Early Stage Software Reliability Modeling using Requirements and Object-Oriented Design Metrics: Fuzzy Logic 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
In the current scenario as the influence of information technology has been rising day by day, the industry is facing the pressure of developing software with higher level of reliability. Generally it is an accepted fact that the roots of unreliability lies in ill defined requirements and design documents. With this spirit, researcher has proposed and implemented a reliability prediction model through fuzzy inference system that utilizes early stage product based measures from requirements and object-oriented design stages. The study starts with the review findings those have been used as foundation for proposing a reliability quantification framework. Subsequently this framework has implemented in the form of reliability prediction model that predicts reliability at the requirements as well as design level through its output variable. The model has been validated as well as quantitatively compared with two existing reliability models. The obtained results are quite encouraging and supports that the proposed framework and reliability prediction model are better. Consideration of requirements phase along with the object-oriented design provides this paper an edge over other similar studies those are based on only design phase. Because ignoring requirements deficiencies and only concentrating on design constructs will not help in developing reliable software.

Keywords
Software Requirements, Software Reliability, Fuzzy Logic, Early Reliability Prediction, Object-oriented Design, Software Reliability Model.

1. INTRODUCTION
With the start of the twenty-first century it is observed that every sector of the society is depending more on software than before. The presence of software is impacting directly or indirectly, almost everyone living on the globe [1]. Whether it is transportation, health, defense, telecommunication, e-commerce, entertainment, home appliances etc. all domains are governed by the software directly or indirectly. Nobody can think about a life without the devices controlled by software. Reliability is a quality factor that needs to be assured in almost all safety-critical systems [2]. Industry is under pressure to develop and deliver reliable and quality software with shorter lead-times and low development costs. Over the last two decades software reliability has become one of the key factors that are being considered as a differentiator among different competitors in the industry. It is the product's reliability that establishes the success of a company in the global market. Literature has defined software reliability as the probability of failure free operations for a specified period of time in a specific environment [3].

Review of the literature highlights several unfortunate events that had already occurred in various domains due to unreliability of corresponding software applications [4]. In general revealing the presence of defects is considered as a method to measure the reliability. The reliability of software depends on the number of defects those originates in early stages and subsequently propagated undetected to later stages of development [5]. After realizing reliability as one the key quality attribute, its prediction cannot be delayed or ignored. While there exists a significant number of reliability models in the literature that estimates or predicts reliability, at various development stages, by utilizing different measures as well as variety of techniques, but there is no work in the literature that has considered the combination of requirements stage measures with object-oriented design for predicting the reliability of the developing software before its coding starts. Even though, it is a universally accepted statistic that 70 - 80% of all the faults in software are get introduced during the requirements phase [6], this phase of development lifecycle had not been given needed importance while predicting the reliability. Because timing of prediction is the key for the final quality of any software product, the more early it is monitored or control the higher level of reliability can be achieved [7]. Majority of existing reliability models are applicable only in the later stages of development, and helping developers either by the end of coding phase or in the testing stage. That becomes too late for developers to take corrective measure to improve its reliability as well as quality [8]. Therefore, in order to fill the above identified gap, it appears highly advantageous and significant to develop a reliability prediction model that will consider requirements and object-oriented design measures for predicting the reliability before the coding of the software starts.

The rest of the paper is organized as follows; section 2 describes the state-of-art on reliability prediction studies. Section 3 presents the overview of reliability quantification framework. Framework has been systematically implemented as a reliability prediction model in section 4. Section 5 of the paper statistically validated the reliability model, and its predictive accuracy results are presented in section 6. Section 7 quantitatively compared the developed model with two existing reliability models and finally the paper concludes with future work in section 8.

2. RELATED WORK
During the last three decades the literature has been witnessing a significant number of reliability studies [9, 10, 11, 12, 13, 14, 15]. The researcher has already critically reviewed some of these studies in earlier papers [16, 17, 18]. However, following paragraphs are further reviewing some

recent as well as critical efforts in the domain of reliability engineering. In a study [19] Jaiswal and Giri, presented a model for reliability estimation of component-based software. Along with this model author had also proposed a model for quantifying the reusability. Five quality attributes (understandability, variability, portability, maintainability and flexibility) were identified to estimate reusability. But an important observation is that even though these factors may have different degree of influence on reusability, all were multiplied by a constant value (i.e. 0.2) in the model's equation. The study did not mention any justification in this regards. Besides that development as well as validation process had not been described clearly. It is unclear how accurate the reliability prediction given by this approach would be.

In another study [20] Kumar and Dhanda, developed a model for predicting the reliability of object-oriented design. Initially the study had developed two multivariate regression models for computing effectiveness as well as functionality, subsequently these two quality attributes were used as independent variables for estimating the software reliability at the design stage. But the authors did not justified why effectiveness and functionality were used in reliability prediction in the presence of other factors that have more significant impact on reliability. Another study [21], developed two multivariate regression models for quantifying software complexity and reliability of object-oriented design. Initially complexity was estimated in terms of encapsulation, cohesion, inheritance and coupling, followed by reliability computation in terms of complexity. But the author had not justified the goodness or statistical significance of neither of the multivariate model. It is unclear how competently these models are estimating their respective dependent variables, besides that the significance of individual independent variable was not shown to justify their involvement as independent variables in the complexity model. Although, 't' test of statistic might be used for this. Beside that one weakness that had not been taken care of by the author is the multicollinearity and autocorrelation, the two problem with the multiple linear regression.

During a similar effort [22] Wende Kong, presented an approach that focuses on the prediction of software reliability at the requirements phase. The point of attraction was to identify weaknesses in the SRS document, and how to make SRS correct and complete. The technique of Cause-Effect Graph Analysis was used for reliability prediction. The study mathematically formalized the Cause Effect Graph (CEG), and applied it on SRS to identify its faults, subsequently fault tree was built through the identified SRS faults. Binary Decision Diagram (BDD) approach had used with an algorithm to analyze the fault tree and quantifying the influence of the detected requirements faults on software reliability. Although the effort is quite influencing but the process of identifying SRS faults is totally manual, requires a good level of domain knowledge and understanding of the system under study along with inspector's creativity, experience and even intuition. Similarly the scalability is also one the issue, for large SRS it will be very difficult to build and analyze the Cause-Effect Graph (CEG). The author had also mention that validation process was not up to the mark and it is unclear how accurate the reliability prediction given by this approach would be. One more important issue was that without prior and comprehensive knowledge of the system, the faults found through CEGA may not be correct and the final reliability estimation may not be very meaningful, besides that proposed approach is very costly and also time-consuming, specially, to construct an initial Cause Effect Graph (CEG) from a given informal specification. One more point is that not every aspect of software will be specifiable by a CEG, because a CEG can only capture functional requirements specified in the SRS. CEG analysis could not detect hidden requirements.

In another study [23], regarding utilizing formal method for reliability quantification Hooshmand and Isazadeh, performed an effort for early software reliability assessment on the basis of software behavioral requirements. Viewchart was used (as formal method) to specify the behavior description of software systems. The concept of Markov chain was also used with viewchart, to know the rate of system's transition from one stage to other. The study further predicated some states, for each of the system's view, those may cause system failures, and assess software reliability as the union of the probabilities of these failure states. But some finding have been noticed during the critical review like as the reliability assessment is totally based on the union of the probabilities of failure states, therefore for each of the view identifying and introducing the probable events those may cause a system failure, needs the comprehensive knowledge about the different behaviors of the system. Also the study had not specified any rule or guidelines for drawing the viewchart specification from the corresponding system behavior. Further to compute system state transition rate, a system's prototype has to be develop on the basis of its viewchart specifications. And subsequently this prototype needs to be executed with some input values belonging to the corresponding operational profile. This makes the approach quite complicated and expert specific, especially at the requirement stage. Apart from these there is also an issue of scalability, developing viewchart specification for a system of significant size and complexity would be a challenging task.

After revisiting a range of studies on reliability prediction or estimation, the critical findings summed up as follows:

- No consensus or standard steps/procedure among researchers for predicting software reliability.
- Studies utilizing multiple linear regression for reliability quantification, had not bothered about multicollinearity and autocorrelation at all.
- The appropriate size as well as quality level of the dataset has been a serious concern for empirical analysis.
- Reliability quantification should also be accompanied by suitable suggestive measures so that in future proactive actions could be initiated in time.
- One of the observations that cannot be overlooked is the need of timely identification and subsequent fixation of residual defects so that reliable software could be delivered in time.
- The best time to detect and arrest faults is the requirements and design stages. To accomplish this task researchers are bound to use quality measures based on these stages. But usually most of metric values in early stages are subjective as their sources are subjective, like opinions of domain experts.
- Therefore, to deal with such intrinsic subjectivity and vagueness, fuzzy techniques have come up as a dependable tool in capturing and processing these early stage metric values.
- There are just a few attempts where fuzzy techniques were used to quantify the reliability. But the key concern is the time and the stage of SDLC. These

models are helping developers either by the end of coding phase or in the testing stage. These feedbacks make it too late to improve the existing product towards a more reliable one.

After going through the critical issues raised in above paragraphs, it is needed to advise some way out that will triumph over the shortcomings identified and highlighted in above points. Therefore, in the next section the researcher is going to present a roadmap in the form of a prescriptive framework.

3. RELIABILITY QUANTIFICATION FRAMEWORK (^{FL}SRQF)

In continuation with the highlighted need and significance as discussed in previous section, the researcher has already proposed a structured framework (Fuzzy Logic based Software Reliability Quantification Framework (^{FL}SRQF)) as a solution for the identified inadequacies present in earlier reliability prediction studies [24]. The framework described a comprehensive reliability quantification process through its eight phases (Conceptualization, Identification, Association, Quantification, Corroboration, Analysis, Assessment and Amendment and Packaging) as depicted in fig 1. It has been structured in a manner that could be easily implementable by industry personnel as well as researchers. The focus of the framework is on the requirement and design phase of the development life cycle. In [24] the researcher had comprehensively described all the phases of the framework along with its salient characteristics those support its claim to be a better reliability framework.

4. FRAMEWORK IMPLEMENTATION

This section of the paper is going to systematically implement each phase of the proposed framework (^{FL}SRQF). In order to implement the framework the researcher has developed a model as depicted in the figure 2. The model is referred as Early Stage Reliability Prediction Model (ESRPM) and is based on the assumption that the software reliability and its quality are adversely affected by the weaknesses of requirements and design constructs. Therefore the model focuses on these two, most significant, early phases of SDLC. Looking at the architecture of the model it can be easily noticed that the model integrates requirements and object-oriented design measures as input to the fuzzy inference system and predict the reliability of the developing software up to its design stage before the coding starts.

4.1 Implementing Conceptualization Phase

As far as this phase of the framework is concern, it provides foundation for the rest of the phases. It is considered as the primary step to device a comprehensive solution for an important problem. As shown in figure 1, it has four sub tasks: Assess Need and Significance; Explore Advantage at Early Stage; Assess the Contribution of Fuzzy Logic; Explore Developmental Feasibility. All these four conceptual sub-tasks have already been discussed in the first two sections of this paper. Therefore the researcher is not going to repeat it again.

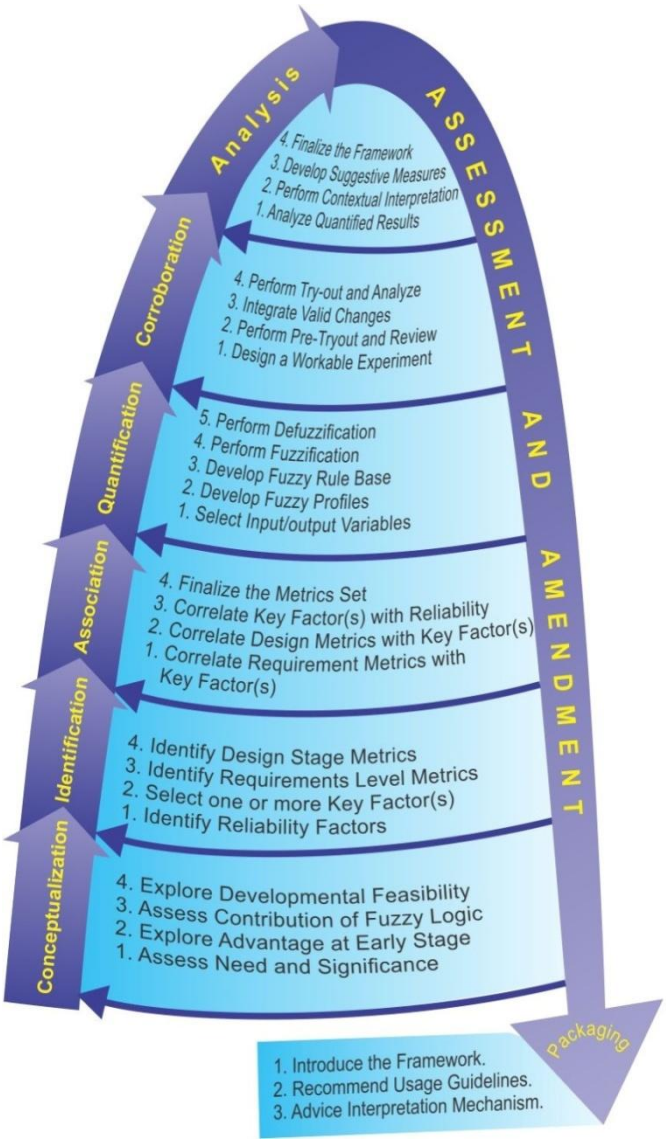

Fig 1: Software Reliability Quantification Framework

4.2 Implementing Identification Phase

In order to reach to an appreciable solution, it is needed to identify the factors that are influencing directly or indirectly to the problem and its solution. The objective of the identification phase is to identify the factors that are related directly or indirectly to the reliability prediction. There is no doubt, that quantified reliability will not have significant value if its underlying factors are not identified appropriately [24].

4.2.1 Identify Reliability Factors

In this study the researcher has followed the methodology suggested by Dromey [25] that is to quantify any higher level quality attribute, it should be decomposed into lower level attributes. Therefore to quantify the reliability as per this methodology, researcher has shortlisted the some researches, highlighting a variety of factors impacting the reliability positively or negatively. After scanning McCall's [26], Dromey [27], Boehm [28], ISO/IEC 9126 and ISO, 2001 [29] researcher has shortlisted twelve factors shown in figure 3.

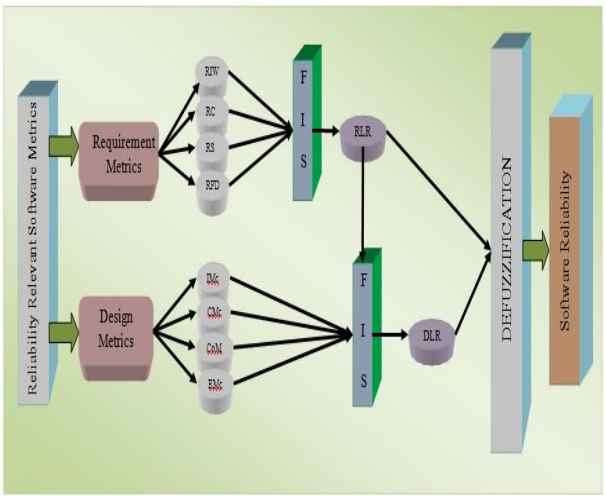

Fig 2: Early Stage Reliability Prediction Model (ESRPM)

4.2.2 Identify Requirements Level Metrics

After recognizing the criticality of requirements stage for early reliability prediction, it is needed to consider appropriate measures form this stage. Consequently, study has focused on the identification of reliability-relevant software metrics and gathered following requirements metrics [12, 30, 31, 32].

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

Fig. 3: Reliability Factors

4.2.3 Identify Design Stage Metrics

As the study concentrates on four object-oriented design constructs therefore the researcher has gathered following object-oriented design metrics from the literature [33, 34, 35, 36, 37, 38, 39, 40, 41, 42]. LCOM (Lack of Cohesion in Methods), AIF (Attribute Inheritance Factor), MPC (Message Pass Coupling), DIT (Depth of Inheritance), IMc (Inheritance Metric Complexity Perspective), NOC (Number of Children), EMc (Encapsulation Metric Complexity Perspective), WMC (Weighted Method per Class), CBO (Coupling Between Objects), Response for a Class (RFC), CoMc (Cohesion Metric Complexity Perspective), CMc (Coupling Metric Complexity Perspective), DAC (Data Abstraction Coupling) and AHF (Attribute Hiding Factor).

4.3 Implementing Association Phase

The aim of this phase in the proposed framework is to align all the components together by justifying their role in the early prediction. On the basis of their part to predict the reliability, the researcher has shortlisted eight metrics out of twenty six (Requirements (12) and Design (14)) metrics identified in the previous identification phase. Out of these eight metrics four belongs to requirement phase (RS, RIW, RC and RFD) and four belongs to object-oriented design (IMc, CMc, EMc and CoMc). Following paragraphs are providing a brief description about the selected metrics along with their relationship with software reliability.

RS: Requirements Stability is inversely proportional to the number of change request initiated by the client regarding software requirements. Higher frequency of change requests give rise to the probability of errors that may be creep into the requirements documents, and subsequently infect the subsequent phases of development [31].

More Change Requests => Low Requirements Stability => Less Reliability

RIW: Similarly RIW (Review, Inspections and Walkthrough) is also a valuable mean for identification as well as rectification of requirements faults to improve its reliability. More the number of RIWs the more error free the SRS will be [31].

High RIW => More Defect Identification and Removal => More Reliable SRS

RFD: Third metric RFD (Requirements Fault Density) measures the fraction of faulty requirements specification document. Requirement fault density provides an indicator of the software quality of developing software during requirement analysis phase [31].

High Fault Density => Low Reliability

RC: Similarly the fourth identified metric RC (Complexity of New Functionality) also negatively impact the reliability of the developing software [31].

High value of RC => Make the SRS Complex => Low Reliability

IMc: Inheritance metric (complexity perspective) provides overall complexity of a design hierarchy through inherited methods and attributes and estimated by taking the average of 'Inheritance metric complexity perspective of every class' [21, 43].

CMc: Coupling metric complexity perspective computes the overall complexity of the design hierarchy, through aggregating the coupling of involved classes in the design [21].

EMc: Encapsulation metric (complexity perspective) provides overall complexity of a design hierarchy through encapsulated methods and attributes and is estimated by taking the average of 'Encapsulation metric complexity perspective of every class' [40].

CoMc: Cohesion metric complexity perspective is defined as the average of 'Cohesion metric complexity perspective per class' [36, 41, 21].

Therefore in summarized form it can be shown how the selected requirements and design metrics are associated with the reliability. *RS* α *Reliability; RC* α *1/Reliability; RFD* α

1/Reliability; RIW α Reliability; IMc α 1/Reliability; CMc α 1/Reliability; CoMc α Reliability; EMc α Reliability.

4.4 Implementing Quantification Phase

It is the most critical phase of the framework, because the actual development of the reliability prediction model takes place in this phase itself. The model is implemented in MATLAB utilizing fuzzy logic toolbox. The basic steps of the model development are selection of reliability-relevant software metrics as input/output variables, development of fuzzy profile of these input/output variables, building the fuzzy rule base and reliability prediction at the end of requirements and design phase using fuzzy inference system (FIS).

4.4.1 Select Input and Output Variables

As already discussed in the identification phase that out of total eight metrics four (RS, RIW, RC, RFD) have been selected for the requirements phase and rest four (EMc, CoMc, CMc, and IMc) for the design phase. These metrics (shown in Table 1) are considered as input variables for the fuzzy based reliability prediction model (ESRPM) and can be applied to the requirement and design phases. 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.

Table 1. Input and Output Variables

Phase	Input Variable	Output Variable
Requirement	RS, RIW, RC, RFD	RLR
Design	RLR, EMc, CoMc, CMc, IMc	DLR

4.4.2 Develop Fuzzy Profiles

Input/output variables selected at the previous steps are fuzzy in nature and are characterized by membership function. Developing a membership function with help of domain expert knowledge is one of the basic steps in the design of a problem which is to be solved by fuzzy set theory. In this research, membership functions of all the input and output metrics are defined with the help of domain experts. Membership function can have a variety of shapes like polygonal, trapezoidal, triangular, and so on [44].

Table 2. Fuzzy Profiles for Requirements Measures

Value	RC (0-1)	RS (0-1)	RFD (0-1)	RIW (0-1)	RLR (0-1)
Very low					(0;0;0.35)
Low	(0;0;0.3)	(0;0;0.35)	(0;0;0.4)	(0;0;0.4)	(0.25;0.4;0.55)
Medium	(0.2;0.4;0.6)	(0.25;0.45;0.75)	(0.2;0.4;0.7)	(0.2;0.4;0.6)	(0.45;0.6;0.85)
High	(0.5;1;1)	(0.6;1;1)	(0.5;1;1)	(0.4;1;1)	(0.65;0.8;0.95)
Very high					(0.85;1;1)

In this research triangular membership functions are considered for fuzzy profile development of identified input/output variables. Triangular membership functions (TMFs) are widely used for calculating and interpreting reliability data because they are simple and easy to understand [45]. Also, they have the advantage of simplicity and are commonly used in reliability analysis.

Table 3. Fuzzy Profiles for Design Stage Measures

Value	RLR (0-1)	IMc (0-1)	EMc (0-1)	CMc (0-1)	CoMc (0-1)	DLR (0-1)
Very low	(0;0;0.35)					(0;0;0.3)
Low	(0.25;0.4;0.55)	(0;0;0.4)	(0;0;0.35)	(0;0;0.4)	(0;0;0.4)	(0.2;0.35;0.5)
Medium	(0.45;0.6;0.85)	(0.3;0.5;0.7)	(0.25;0.45;0.75)	(0.25;0.5;0.7)	(0.3;0.5;0.75)	(0.4;0.55;0.7)
High	(0.65;0.8;0.95)	(0.6;1;1)	(0.65;1;1)	(0.6;1;1)	(0.65;1;1)	(0.6;0.75;0.9)
Very high	(0.85;1;1)					(0.8;1;1)

Fuzzy membership functions are generated utilizing the linguistic categories such as very low (VL), low (L), medium (M), high (H), and very high (VH), identified by a human expert to express his/her assessment. Table 2 and 3 lists the selected input/output variables along with their fuzzy range as well as profile. For visualization purpose these membership function are also shown in Figs. 4-13.

4.4.3 Develop Fuzzy Rule Base

In this step fuzzy rules are defined in the form of IF-THEN conditional statement. IF part of the rule is known as antecedent, and THEN part is consequent [44, 46]. The fuzzy rule base can be designed from different sources such as domain experts, historical data analysis, and knowledge engineering from existing literature [31, 47]. In this research the fuzzy rules that are required for the prediction of the reliability are defined with the help of domain experts. In case of the model developed in this study each of the four requirements phase input metrics has three linguistic states *i.e.*, low (L), medium (M) and high (H). Therefore, total number of rules is 81. Similarly in design phase total number of rules is 405.

4.4.4 Perform Fuzzification

In this phase, fuzzy inference engine evaluates and combines the result of each fuzzy rule. It maps all the inputs to an output. This process of mapping inputs onto output is known as fuzzy inference process [7, 46]. The two main activities for information processing are as follows: combining input from all the ‘‘if’’ part of fuzzy rules and aggregation of ‘‘then’’ part to produce the final output. The Mamdani fuzzy inference system [48] is considered here for all the information processing.

4.4.5 Perform Defuzzification

Defuzzification is the process of deriving a crisp value from a fuzzy set using any defuzzification methods such as centroid, bisector, middle of maximum, largest of maximum and smallest of of maximum [44]. Centroid method is used in the

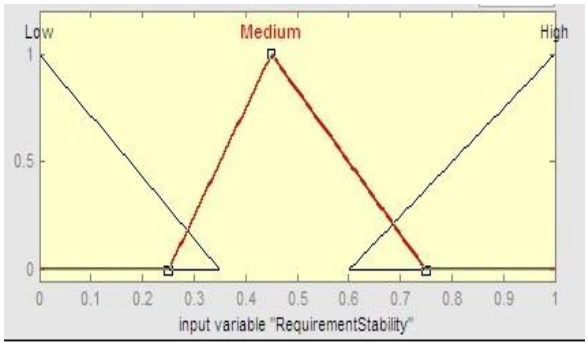

Fig. 4 Fuzzy Profile of RS

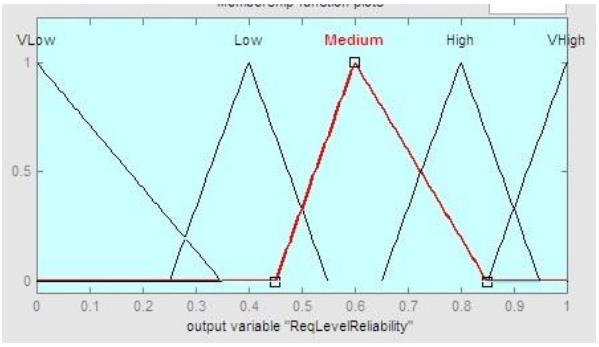

Fig. 8 Fuzzy Profile of RLR

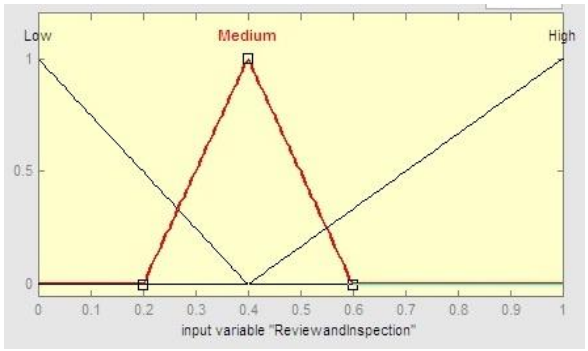

Fig. 5 Fuzzy Profile of RIW

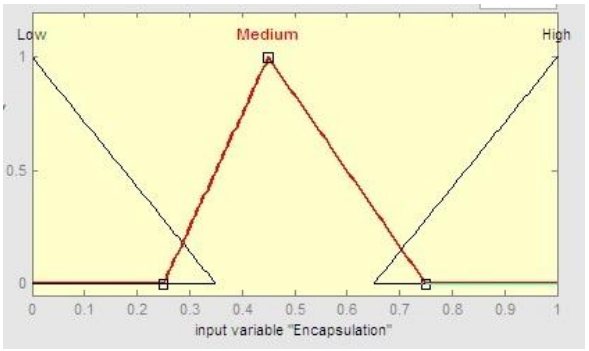

Fig. 9 Fuzzy Profile of EMc

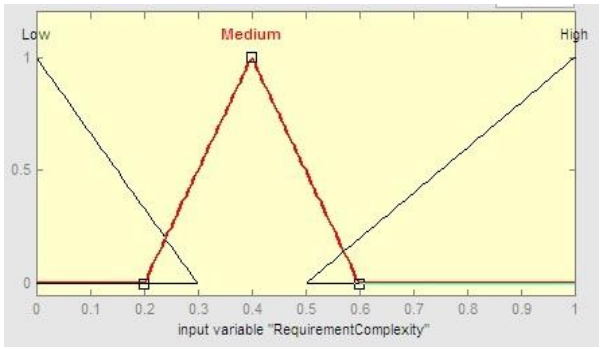

Fig. 6 Fuzzy Profile of RC

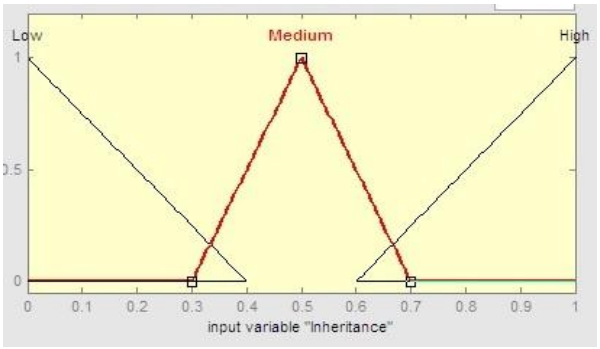

Fig. 10 Fuzzy Profile of IMc

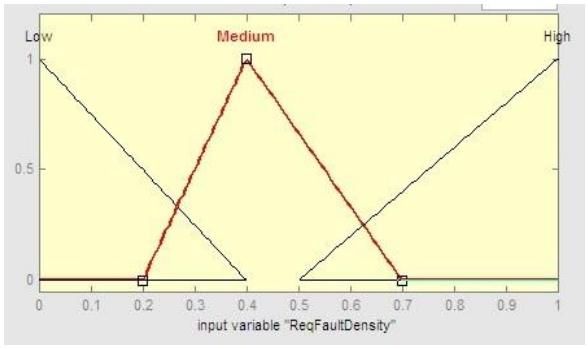

Fig. 7 Fuzzy Profile of RFD

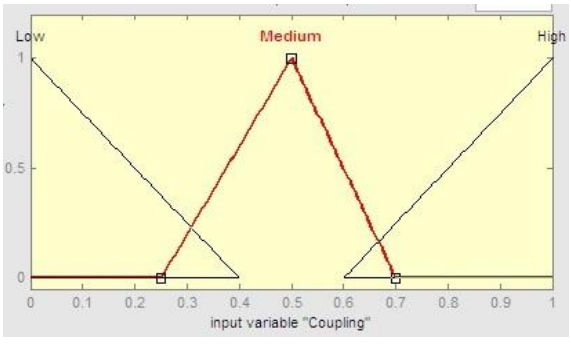

Fig. 11 Fuzzy Profile of CMc

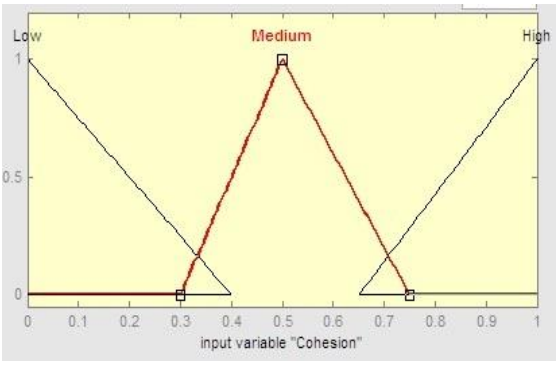

Fig. 12 Fuzzy Profile of CoMc

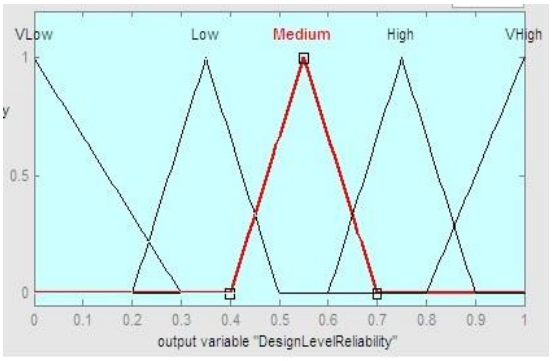

Fig. 13 Fuzzy Profile of DLR

present research for finding the crisp value, representing the requirements and design level reliability at the end of requirements and design phase respectively.

4.5 Implementing Corroboration Phase

Although the developed reliability prediction model (ESRPM) has been corroborated empirically in the next Section, even though in order to analyze the fault prediction consistency and influence of various software metrics on early fault prediction some analysis has been presented.

Table 4. 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 5. Reliability Prediction at Design Stage

	RLR	EMc	CoMc	IMc	CMc	DLR
Worst Case	0	0.1	0.1	0.9	0.9	0.096
Average 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 4 and 5, presents the values of RLR (Requirements Level Reliability) and DLR (Design Level Reliability) by the proposed model 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 extremes. 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 software reliability. Once the impact of the particular software metric 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.

4.6 Implementing Analysis Phase

After implementing the quantification phase successfully this is the next critical phase of the framework. The following sub sections analyses different quantitative input as well as output values and inferred the suggestive measures along with the guidelines for improving the software reliability.

4.6.1 Sensitivity Analysis

In order to justify the influence of software metrics in the proposed model, sensitivity analysis has been preformed. In this analysis, the impact of input variable on output variable is analyzed. It is desirable to know the significance of input metrics in software reliability prediction. As explained in the previous section that the Design Level Reliability (DLR) has been computed in terms of Requirements Level Reliability (RLR), along with four other Object Oriented Design metrics (IMc, EMc, CMc, CoMc). While, the value of RLR depends on four requirements stage metrics RS, RIW, RC and RFD. Therefore, it seems important to determine the impact of a particular software metrics on the software reliability. Once the impact of the particular software metrics on reliability has been inferred, the better and more proactively it can be controlled to improve the overall reliability as well as quality of the product. Figure 14-31 are elaborating the sensitivity of RLR or DLR with respect to various input variables.

4.6.2 Quantified Reliability and Metrics

After ensuring that the developed model is running successfully, in this phase various artifacts involved in the reliability prediction needs to be further analyzed to know more about their behavior. The following sub sections will perform this task for requirements and design phase separately. Figure 14-31 are elaborating the sensitivity of RLR or DLR with respect to various input variables.

4.6.2.1 Analyzing the Requirements Metrics

Observing the quantitative change in the Requirements Level Reliability (RLR), on the basis of the quantitative variation in the values of requirement metrics, following observations are noticed:

(a) Individual Variation

As the value of RS moves towards 0 to 1 the value of RLR also increases from 0 to 1.

As the value of RS moves towards 1 to 0 the value of RLR also decreases towards 0.

As the value of RIW moves towards 0 to 1 the value of RLR also move from 0 to 1.

As the value of RIW decreases from 1 to 0 the RLR also decreases in the same direction.

As the value of RC moves from 1 to 0 the value of RLR move in opposite direction (0 to 1).

As the value of RC increases from 0 to 1 the value of RLR decreases towards 0.
As the value of RFD moves towards 1 the value of RLR move in reverse direction (1 to 0).
As the value of RFD decreases, the value of RLR increases from 0 to 1.

(b) Combinational Variation

As the values of RC along with RFD move towards 1 to 0 the value of RLR moves towards 0 to 1.
As the values of RC and RFD move towards 0 to 1 the RLR decrease from 1 towards 0.
As the values of RS along with RIW decreases, the value of RLR responds in the same direction.
As the values of RS and RIW move from 0 to 1 the value of RLR also increases from 0 to 1.
As the values of RC and RIW move towards 1 or 0 the value of RLR neither increases nor decreases.
As the values of RS and RFD vary from 0 to 1 or 1 to 0 the value of RLR neither increases nor decreases.
As the values of RC and RS move towards 1 or 0 the RLR reflects no influence, neither increases nor decreases.

After going through afore mentioned empirical observations in the form of individual and combinational variations, following conclusion may be drawn.

"Higher the value of RS the more reliable the requirements will be"

"Higher the value of RC the less reliable the requirements will be"

"Higher the value of RIW the more reliable the requirements will be"

"Higher the value of RFD the less reliable the requirements will be"

4.6.2.2 Analyzing the Design Metrics

Similarly, observing the quantitative change in the Design Level Reliability (DLR), on the basis of the quantitative variation in the values of Object-Oriented Design metrics following observations are noticed:

(a) Individual Variation

As the value of RLR moves from 0 to 1, the value of DLR also increases in the same direction.
As the value of RLR decreases from 0 to 1, the value of DLR also decreases from 0 to 1.
As the value of EMc increases towards 1 the value of DLR also increase.
As the value of EMc decreases towards 0 the value of DLR also decreases.
As the value of CoMc moves towards 1 the value of DLR also increases.
As the value of CoMc moves towards 0 the value of DLR also decreases.
As the value of IMc decreases from 1 to 0, the value of DLR increases towards 1 from 0.
As the value of IMc moves towards 1 from 0, the value of DLR decreases towards 0 from 1.
As the value of CMc moves from 0 to 1, the value of DLR moves in reverse direction.
As the value of CMc moves towards 0 the value of DLR increases.

RS = 0%		RS = 25%		RS = 50%		RS = 75%		RS = 100%	
RIW = 0%		RIW = 25%		RIW = 50%		RIW = 75%		RIW = 100%	
RC = 0%		RC = 25%		RC = 50%		RC = 75%		RC = 100%	
RFD	RLR	RFD	RLR	RFD	RLR	RFD	RLR	RFD	RLR
0.0	0.633	0.0	0.544	0.0	0.665	0.0	0.800	0.0	0.800
0.2	0.639	0.2	0.544	0.2	0.665	0.2	0.800	0.2	0.800
0.4	0.633	0.4	0.544	0.4	0.665	0.4	0.641	0.4	0.633
0.6	0.569	0.6	0.427	0.6	0.602	0.6	0.642	0.6	0.642
0.8	0.400	0.8	0.263	0.8	0.486	0.8	0.641	0.8	0.637
1.0	0.400	1.0	0.263	1.0	0.486	1.0	0.641	1.0	0.633

Figure:14 Sensitivity of RLR with respect to RFD

RFD = 0%		RFD = 25%		RFD = 50%		RFD = 75%		RFD = 100%	
RIW = 0%		RIW = 25%		RIW = 50%		RIW = 75%		RIW = 100%	
RC = 0%		RC = 25%		RC = 50%		RC = 75%		RC = 100%	
RS	RLR	RS	RLR	RS	RLR	RS	RLR	RS	RLR
0.0	0.633	0.0	0.544	0.0	0.665	0.0	0.400	0.0	0.400
0.2	0.640	0.2	0.544	0.2	0.669	0.2	0.400	0.2	0.400
0.4	0.635	0.4	0.583	0.4	0.665	0.4	0.639	0.4	0.635
0.6	0.639	0.6	0.583	0.6	0.665	0.6	0.639	0.6	0.639
0.8	0.800	0.8	0.724	0.8	0.800	0.8	0.639	0.8	0.639
1.0	0.800	1.0	0.724	1.0	0.800	1.0	0.639	1.0	0.633

Figure:15 Sensitivity of RLR with respect to RS

RS = 0%		RS = 25%		RS = 50%		RS = 75%		RS = 100%	
RIW = 0%		RIW = 25%		RIW = 50%		RIW = 75%		RIW = 100%	
RFD = 0%		RFD = 25%		RFD = 50%		RFD = 75%		RFD = 100%	
RC	RLR	RC	RLR	RC	RLR	RC	RLR	RC	RLR
0.0	0.633	0.0	0.551	0.0	0.665	0.0	0.800	0.0	0.800
0.2	0.642	0.2	0.551	0.2	0.676	0.2	0.800	0.2	0.800
0.4	0.633	0.4	0.551	0.4	0.665	0.4	0.641	0.4	0.633
0.6	0.400	0.6	0.265	0.6	0.534	0.6	0.645	0.6	0.645
0.8	0.400	0.8	0.271	0.8	0.486	0.8	0.641	0.8	0.637
1.0	0.400	1.0	0.271	1.0	0.486	1.0	0.641	1.0	0.633

Figure:16 Sensitivity of DLR with respect to RC

EM = 0%		EM = 25%		EM = 50%		EM = 75%		EM = 100%	
IM = 0%		IM = 25%		IM = 50%		IM = 75%		IM = 100%	
CM = 0%		CM = 25%		CM = 50%		CM = 75%		CM = 100%	
RLR = 0%		RLR = 25%		RLR = 50%		RLR = 75%		RLR = 100%	
CoM	DLR	CoM	DLR	CoM	DLR	CoM	DLR	CoM	DLR
0.0	0.350	0.0	0.350	0.0	0.450	0.0	0.450	0.0	0.550
0.2	0.350	0.2	0.350	0.2	0.450	0.2	0.450	0.2	0.550
0.4	0.350	0.4	0.350	0.4	0.550	0.4	0.550	0.4	0.750
0.6	0.350	0.6	0.350	0.6	0.550	0.6	0.550	0.6	0.750
0.8	0.550	0.8	0.550	0.8	0.650	0.8	0.650	0.8	0.750
1.0	0.550	1.0	0.550	1.0	0.650	1.0	0.650	1.0	0.750

Figure:17 Sensitivity of DLR with respect to Cohesion

RS = 0%		RS = 25%		RS = 50%		RS = 75%		RS = 100%	
RFD = 0%		RFD = 25%		RFD = 50%		RFD = 75%		RFD = 100%	
RC = 0%		RC = 25%		RC = 50%		RC = 75%		RC = 100%	
RIW	RLR	RIW	RLR	RIW	RLR	RIW	RLR	RIW	RLR
0.0	0.633	0.0	0.544	0.0	0.639	0.0	0.400	0.0	0.400
0.2	0.639	0.2	0.544	0.2	0.639	0.2	0.400	0.2	0.400
0.4	0.633	0.4	0.544	0.4	0.639	0.4	0.641	0.4	0.633
0.6	0.800	0.6	0.700	0.6	0.800	0.6	0.642	0.6	0.642
0.8	0.800	0.8	0.700	0.8	0.800	0.8	0.641	0.8	0.636
1.0	0.800	1.0	0.700	1.0	0.800	1.0	0.641	1.0	0.633

Figure:18 Sensitivity of RLR with respect to RIW

CoM = 0%		CoM = 25%		CoM = 50%		CoM = 75%		CoM = 100%	
IM = 0%		IM = 25%		IM = 50%		IM = 75%		IM = 100%	
CM = 0%		CM = 25%		CM = 50%		CM = 75%		CM = 100%	
RLR = 0%		RLR = 25%		RLR = 50%		RLR = 75%		RLR = 100%	
EM	DLR	EM	DLR	EM	DLR	EM	DLR	EM	DLR
0.0	0.350	0.0	0.350	0.0	0.550	0.0	0.450	0.0	0.750
0.2	0.350	0.2	0.350	0.2	0.550	0.2	0.450	0.2	0.750
0.4	0.350	0.4	0.350	0.4	0.550	0.4	0.550	0.4	0.750
0.6	0.350	0.6	0.350	0.6	0.550	0.6	0.550	0.6	0.750
0.8	0.550	0.8	0.550	0.8	0.650	0.8	0.650	0.8	0.750
1.0	0.550	1.0	0.550	1.0	0.650	1.0	0.650	1.0	0.750

Figure:19 Sensitivity of DLR with respect to EM

EM = 0%		EM = 25%		EM = 50%		EM = 75%		EM = 100%	
CoM = 0%		CoM = 25%		CoM = 50%		CoM = 75%		CoM = 100%	
CM = 0%		CM = 25%		CM = 50%		CM = 75%		CM = 100%	
RLR = 0%		RLR = 25%		RLR = 50%		RLR = 75%		RLR = 100%	
IM	DLR	IM	DLR	IM	DLR	IM	DLR	IM	DLR
0.0	0.350	0.0	0.350	0.0	0.550	0.0	0.750	0.0	0.937
0.2	0.350	0.2	0.350	0.2	0.550	0.2	0.750	0.2	0.925
0.4	0.350	0.4	0.350	0.4	0.550	0.4	0.650	0.4	0.925
0.6	0.350	0.6	0.350	0.6	0.550	0.6	0.650	0.6	0.925
0.8	0.114	0.8	0.127	0.8	0.350	0.8	0.650	0.8	0.750
1.0	0.096	1.0	0.127	1.0	0.350	1.0	0.650	1.0	0.750

Figure: 20 Sensitivity of DLR with respect to Inheritance

EM = 0%		EM = 25%		EM = 50%		EM = 75%		EM = 100%	
IM = 0%		IM = 25%		IM = 50%		IM = 75%		IM = 100%	
CoM = 0%		CoM = 25%		CoM = 50%		CoM = 75%		CoM = 100%	
RLR = 0%		RLR = 25%		RLR = 50%		RLR = 75%		RLR = 100%	
CM	DLR	CM	DLR	CM	DLR	CM	DLR	CM	DLR
0.0	0.350	0.0	0.350	0.0	0.550	0.0	0.650	0.0	0.937
0.2	0.350	0.2	0.350	0.2	0.550	0.2	0.650	0.2	0.925
0.4	0.109	0.4	0.127	0.4	0.550	0.4	0.650	0.4	0.928
0.6	0.114	0.6	0.127	0.6	0.550	0.6	0.650	0.6	0.925
0.8	0.114	0.8	0.127	0.8	0.350	0.8	0.650	0.8	0.750
1.0	0.096	1.0	0.127	1.0	0.350	1.0	0.650	1.0	0.750

Figure: 21 Sensitivity of DLR with respect to Coupling

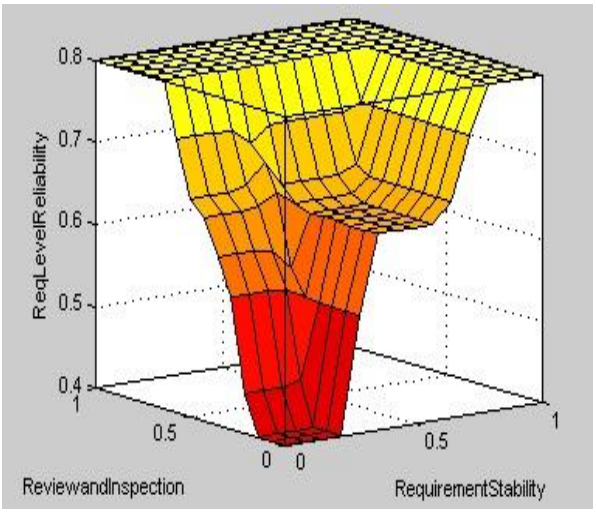

Figure:22 Sensitivity of RLR with respect to RIW and RS

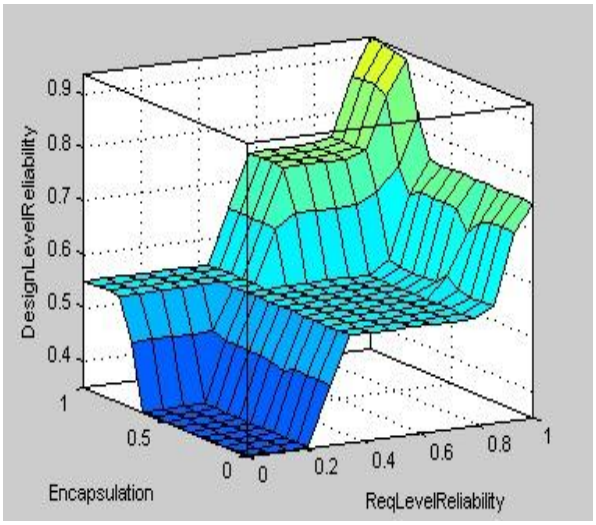

Figure: 23 Sensitivity of DLR with respect to RLR and EM

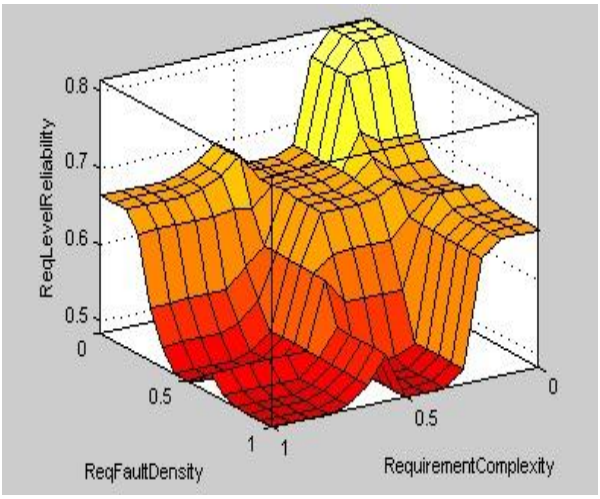

Figure: 24 Sensitivity of RLR with respect to RFD and RC

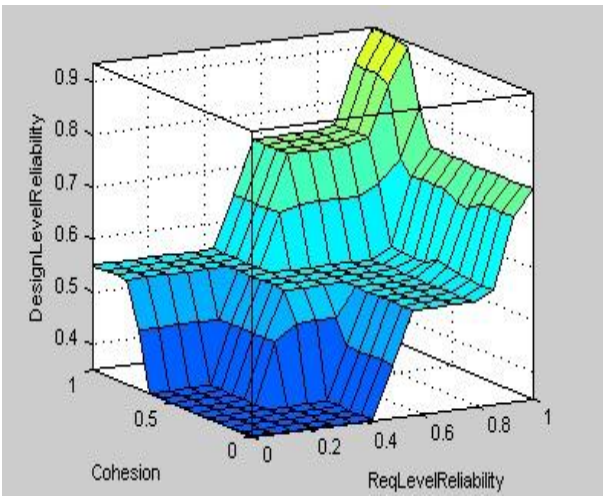

Figure:25 Sensitivity of DLR with respect to RLR and Cohesion

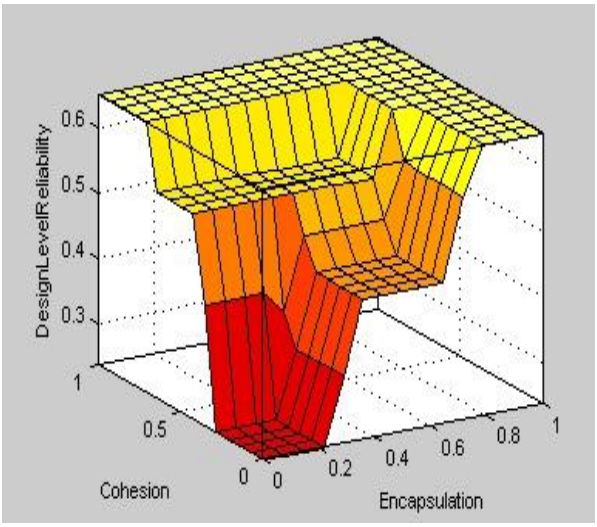

Figure:26 Sensitivity of DLR with respect to Cohesion and Encapsulation

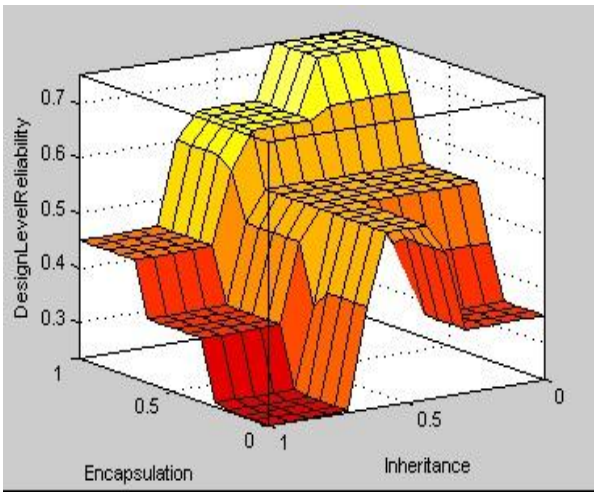

Figure:27 Sensitivity of DLR with respect to Encapsulation and Inheritance

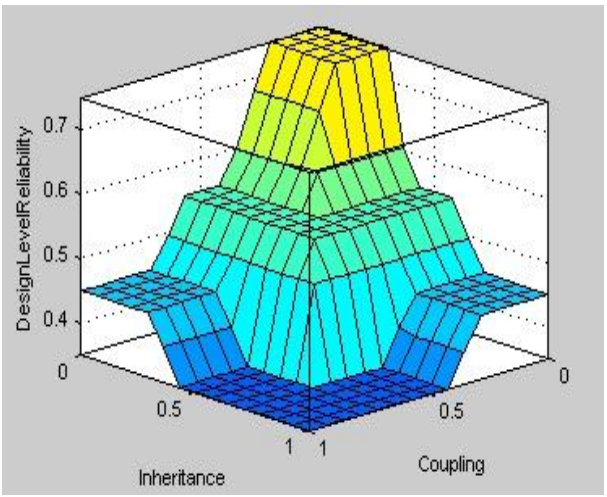

Figure:28 Sensitivity of DLR with respect to Inheritance and Coupling

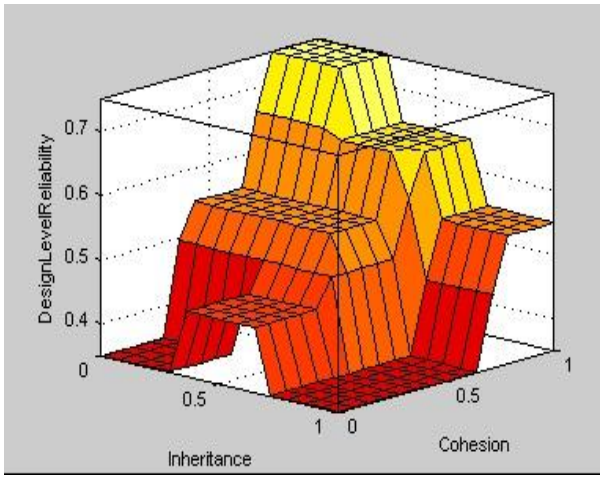

Figure:29 Sensitivity of DLR with respect to Inheritance and Cohesion

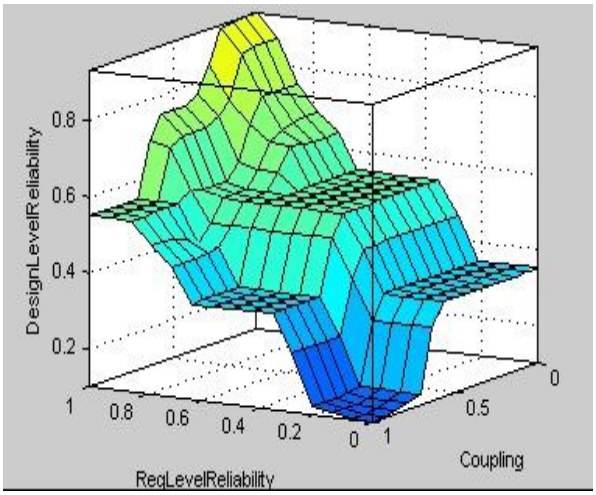

Figure:30 Sensitivity of DLR with respect to RLR and Coupling

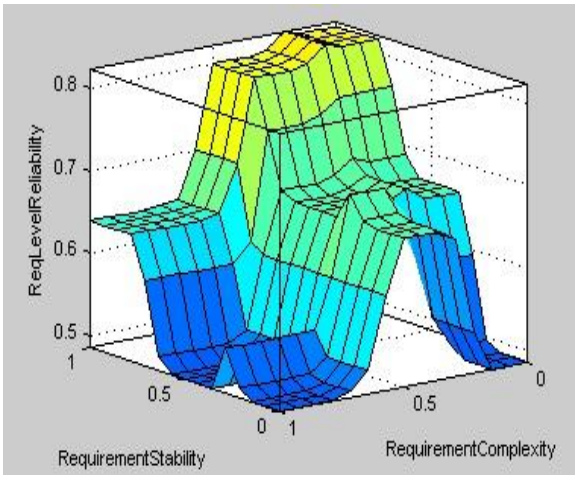

Figure:31 Sensitivity of RLR with respect to Requirement Stability and Complexity

(b) Combinational Variation

As the value of EMc along with CoMc move from 1 to 0 the value of DLR will also move in the same direction.

As the values of EMc and CoMc increase from 0 to 1, the value of DLR also moves towards 1.

As the values of IMc and CMc move from 1 to 0, the value of DLR will move in reverse direction.

As the values of IMc and CMc moved towards 0 to 1 the value of DLR moves from 1 to 0.

As the values of EMc and CMc vary from 1 or 0 the value of DLR neither increases nor decreases.

As the values of CMc and CoMc move towards 1 or 0 the change in DLR reflects no significant direction.

As the values of EMc and IMc moved towards 1 or 0 the value of DLR neither increases nor decreases.

Therefore on the basis of above observations following conclusion may be drawn.

"Higher the value of RLR the more reliable the design will be"
"Higher the value of CMc the less reliable the design will be"
"Higher the value of EMc the more reliable the design will be"
"Higher the value of IMc the less reliable the design will be"
"Higher the value of CoMc the more reliable the design will be"

4.6.3 Contextual Interpretation and Suggestive Measures

On the basis of the analysis being performed in the previous step, the next task is to frame different suggestive measures. These measures will be used as reliability improvement guidelines. These guidelines will assist to regulate the values of the requirements and design metrics, and improve the reliability of the developing software before the coding starts. Therefore following recommendations are developed for the personnel involved at the requirements and design phase during software development.

a) Keep the requirements change requests as low as possible.

b) Perform more and more review inspections and walkthrough.

c) Control and reduce the complexity of the newly added functional requirements.

d) Modules having complexity needs to handle by experienced requirement engineers.

e) Most critical modules should be review through senior and experienced peers.

f) Try to find as much SRS faults as possible, and try to reduce the fault density.

g) Identify the requirements ambiguities as well as inconsistencies as early as possible.

h) Keep the level of inheritance as low as possible, because unnecessary data members and methods in sub classes increase the complexity of the class hierarchy.

i) Keep the level of class coupling as minimum as possible.

j) Sharing of data and method across the methods of other classes must be avoided until necessary.

k) Critical data members should not be declared as public until unavoidable.

l) Sensitive data should not be passed as parameter to other methods.

m) Try to enhance the encapsulation in the design as much as possible.

n) Develop the classes as cohesive as possible.

o) Visibility options should be handle very intelligently. Only in unavoidable situations data and methods should be declared as public.

These revisions will definitely proved to be significant in making the finally delivered software more reliable.

5. EMPIRICAL VALIDATION OF THE ESRPM

This section assesses how effectively the reliability model (ESRPM) developed in the previous section 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 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. Subsequently in order to statistically validate the ESRPM, the 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 values of DLR have been computed using the fuzzy toolbox of MATLAB, for the aforesaid 20 software projects.

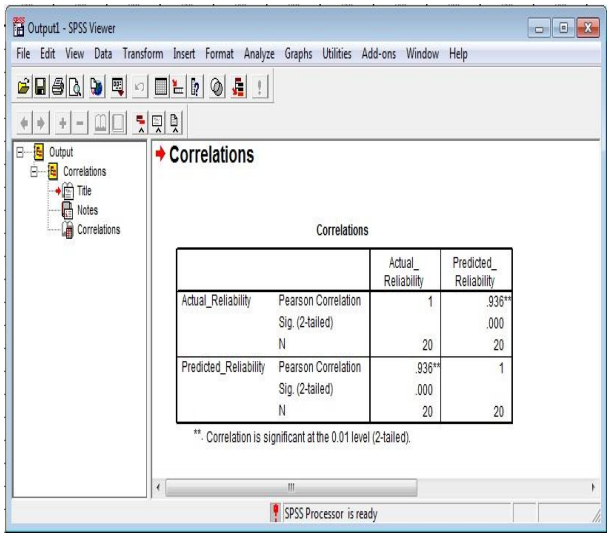

Figure: 32 SPSS Correlation Analysis

These calculated values are presented in the Table 6, along with the corresponding actual reliability values. The correlation has been computed through SPSS, and its value is (**0.936)** as shown figure 32. It is evident from the correlation value, that the reliability predicted by the ESRPM is strongly correlated with already known reliability values. Therefore, it can be concluded that the proposed model ESRPM is quantifying reliability quiet efficiently.

6. ESRPM's PREDICTIVE ACCURACY

Along with validating a model ensuring its predictive accuracy is one of the important aspects of any model being developed. Any improvement in the accuracy of reliability prediction can significantly impact the quality of the developing software application [49]. It is evident from the literature that the most popular measures include Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Balanced MRE, Mean Absolute Percentage Error (MAPE), Median Magnitude of Relative Error (MdMRE) and Prediction at level n (Pred(n)) [50, 51]. In order to compute the predictive accuracy, researcher has calculated the design level reliability (DLR) of software projects belonging to the data set using the fuzzy toolbox of MATLAB. Subsequently the Magnitude of Relative Error (MRE) has been computed. The values of actual reliability, reliability predicted through the model (ESRPM) and the corresponding MREs for all the 20 projects are shown in Table 6. Now after calculating the MRE values, the next task is to compute the Mean of these MRE values i.e. MMRE (Mean Magnitude of Relative Error).

Sum of MRE_1, MRE_2,…….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 et al [50] suggests an MMRE $\leq$ 0.25 as acceptable prediction accuracy for software development effort prediction models. After computing the MMRE, next important accuracy measures to be computed are Balanced Mean Magnitude of Relative Error (BMMRE) and Mean Absolute Percentage Error MAPE as shown in Table 6.

Sum of $BMRE_1$, $BMRE_2$,…….$BMRE_{20}$ = **2.099**

Balanced MMRE (BMMRE) = 2.099/20 = 0.104951

Sum of percentage errors = **196.360**

Mean Absolute Percentage Error (MAPE) = 196.360/20 = 9.818023

Table 6. Predictive Accuracy Measures

Project No.	Predicted Reliability	Actual Reliability	MRE	BMRE	% Error
P1	0.832	0.9	0.076	0.082	7.556
P2	0.721	0.9	0.199	0.248	19.88
P3	0.912	0.9	0.013	0.013	1.333
P4	0.600	0.75	0.200	0.250	20.00
P5	0.750	0.75	0.000	0.000	0.000
P6	0.587	0.55	0.067	0.067	6.727
P7	0.750	0.75	0.000	0.000	0.000
P8	0.550	0.55	0.000	0.000	0.000
P9	0.586	0.55	0.065	0.065	6.545
P10	0.750	0.75	0.000	0.000	0.000
P11	0.629	0.55	0.144	0.144	14.36
P12	0.614	0.55	0.116	0.116	11.63
P13	0.565	0.55	0.027	0.027	2.727

P14	**0.752**	**0.75**	**0.003**	**0.003**	**0.267**
P15	**0.761**	**0.55**	**0.384**	**0.384**	**38.36**
P16	**0.320**	**0.35**	**0.086**	**0.094**	**8.571**
P17	**0.330**	**0.35**	**0.057**	**0.061**	**5.714**
P18	**0.350**	**0.35**	**0.000**	**0.000**	**0.000**
P19	**0.131**	**0.15**	**0.127**	**0.145**	**12.66**
P20	**0.210**	**0.15**	**0.400**	**0.400**	**40.00**

Like MMRE the values of BMMRE and MAPE are also comes out very promising, and reemphasizing that the model (ESRPM) has superior 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.

Median Magnitude of Relative Error (MdMRE) = 0.066

P_{25} (Ist Quartile) = 0.0000

P_{75} (IIIrd Quartile) = 0.135152

The values of MdMRE P_{25} and P_{75} are also quiet encouraging. Besides these the study has also computed the Pred(0.25), that reports the percentage of the estimates with an MRE less than or equal to 0.25. The following value of Pred(0.25) indicating that the 90% of the predicted DLR values by the ESRPM have MREs less than or equal to 0.25, that is once again quiet encouraging.

Pred (0.25) = 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.

7. QUANTITATIVE COMPARISON

The researcher has developed an 'Early Stage Reliability Prediction Model' (ESRPM) to implement the already proposed 'Fuzzy Logic based Software Reliability Quantification Framework' (^{FL}SRQF). Though, the newly developed model has been validated in section 5 and its prediction accuracy has also been ensured in section 6. It appears worthwhile to compare the proposed model 'ESRPM' with an existing model in order to show how it is better from earlier one. The second section of the paper presented the critical review of recent studies on software reliability prediction. Now this section has identified a reliability prediction study, on the basis of its relevance with the reliability prediction model (ESRPM), and is going to quantitatively compare it in terms of their quantified reliability values and correlation coefficient.

7.1 Comparison on Reliability Values

Kumar and Dhanda, [20] proposed a Reliability estimation model for object-oriented software in design phase. The model computes reliability in terms of effectiveness and functionality. Prior to develop reliability model, study had developed separate models for effectiveness as well as functionality. The equations of developed multivariate models are as follows:

Effectiveness = – 4.559 + 2.557 * (Encapsulation) + 0.738 * (Coupling) + 5.353 * (Inheritance)

Functionality = 1.656 + 1.141 * (Coupling) + 13.336 * (Cohesion) – 1.043 * (Inheritance)

Reliability = 1.384 – 0.284 * (Effectiveness) – 0.096 * (Functionality)

Table 7. Empirical Comparison (ESRPM *Vs* Kumar and Dhanda, [20])

S. No	Model of this Research	Model Developed in [20]		
	DLR	Reliability	Effectiveness	Functionality
1	**0.832**	**0.700**	-1.386	11.225
2	**0.721**	**0.430**	-0.176	10.461
3	**0.912**	**0.645**	-1.446	11.973
4	**0.600**	**0.275**	-0.121	11.907
5	**0.750**	**0.546**	-0.123	9.092
6	**0.587**	**0.106**	0.667	11.339
7	**0.750**	**0.554**	-1.441	12.906
8	**0.550**	**0.518**	0.026	8.947
9	**0.586**	**0.530**	-0.033	8.992
10	**0.750**	**0.119**	-0.412	14.391
11	**0.629**	**0.387**	0.891	7.747
12	**0.614**	**0.306**	-0.758	13.469
13	**0.565**	**0.520**	-0.065	9.189
14	**0.752**	**0.638**	-1.578	12.437
15	**0.761**	**0.487**	-1.121	12.656
16	**0.320**	**0.319**	0.990	8.170
17	**0.330**	**0.279**	2.122	5.232
18	**0.350**	**0.228**	1.287	8.232
19	**0.131**	**0.119**	0.846	10.677
20	**0.210**	**0.201**	1.827	6.920

Table 7 presents the reliability values at the design stage from both the models. Now observing the table values it can be easily inferred that the reliability model (ESRPM) developed in this research is predicting the software reliability of the developing software at the design stage quiet accurately than the model developed in [20].

Table 8. Comparison through Pearson's Correlation Coefficient

S. No.	Reliability Model	Pearson's Correlation Coefficient	Correlation Level
1	Kumar and Dhanda [20]	0.615	Moderate Positive
2	Proposed Model (ESRPM)	**0.936**	**High Positive**

7.2 Comparison on Correlation Coefficient

The researcher has computed the Pearson's Correlation Coefficient between the predicted values of reliability (through the proposed model and [20]) and the actual values of the reliability (from the dataset). Looking at the values of the following Table 8 it can be easily noticed that the developed model (ESRPM) in this research has a very High Positive Correlation, While, the model developed by Kumar and Dhanda, has Moderate Positive Correlation.

Therefore it can be conclude, on the basis of quantitative values, that the model (ESRPM) developed in this research is better than the existing reliability model.

8. CONCLUSIONS

Generally, reliability can be evaluated once the software product is finished or nearly finished. Therefore estimating reliability early in the development life cycle can help designers to incorporate required enhancement and corrections, by the coding phase begins. Such early estimation not only minimize future efforts, but also reduces unavoidable rework to be done on the software product. Requirements and Design stages of development life cycle play a very critical role in the development of reliable software. Therefore predicting reliability of developing software at the end of its design stage would help in predicting the reliability of the software to be delivered.

To accomplish this task, researcher has performed a comprehensive literature review, and proposed a fuzzy based framework for predicting the reliability, as a solution to the identified shortcomings during the review. Further, in order to implement the framework, this research has developed an Early Stage Reliability Prediction Model (ESRPM) which predicts reliability before the coding phase. Subsequently, the developed model has been validated as well as compared quantitatively with two existing reliability models. The results are quiet encouraging and support the claim that the developed model (ESRPM) has improved the reliability modeling quite efficiently in the early stage of software development.

Following points reiterate the significance of the study:

a) Consideration of the requirements phase along with the design provides this research an edge over other studies those are based on only design phase, because ignoring or overlooking requirements deficiencies and only concentrating on making the design constructs superior will not seems good enough.
b) The suitability of various requirement and design measures as a contributor for the software reliability has been identified.
c) The developed reliability model may help software professionals to take appropriate corrective measures right from its requirements phase, to deliver software with an improved reliability level, close to the user's expectation.
d) Based on the analysis of quantified values, the research assists developers by providing them an opportunity, to once again improve requirements and design related internal characteristics ahead of writing the final code.
e) The proposed model may help designers as well as developers to predict the reliability of the developing software upto its design stage, early in the development life cycle.
f) Based on the predicted reliability of developing software upto its design stage, the developers may predict the reliability of the final software to be delivered in future.
g) In order to overcome the limitations of subjective values of requirements metrics, the research has utilized the strength of fuzzy inference process in its quantification phase.
h) In most of the cases, developed models only provide quantitative values but neither provides suggestions on how to make improvement, nor the precautions on how to avoid abnormalities. Therefore, to fill this gap this study has provided the suggestive measures along with the recommendations based on the results and contextual interpretations.
i) Apart from the above, reassessment of previously developed or underdevelopment reliability prediction models could be done as per the guidance proposed and recommended in this study.
j) Beside this, as far as further research is concern, the model may open fresh avenues for the researchers, doing research on reliability estimation as well as reliability prediction.

Future Directions suggested by the researcher are as follows:

a) As this study assumes that the SRSs are written in plain English text. Therefore one possible future direction may be developing some automated approach that could be helpful in identifying the inconsistencies, ambiguities and incompleteness in the SRS.
b) The study has considered only four object-oriented design constructs to quantify the reliability at the design stage. (i.e. Encapsulation, Coupling, Inheritance and Cohesion). Therefore it further opens more avenues to consider some other object-oriented or non object-oriented design constructs, to move ahead in the direction of reliability prediction.
c) The reliability prediction model developed in present study focuses on object-oriented paradigm, but in future more generalized reliability prediction model can be developed.
d) The future research may also focus on measuring other quality factors like those proposed in the ISO 9126.
e) Besides these, there are also several directions that reliability prediction can take in future includes (i) the use of automated reliability prediction models as acceptance criteria for software quality control and third party software procurement; (ii) the analysis of how software reliability affects other software quality factors such as maintainability, re-usability and portability.

9. REFERENCES

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

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

[3] Okutan, and Yildiz, O.T. 2014. Software Defect Prediction using Bayesian Networks. Empirical Software Engineering, 19(1), 154-181.

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

[5] 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.

[6] 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.

[7] Bowles, J.B., and Pelaez, C.E. 1995. Application of fuzzy logic to reliability engineering. Proceedings of IEEE, 83(3), 435–449.

[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] Ying, M., Shunzhi, Z., Ke, Q., and Guangchun, L. 2014. Combining the requirement information for software defect estimation in design time. Information Processing Letters, 114(9), 469–474.

[10] Catal, C. 2011. Software Fault Prediction:A Literature Review and Current Trends. Expert System with Applications, 38(4), 4626-4636.

[11] Catal, C., and Diri, B. 2009. A Systematic Review of Software Fault Predictions Studies. Expert System with Applications, 36(4), 7346-7354.

[12] 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.

[13] Mizuno, O. and Hata, H. 2009. Yet another Metric for Predicting Fault-Prone Modules. Advances in Software Engineering Communications in Computer and Information Science, Springer, 59, 296-304.

[14] He, Z., Shu, F., Yang,Y., Li, M., and Wang, Q. 2012. An Investigation on the Feasibility of Cross-Project Defect Prediction. Journal of Automated Software Engineering, 19(2), 167-199.

[15] Maa, Y., Zhua, S., Qin, K. and Luo, G. 2014. Combining the Requirement Information for Software Defect Estimation in Design Time. Information Processing Letters, 114(9), 469-474.

[16] 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.

[17] 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.

[18] 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.

[19] Jaiswal, G.P. and Giri, R. N. 2015. A Fuzzy Inference Model for Reliability Estimation of Component Based Software System. International Journal of Computer Science and Technology, 3(3), 177-182.

[20] Kumar, A. and Dhanda, N. 2015. Reliability Estimation of Object-oriented Software: Design Phase Perspective. International Journal of Advanced Research in Computer and Communication Engineering, 4(3), 573-577.

[21] Yadav, A., and Khan, R. A. 2012a. Reliability Quantification of an Object-Oriented Design: Complexity Perspective. Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012), May 25-27, 2012, New Delhi, Advances in Intelligent and Soft Computing, Springer, 166, 577-585.

[22] Kong, W. 2009. Towards a Formal and Scalable Approach for Quantifying Software Reliability at Early Development Stages. Ph.D. Thesis University of Maryland.

[23] Hooshmand, A. and Isazadeh, A. 2008. Software Reliability Assessment Based on a Formal Requirements Specification, Proceedings of the Conference on Human System Interactions, Publisher IEEE Krakow, Poland, 311-316.

[24] 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.

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

[26] McCall, J.A., Richards, P.K., Walters, G.F. 1977. Factors in software quality, RADC (Rome: Rome Air Development Center), TR-77-369.

[27] Dromey, R.G. 1996. Concerning the Chimera. IEEE Software. 1, 33-43.

[28] Boehm, B.W. 1987. Improving Software Productivity. IEEE Computer, 20(9), 43-57.

[29] ISO, 2001. ISO/IEC 9126-1: Software Engineering-Product Quality –Part I: Quality Model. Geneva, Switzerland.

[30] 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.

[31] 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.

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

[33] 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.

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

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

[36] 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.

[37] 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.

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

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

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

[41] 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.

[42] 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.

[43] Yadav, A. and Khan, R.A. 2009b. Measuring Design Complexity–An Inherited Method Perspective. ACM Software Engineering Notes, 34(4), 1-5.

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

[45] 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(1), 63–74.

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

[47] Zhang, X. and Pham, H. 2000. An analysis of factors affecting software reliability. Journal of Systems and Software, 50(1), 43–56.

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

[49] 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.

[50] 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.

[51] Kitchenham, B.A., Pickard, L.M., MacDonell, S.G. and Shepperd, M.J. 2001. What Accuracy Statistics Really Measure. IEEE Software, 148(3), 81–85.