate measures form these stages. Therefore this study has focused on the identification of reliability-relevant software metrics or measure for early reliability prediction. For this, a comprehensive model as depicted in figure 1 has been proposed.

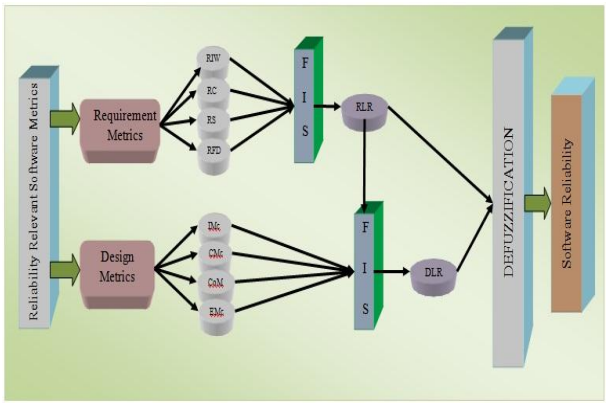

Fig 1: Early Stage Reliability Prediction Model

The model integrates requirements and design metrics as input to the fuzzy inference system to predict the reliability of the developing software up to its design stage before the coding starts. The model is referred as Early Stage Reliability Prediction Model (ESRPM) and is based on the assumption that the reliability and quality of a software system are adversely affected by the weaknesses of requirements and design constructs. Therefore the model focuses on these two, most significant, early phases of SDLC.

4. MODEL IMPLEMENTATION

The proposed model is predicting the reliability using Fuzzy Inference System (FIS), therefore implementation has been performed through fuzzy logic toolbox of MATLAB. The basic steps of the model implementation are identification of requirements and design metrics as input/output variables, development of fuzzy profile and membership functions of the identified variables and development of fuzzy rule base. These steps are discussed in the following sub-sections.

4.1 Identify Requirements Level Metrics

Most of the existing reliability or defect prediction models has considered a significant number of software metrics such as traditional software metrics, object oriented metrics and process metrics. However, utilizing all metrics for predicting software reliability have various drawbacks like computational complexity, high processing cost, larger time complexity etc. [25]. Therefore appropriate selection of metrics could improve the prediction accuracy. However, it is essential to consider the metrics which are most important from reliability point of view and the researcher has gathered following twelve software requirements metrics from various available sources [26, 27, 28, 29].

RS (Requirements Stability), PM (Process Maturity), RSDR (Regularity of Specification and Documentation Reviews), RIW (Review Inspection and Walkthrough), RFD (Requirement Defect Density), RCR (Requirement Change Request), Scale of New Functionality Implemented, ERT (Experience of Requirement Team), RC (Complexity of New Functionality), QDI (Quality of Documentation Inspected), DSM (Development Staff Motivation), and RM (Requirements Management).

4.2 Identify Design Level Metrics

As the proposed reliability model concentrates on object-oriented paradigm in its design phase. Therefore the researcher has gathered following fourteen object-oriented design metrics from various available sources [30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. LCOM (Lack of Cohesion in Methods), MPC (Message Pass Coupling), IMc (Inheritance

Metric Complexity Perspective), DIT (Depth of Inheritance), NOC (Number of Children), EMc (Encapsulation Metric Complexity Perspective), CMc (Coupling Metric Complexity Perspective), WMC (Weighted Method per Class), CBO (Coupling Between Objects), Response for a Class (RFC), CoMc (Cohesion Metric Complexity Perspective), DAC (Data Abstraction Coupling), AHF (Attribute Hiding Factor) and AIF (Attribute Inheritance Factor).

4.3 Select Input and Output Variables

After analyzing the twenty six identified requirements and design metrics, eight metrics have been shortlisted and out of these four (RS, RIW, RC, RFD) have been selected for the requirements phase and rest four (EMc, CoMc, CMc, and IMc) belongs to the design phase. These metrics are considered as input variables for the fuzzy based reliability prediction model. Apart from that, two output variables RLR and DLR are also taken as the output for the model. RLR and DLR represent the level of reliability at the end of requirements and design phases, respectively.

4.4 Develop Fuzzy Profiles and Rule Base

This is the primary step to systematically incorporate expert knowledge into the developing system [40]. As the selected Input/output variables are fuzzy in nature and therefore should be characterized through membership functions. In this study, membership functions of all the input and output variables are defined with the help of domain experts. Membership function can have a variety of shapes like polygonal, trapezoidal, triangular, and so on [41]. This study has used triangular membership functions, for fuzzy profile development of identified input/output variables, as its shape provides a convenient representation of expert knowledge and it also simplifies the process of computation. Membership functions for the four variables (RIW, Inheritance, Cohesion and DLR) are shown in figure 2, 3, 4 and 5 for visualization purpose. The range for the values of all input and output variables has been taken from 0 to 1.

As described above that the proposed reliability model, has four input variables at the requirements phase, and each has three linguistic states *i.e.*, low (L), medium (M) and high (H). Therefore, total number of rules is 81. Similarly in design phase the number of input variables are five, four has three linguistic states (*i.e.*, low (L), medium (M) and high (H)), while one input variable has five states (*i.e.*, Very Low (VL), Low (L), Medium (M) High (H) and Very High (VH)). Considering all the selected Input/output variables simultaneously may results into a large number of rules. Therefore, to reduce the number of rules the researcher has developed two set of rules corresponding to the requirements and design phase.

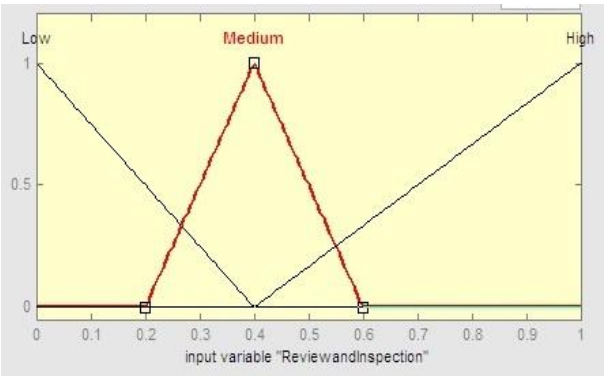

Fig 2: Fuzzy Profile of RIW

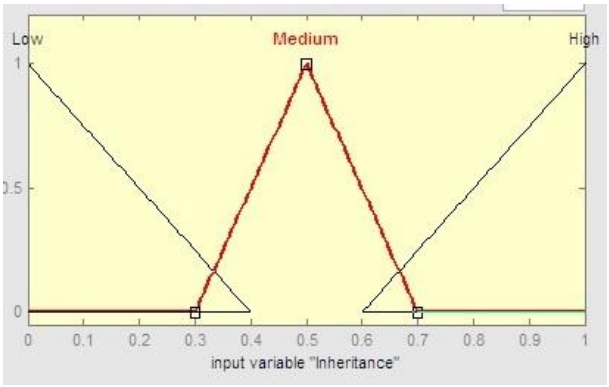

Fig 3: Fuzzy Profile of Inheritance

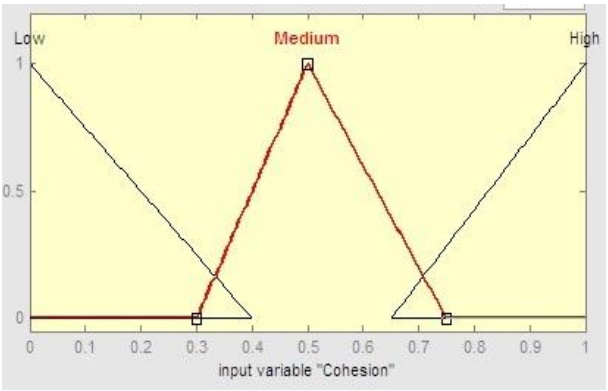

Fig 4: Fuzzy Profile of Cohesion

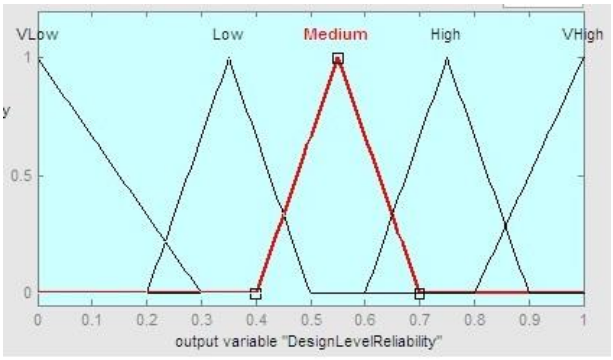

Fig 5: Fuzzy Profile of DLR

4.5 Verifying the Prediction Range

Although the prediction accuracy of the developed reliability prediction model has been computed and presented in the fifth section, even though to analyze the reliability prediction consistency and influence of different involved metrics on reliability prediction, evaluating the model's prediction range seem quite reasonable.

Table 1. Reliability Prediction at Requirements Stage

	RS	RIW	RC	RFD	RLR
Best Case	1	1	0	0	0.953
Average Case	0.5	0.5	0.5	0.5	0.665
Worst Case	0	0	1	1	0.113

Table 2. Reliability Prediction at Design Stage

	RLR	EMc	CoMc	IMc	CMc	DLR
Worst Case	0	0.1	0.1	0.9	0.9	0.096
Avg. Case	0.5	0.5	0.5	0.5	0.5	0.55
Best Case	1	0.9	0.9	0.1	0.1	0.937

Table 1 and 2, and the figures 6, 7, 8, 9, 10 and11 presents the values of RLR (Requirements Level Reliability) and DLR (Design Level Reliability) by the developed model (ESRPM) for the best, average and worst-case input values of different input metrics. These values of RLR and DLR signifying the lower and upper bounds of prediction range at the requirements and design phase respectively. It can be easily noticed that the value of the RLR is 0.113 in the worst case, because the values of corresponding requirements level measure are at their worst.

The RLR at the end of requirements phase range from 0.113 to 0.953, while the range for DLR is 0.096 to 0.937, which is quiet satisfactory. The model also helps to determine the influence of a particular software metrics on the requirement or design level reliability. Once the impact of the particular software measure on reliability has been identified, the better and more cost effectively it can be controlled to improve the overall reliability and quality of the product.

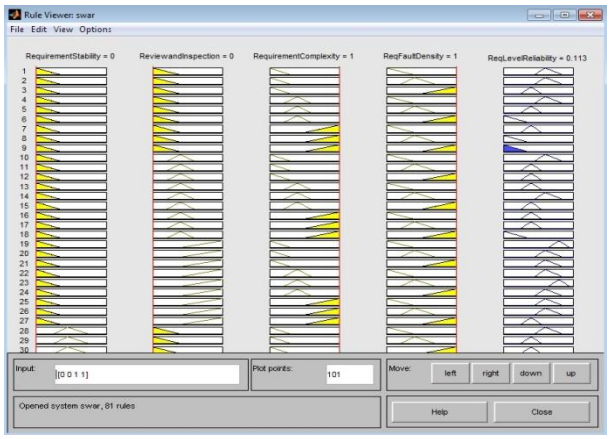

Fig 6: Worst Case at Requirements Phase

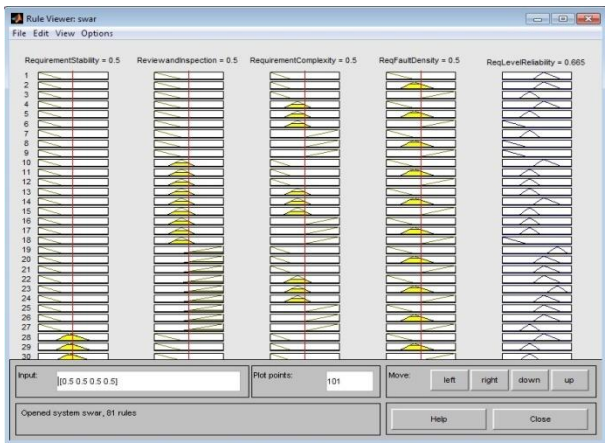

Fig 7: Average Case at Requirements Phase

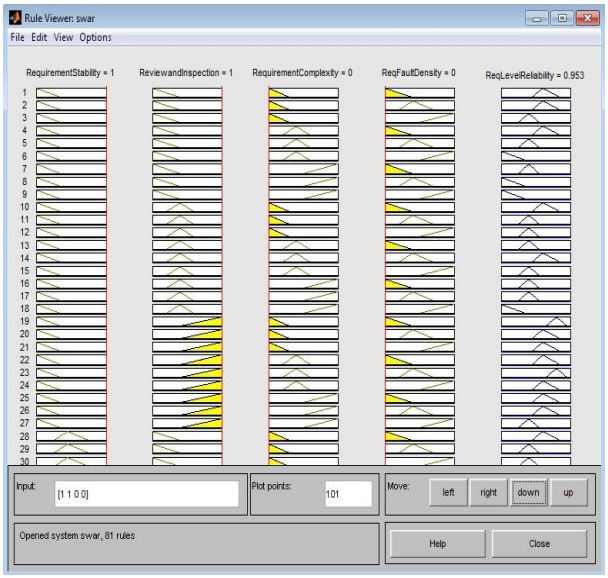

Fig 8: Best Case at Requirements Phase

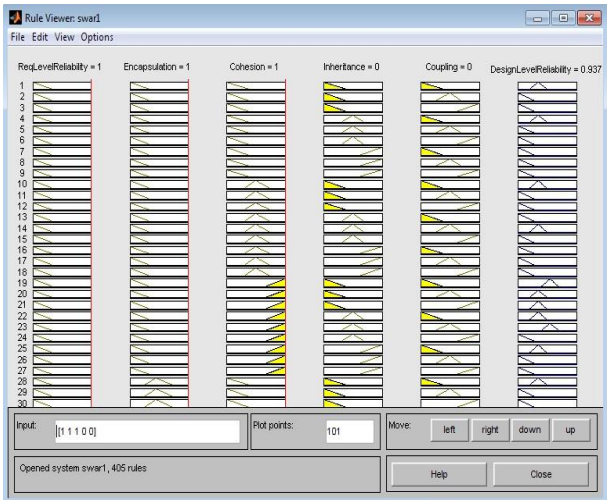

Fig 9: Best Case at Design Phase

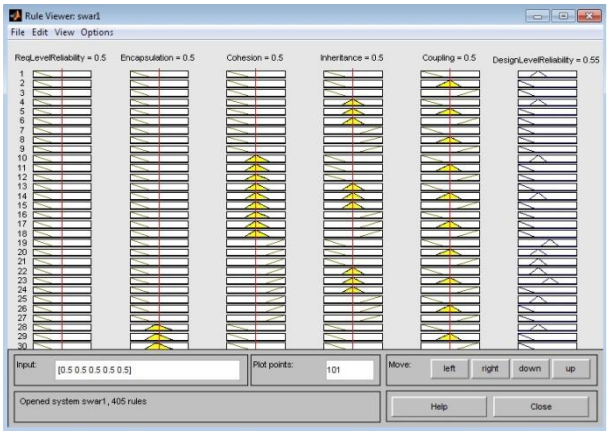

Fig 10: Average Case at Design Phase

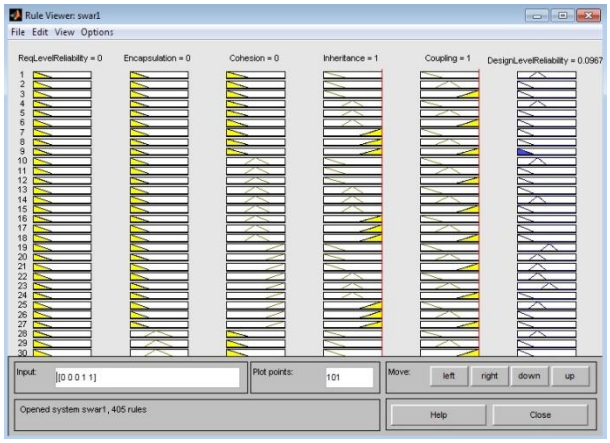

Fig 11: Worst Case at Design Phase

Apart from ensuring the reliability prediction range from the above diagrams, it is equally important to look at the following surface diagrams, demonstrating the impact of constituent metrics on the RLR as well as on the DLR.

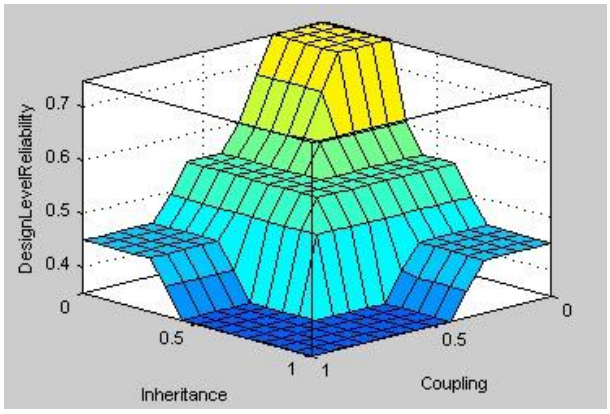

Fig 12: Surface Diagram of DLR Vs. Inheritance and Coupling

Looking carefully at the above figure 12, that shows the impact of inheritance and coupling on the design level reliability, while the values of other variable are constant, it can be easily inferred that for any constant value of EM (encapsulation), CoM (cohesion) and RLR (requirement level reliability), both the metrics CM (coupling) and IM (inheritance) has a negative impact on the DLR. When the value of CM and IM increases, it forces the DLR to decrease. In other words, Reliability of the software will decrease as the coupling as well as inheritance increases in the object oriented design.

Similarly, looking at the above figure 13, it can be easily noticed that for any constant value of IM, CM and RLR, both the metric EM (encapsulation) and CoM (cohesion) has a positive impact on the DLR (design level reliability). When the values of EM and CoM increase, it forces the DLR to increase also. In other words, the developing software will be more reliable if the OOD has higher level of Cohesion as well as Encapsulation. Like the above two diagrams, the following figure 14, represents the influence of Requirements Complexity (RC) and Requirements Fault Density (RFD) on Requirements Level Reliability (RLR). The diagram reflects that for any constant value of RS and RIW, the metric values of RC and RFD have a negative impact on the RLR. When the value of RC and RFD increases, it forces the RLR to decrease.

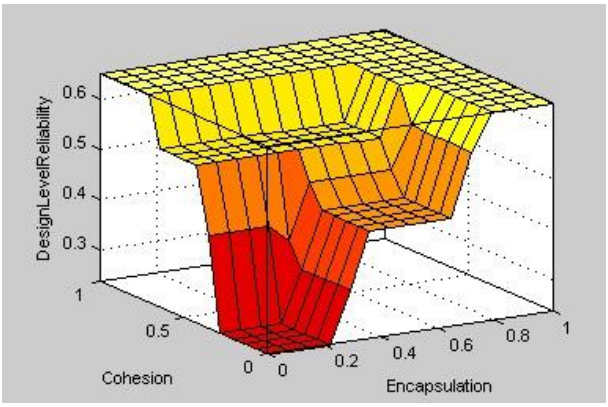

Fig 13: Surface Diagram of DLR Vs. Cohesion and Encapsulation

In other words, reliability of the developing software at the requirements as well as design stage will decrease as the functionalities get more complicated in the SRS along with the density of faults in a SRS document.

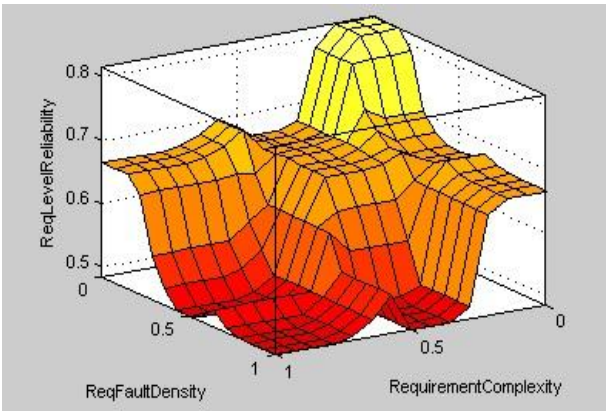

Fig 14: Surface Diagram of RLR Vs. RFD and RC

5. STATISTICAL VALIDATION AND PREDICTIVE ACCURACY

This section assesses how effectively the reliability model is able to predict the reliability of the developing software at its design stage. In order to ensure or validate the quantifying ability of the developed reliability model the researcher has contacted the well established and reputed software developing organizations and subsequently collected the relevant data for requirements and design stage of 20 software projects, those had already been implemented and currently in operation. In order to statistically validate the model, researcher has calculated the Pearson's correlation coefficient between the actual reliability values (already known) and the defuzzified (predicted) values of Design Level Reliability (DLR).

The defuzzified (predicted) values of Design Level Reliability (for 20 software projects) have been computed using the fuzzy toolbox of MATLAB. These calculated values, along with the corresponding actual reliability values can be seen in table 3. Now to ensure the quantifying ability of the model Pearson's correlation coefficient has been computed, between predicted and actual reliability. Table 4 show the correlation values between the predicted reliability and the actual reliability (already known) of the corresponding software project. The correlation has been computed through SPSS, and its value is (0.936) as shown figure 15. It is evident from

the value, that there exists a high positive correlation between the reliability predicted by the ESRPM and the already known values of reliability. Therefore, it can be concluded that the proposed model is quantifying reliability quiet efficiently.

Table 3. Predicted and Actual Reliability Values

S.No.	Project Number	Reliability Predicted by the Proposed Model	Actual Reliability
1	P1	0.832	0.9
2	P2	0.721	0.9
3	P3	0.912	0.9
4	P4	0.600	0.75
5	P5	0.750	0.75
6	P6	0.587	0.55
7	P7	0.750	0.75
8	P8	0.550	0.55
9	P9	0.586	0.55
10	P10	0.750	0.75
11	P11	0.629	0.55
12	P12	0.614	0.55
13	P13	0.565	0.55
14	P14	0.752	0.75
15	P15	0.761	0.55
16	P16	0.320	0.35
17	P17	0.330	0.35
18	P18	0.350	0.35
19	P19	0.131	0.15
20	P20	0.210	0.15

Table 4. Pearson Correlation Coefficient

	Reliability Predicted	Actual Reliability
Reliability Predicted	1	0.936
Actual Reliability	0.936	1

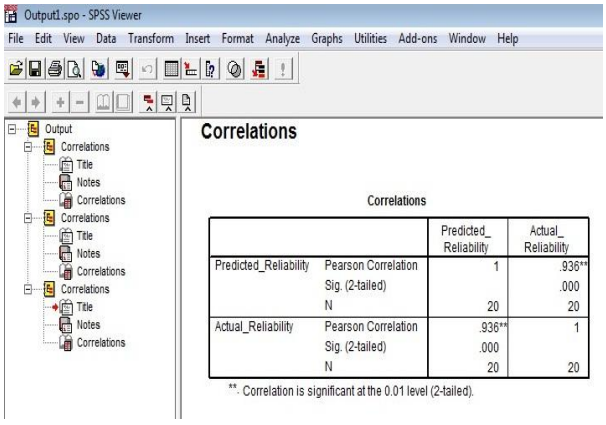

Fig 15: SPSS Output

Along with proper validation, ensuring the predictive accuracy of a model is one of the important aspects that cannot be ignored. Accurate modeling can assists 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 [42]. It is evident from the literature that researchers have been using various measures to ensure the predictive accuracy of the developed models. The most popular measures include 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)). Most of these measures can be calculated through the two terms, the actual and the predicted values [43].

In order to compute these predictive accuracy measures the values of table 3 are used. The next task 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).

$$\text{Sum of } MRE_1, MRE_2, \ldots\ldots MRE_{20} = 1.964$$

$$MMRE = 1.964/20 = 0.09818$$

The value of MMRE is quite encouraging and falls well below the acceptance threshold value of 0.25. Because, Conte [43] suggests that if MMRE $\leq$ 0.25 then it is considered quite acceptable prediction accuracy of any prediction model. After computing the MMRE, next important accuracy measures to be computed are Balanced Mean Magnitude of Relative Error (BMMRE, as it overcomes the limitations of MMRE) and Mean Absolute Percentage Error MAPE as shown below.

$$\text{Sum of } BMRE_1, BMRE_2, \ldots\ldots BMRE_{20} = 2.099$$

$$\text{Balanced MMRE (BMMRE)} = 0.104951$$

$$\text{Sum of percentage errors} = 196.360$$

$$\text{Mean Absolute Percentage Error (MAPE)} = 9.818023$$

Like MMRE the values of BMMRE and MAPE are also comes out very promising, and reemphasizing that the model ESRPM has a higher predictive accuracy. After computing the MMRE and BMMRE, the quartiles of MRE distribution (i.e. MdMRE, P_{25} & P_{75}) are also calculated. In order to compute MdMRE (Median Magnitude of Relative Error), P_{25} (Ist Quartile) & P_{75} (IIIrd Quartile), the values of MREs are arranged in ascending order.

$$\text{Median Magnitude of Relative Error (MdMRE)} = 0.066$$

$$P_{25} \text{ (Ist Quartile)} = 0.0000$$

$$P_{75} \text{ (IIIrd Quartile)} = 0.135152$$

The values of MdMRE P_{25} and P_{75} are also as good as other values. To know the percentage of estimates with an MRE less than or equal to 0.25, the study also computed the Pred(0.25) as follows:

$$Pred(0.25) = 0.90 \ (90\%)$$

The above value of Pred(0.25) indicating that the 90% of the predicted values by the reliability model (ESRPM) have MREs less than or equal to 0.25, that is quiet encouraging. All the above results are also summarized in the following table 5.

Table 5. Summary of Predictive Accuracy Values

S.No.	Name of Measure	Value
1	MMRE	0.09818
2	BMMRE	0.10495
3	MdMRE	0.0660
4	MAPE	9.8180
5	PRED(n)	0.90 (90%)

Looking at the values of various accuracy measures, it is evident that the prediction ability of the reliability model ESRPM is quiet accurate. Therefore it can be concluded that the model can be used to accurately predict the design level

reliability for any Object-Oriented software before its coding starts.

6. CONCLUSION

Although in a period of last 25 years, lot of models for quantifying software reliability has been proposed in the literature by various researchers. But even though, reliability modeling is still attracting more researchers to do some contribution in this direction. With this spirit the researcher has proposed and implemented a fuzzy based software reliability prediction model. The model has used eight product based measures from the requirements and design stage. The main factor that provides this model an edge over other existing model is its approach of prediction. The study has also statistically validated the developed model and computed the various predictive accuracy measures to ensure its prediction efficiency. The results obtained are quite encouraging and it can be concluded that the developed reliability prediction model is a better model.

7. REFERENCES

[1] Reibman, A. L., and Veeraraghawan, M. 1991. Reliability Modeling: an overview for system design. IEEE Computer Sociaty, 24(4), 49-57.

[2] Lions, J. L. 2010. ARIANE 5 Flight - 501 Failures Report.

[3] Lyu, M. R. 1996. Handbook of Software Reliability Engineering. IEEE Computer Society Press, Los Alamitos, California.

[4] Dalal, S. R., Lyu, M. R., and Mallows, C. L. 2014. Software Reliability. John Wiley & Sons.

[5] Khan, R. A., Mustafa, K., and Ahson, S. I. 2004. Operation Profile-a key Factor for Reliability Estimation. University Press, Gautam Das and V. P. Gulati (Eds), CIT, 347-354.

[6] Ogheneovo, E. E. 2014. Software Dysfunction: Why Do Software Fail?. Journal of Computer and Communications, 2, 25-35.

[7] Rizvi, S. W. A., Singh, V. K., and Khan, R. A. 2016. Revisiting Software Reliability Engineering with Fuzzy Techniques. In:(IndiaCom–2016) Proc. of the 3rd IEEE Int. Conf. on Computing for Sustainable Global Development. Published by IEEExplore, 16-18 March, 2016. New Delhi, India.

[8] Yadav, H. B., and Yadav, D. K. 2014. Early Software Reliability Analysis using Reliability Relevant Software Metrics. International Journal of System Assurance Engineering and Management, pp.1-12.

[9] 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.

[10] 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.

[11] Pham, H. 2006. System Software Reliability. London: Reliability Engineering Series, Springer.

[12] Pandey, A. K., and Goyal, N. K. 2013. Early Software Reliability Prediction. Springer, India.

[13] Goel, A. L. 1985. Software Reliability Models: Assumptions, Limitations, and Applicability. IEEE Transaction on Software Engineering, 11(12), 1411-1423.

[14] Yadav, H. B., and Yadav, D. K. 2014. Early Software Reliability Analysis using Reliability Relevant Software Metrics. International Journal of System Assurance Engineering and Management,1-12.

[15] Zadeh, L. 1965. Fuzzy Sets. Information and Control, 8, 338-353.

[16] Khalsa, S. K. 2009. A Fuzzified Approach for the Prediction of Fault Proneness and Defect Density. In: Proceeding of World Congress on Eng., 1, 218-223.

[17] Yadav, O. P, Singh, N., Chinnam, R. B., and Goel, P. S. 2003. A Fuzzy Logic based approach to Reliability Improvement during Product Development. Reliability Engineering and System Safety, 80, 63-74.

[18] Yuan, D., and Zhang, C. 2011. Evaluation Strategy for Software Reliability Based on ANFIS. In: (ICECC-11) Proceedings of the IEEE International Conference on Electronics and Communications and Control, 3738-3741.

[19] Yadav, D. K., Charurvedi, S. K., and Mishra, R. B. 2012. Early Software Defects Prediction using Fuzzy Logic. International Journal of Performability Engineering, 8(4), 399-408.

[20] Aljahdali, S. 2011. Development of Software Reliability Growth Models for Industrial Applications Using Fuzzy Logic. Journal of Computer Science, 7(10),1574-1580.

[21] Cortellesa, V., Singh, H., and Cukic, B. 2002. Early Reliability Assessment of UML Based Software Models. In:Proceedings of the 3rd International Workshop on Software and Performance, 302–309.

[22] Wholin, C., and Runeson, P. 1998. Defect Content Estimations from Review Data. In:Proceedings of 20th International Conference on Software Engineering, 400-409.

[23] Yadav, H. B., and Yadav, D. K. 2015. A Fuzzy Logic based Approach for Phase-wise Software Defects Prediction using Software Metrics. Information and Software Technology, 63, 44-57.

[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] Mohanta, S., Vinod, G., and Mall, R. A. 2011. Technique for Early Prediction of Software Reliability based on Design Metrics. International Journal of System Assurance Engineering and Management, 2(4), 261-281.

[26] He, P., Li, B., Liu, X., Chen, J., and Ma, Y. 2015. An Empirical Study on Software Defect Prediction with a Simplified Metric Set. Information and Software Technology, 59, 170-190.

[27] Li, M., and Smidts, C. 2003. A ranking of software engineering measures based on expert opinion. IEEE Transaction on Software Engineering, 29(9), 811–824.

[28] Martin, N., Fenton, N., and Nielson, L. 2000. Building large-scale Bayesian networks. The Knowledge Engineering review, 15(3), 257–284.

[29] Radjenovic, D., Hericko, M., Torkar, R., and Zivkovic, A. 2013. Software Fault Prediction Metrics: A Systematic Literature Review. Information and Software Technology, 55(8), 1397-1418.

[30] Andersson, M., and Vestergren, P. 2004. Object Oriented Design Quality Metrics. Uppsala Master's Thesis in Computer Science 276, ISSN 11001836, 1-27.

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

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

[33] Birkmeier, D. Q. 2010. On the State of the Art of Coupling and Cohesion Measures for Service-Oriented System Design metrics. Proceedings of Conference on Information Systems (AMCIS), 1-10.

[34] Breesam, K. M. 2007. Metrics for Object-Oriented Design Focusing on Class Inheritance Metrics. 2nd International Conference on Dependability of Computer Systems, June 14-16, 2007, IEEE Computer Society, 231-237.

[35] Dallal, J. A. 2010. Mathematical Validation of Object-Oriented Class Cohesion Metrics. International Journal of Computers, 4(2), 45-52.

[36] Gray, C. L. 2008. A Coupling Complexity Metric Suit for Predicting Software Quality. Thesis submitted to Polytechnic State University, California, 1-71.

[37] Yadav, A., and Khan, R. A. 2012. Development of Encapsulated Class Complexity Metric. International Conference on Computer, Communication, Control and Information Technology (CCCIT-2012), Procedia Technology, 754-760.

[38] Yadav, A., and Khan, R. A. 2011. Class Cohesion Complexity Metric (C_3M). Proceedings of International Conference on Computer and Communication Technology (ICCCT-2011), 363-366.

[39] Yong, C., and Qingxin, Z. 2008. Improved Metrics for Encapsulation Based on Information Hiding. 9th International Conference for Young Computer Scientists, IEEE computer society, 742-724.

[40] Kumar, K. S., and Misra, R. B. 2008. An enhanced model for early software reliability prediction using software engineering metrics. Proceedings of 2nd International Conference on Secure System Integration and Reliability Improvement, 177–178.

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

[42] 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.

[43] 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.