

## Early Software Defects Prediction Using Fuzzy Logic

DILIP KUMAR YADAV, S. K. CHATURVEDI\* and RAVINDRA B. MISRA

*Reliability Engineering Centre, Indian Institute of Technology Kharagpur,  
Kharagpur-721302 (West Bengal) India*

*(Received on April 20, 2011, revised on May 22, 2012)*

**Abstract:** This paper presents a model to predict the number of residual defects before testing phase. The developers need this information a priori for optimal testing resource planning and quality assessment of the software being developed. In the early stages of software development life cycle (SDLC), software residual defects are affected by both product and process characteristics of the project. These product and process characteristics are embedded in software metrics which have subjective assessments in the early stages of SDLC. Therefore, software metrics are considered for developing the model for early software defects prediction. This paper uses the software size metric and three metrics of requirement analysis phase for predicting the residual defects that are likely to be found during testing or operational usage using fuzzy logic. The predictive capability of the proposed approach is examined using qualitative data of requirement metrics of twenty real software projects and results are compared with the existing model.

**Keywords:** *software metrics, fuzzy logic, early software defects prediction, software reliability, qualitative data*

### 1. Introduction

The reliability issues in software based systems are a well-known fact. Most of the modern systems work under direct or indirect influence of software. Hence, the reliability prediction of software is of great importance. An accurate estimate of software reliability can be obtained using software reliability models only in the later phase of software testing. However, with the aim of cost-effectiveness and timely management of resources, the reliability prediction of software in the early phases of software development life-cycle is one of major area of concern. Here, the term 'early' is being used to represents the early phases of the life-cycle of a software, viz., requirement analysis, design, and implementation phases. Residual defects of software are one of the major indicators for software reliability and quality. Therefore, early software defects prediction provides early identification of cost overruns, software development process issues, optimal development strategies, and a reliable forecast of the end quality of the software being developed in terms of product suitability.

In the early phases of SDLC, residual defects of software system may be predicted using product and process characteristics that are embedded in software metrics. Therefore, software metrics are very helpful for predicting the residual defects of software system during early phases of SDLC. Traditional models on early software reliability prediction [1-5] are organization specific and not flexible. There exist very few models [6

---

\*Corresponding author's email: skrec@hijli.iitkgp.ernet.in

-10] to predict the residual defects of software before testing based on software engineering metrics. Khoshgoftaar and Munson [6] mentioned that there is a relationship between program errors and complexity domains of program structure and size. Fenton and Neil [7] also mentioned that most of the software defects prediction models use size and complexity metrics to predict residual defects. In recent times, Jiang *et al.* [8] shows that requirement metrics are highly useful for defect prediction. Fenton *et al.* [9] suggested and explained how the subjective assessment of a metric can be performed through a questionnaire. They also mentioned that good predictions of the software residual defects can be achieved by entering relatively few of the project factors, *e.g.*, software size metric and 2 or 3 other metrics. Fenton *et al.* [10] proposed a model for early life cycle defect prediction with Bayesian Nets by developing a causal model for predicting the number of residual defects that are likely to be found during independent testing or operational usage. They applied their model by analyzing several evaluation measures on a dataset, elicited from 31 completed software projects in the consumer electronics industry, which was gathered using a questionnaire distributed to managers of projects. Their validation results also confirm the need for using the qualitative factors in the model. However, these models developed for early software defects prediction are inadequate to provide trustworthy results due to presence of vague and imprecise data.

The fuzzy set theory provides a way to capture the uncertainty, vagueness and imprecision present in the information and also attempts to capture the way we express our belief. Therefore, in early phases of SDLC, where the data is scarce or is present in form of 'knowledge', a model using fuzzy logic would be more appropriate, which can be adopted for the prediction of software residual defects. Therefore, a fuzzy logic based model is proposed in this paper for early software defects prediction using software size metric and three metrics of requirement analysis phase. Software size metric is used to define range of the output of the proposed fuzzy model, and requirements metrics are used as input to the fuzzy model. In proposed model, only that software requirements metrics have been considered which is measurable in the requirement analysis phase. However, the availability of data in the requirements analysis phase is of subjective (imprecise /vague) nature. Therefore it becomes necessary to rely on expert opinion for the assessment of software metrics of requirement phase.

The rest of the paper is arranged as follows. The proposed model is explained in section II. Section III describes the case studies for twenty real software projects. The model validation is done in section IV. Section V presents the conclusion of this work.

## 2. Proposed Model

Software metrics, contributing directly to the reliability of the software at requirements analysis phase, are considered as input to the proposed fuzzy model. Li and Smidts [15] selected thirty software metrics which influence software reliability. These software metrics were ranked with respect to their ability in predicting software reliability through an expert opinion elicitation process. 12 out of 30 software metrics are available in the requirements analysis phase in which the top four metrics are (i) requirement defect density, (ii) requirement stability, (iii) error distribution, and (iv) reviews, inspections and walkthroughs. Metrics (iii) and (iv) depend on the experience of requirement team members. Therefore, these three metrics of requirements analysis phase *i.e.*, *Experience of Requirement Team* (ERT), *Requirement Defect Density* (RDD), and *Requirement Stability*

(RS) are considered as input to the proposed fuzzy model. ERT metric measures the relevant experience and skill set of team members for executing the project during the requirements analysis phase of SDLC. RDD metric measures the percentage of defective requirements in the requirements specification documents which is obtained through regular reviews. Requirement stability (RS) metric measures the stability of the requirements in the projects. Requirements specification change request may come by the client during the different phases of software development life-cycle. If it comes in the early phase then the requirements stability is on the high side, else it is on the low side.

Besides these three requirements analysis phase metrics, software size metric is also considered for the model development. It helps in developing the fuzzy profile of output metric of the proposed fuzzy model. In fact, residual defects of software are very much affected by the intrinsic complexity of software and software size metric can characterize the intrinsic complexity of the software. Software size metric (in KLOC) measures program length which in turn can be termed as the intrinsic complexity of the software.

## 2.1 Model Architecture

The architecture of the proposed model is shown in the Fig. 1. Three metrics of requirements analysis phase are used as inputs to the fuzzy rule-base inference system, which gives an aggregated fuzzy set of total number of residual defects as output. The total number of residual defects obtained through defuzzification further helps deciding the testing strategy during testing phase of the software.

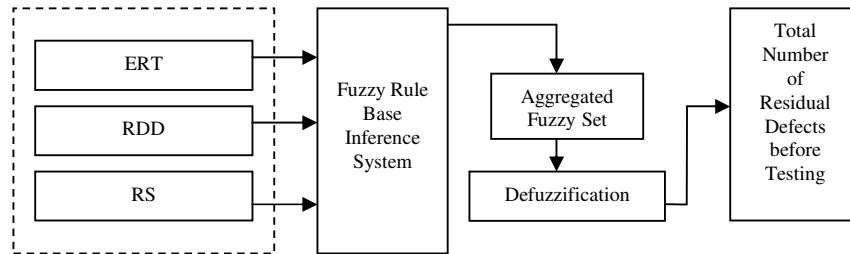


Figure 1: Architecture of the Proposed Model

## 2.2 Model Details

The general steps to apply this methodology are as follow:

- Step 1 Define the membership function for each input and output metrics.
- Step 2 Design the fuzzy rules and collect the expert's opinion for the consequent part of each rule.
- Step 3 Aggregate all the individual fuzzy sets (consequent parts) for various rules and find out a crisp value by defuzzifying the aggregated fuzzy set.

Each steps of the proposed methodology is explained in the following paragraphs.

### 2.2.1 Defining Membership Function of the Input and Output Metrics

There are many ways to assign membership functions to fuzzy variables [16-19]. This assignment process can be intuitive or it can be based on some algorithmic or logical operations. In the *intuition* method, membership function is derived from the capacity of humans to develop membership functions through their own innate intelligence and

understanding. In the *inference* method of membership function development, one uses knowledge to perform deductive reasoning. Assessing preference by a single individual, a committee, a poll, and other opinion methods can be used to assign membership values to a fuzzy variable in the *rank ordering* method of membership function development. Membership functions for all the input and output metrics which are considered in the proposed model should be defined by domain experts. In the proposed model, triangular and trapezoidal membership functions are considered for representing the linguistic states (low, medium, high) of input metrics, and triangular membership function is considered for representing the linguistic states (very low, low, medium, high, very high) of output metric of the fuzzy model. The maximum value of abscissa (range) for all input metrics is taken 1, but it may also be decided by domain expert. For the output metric (*e.g.*, total number of defects), maximum value of abscissa (range) is calculated using (1) [11],

$$D = 4.2 + 0.0015(L)^{4/3} \quad (1)$$

where,  $D$  is the approximate total number of defects and  $L$  is total number of lines of code (LOC). Optimal module size has been advised 877 LOC. Hence, the maximum range of the output metric is calculated as:

Approximate total number of defects = ((Project size in KLOC)/0.877)\*17

It gives an approximation to the maximum range of output metric to make output fuzzy profile to be used in inference system. However, historical data of similar projects is beneficial for its verification, which may modify it by the maximum value of defects found in the similar projects executed already.

### 2.2.2 Design of Fuzzy Rules

Fuzzy rules are the constituents of the knowledge based system. The AND rules of the IF\_THEN form have been used in the proposed model. IF part of the rules are the Antecedents, whereas the THEN part are the Consequents. In our proposed model, the antecedents are the software input metrics and the consequents are expected total number of defects before testing. For each rule, value of the consequent is assessed by the domain expert.

In our model, there are three input metrics. Each input metric has three linguistic states *i.e.*, low (L), medium (M) and high (H). Therefore, total number of rules is 27. These rules are as follows:

1. If ERT is L and RDD is L and RS is L then Expected total number of defects is H
2. If ERT is L and RDD is L and RS is M then Expected total number of defects is M
- ...
26. If ERT is H and RDD is H and RS is M then Expected total number of defects is H
27. If ERT is H and RDD is H and RS is H then Expected total number of defects is M

### 2.2.3 Fuzzy Inference, Aggregation of the Rules and Defuzzification

A fuzzy *Max-Min* operator is used for this step. The minimum of the degrees of membership will result in a resultant fuzzy set for a particular rule. The aggregated fuzzy set is the summation of all the individual fuzzy sets for various rules. Finally, defuzzification of the aggregated fuzzy set provides the final result as a crisp value.

### 3. Case Studies

#### 3.1 Projects Data Sets for Case Studies

Twenty real software project data sets are taken from [9] for case studies and are reproduced in Table 1 for the sake of completeness.

**Table 1:** Assessment of Software Projects

Project #	Software Project [ 9 ]	KLoC	ERT	RDD	RS
1	1	6	H	H	L
2	2	0.9	H	H	H
3	3	53.9	H	H	H
4	7	21	L	L	M
5	8	5.8	M	L	H
6	9	2.5	H	M	H
7	10	4.8	H	M	H
8	11	4.4	H	H	H
9	12	19	H	M	L
10	13	49.1	H	H	L
11	15	154	H	H	L
12	16	26.7	H	H	M
13	17	33	H	H	M
14	19	87	H	H	M
15	20	50	L	M	L
16	21	22	H	M	M
17	22	44	L	M	L
18	24	99	M	H	L
19	29	11	H	M	H
20	30	1	H	M	H

#### 3.2 Model Illustration: Case Study 1

Software project#1 is considered to explain the approach proposed in this paper. Following are the steps for finding the total number of residual defects for project # 1 (Table 1):

Step 1 Membership function for each input and output metrics are shown in Fig. 2, Fig. 3, Fig. 4, and Fig. 5, respectively. Note that the metrics have been expressed in their normalized form. The size metric of project#1 is 6 KLOC. Therefore,

$$\text{Maximum value of output metric} = \left( \frac{6}{0.877} \right) \times 17 = 117$$

Had similar projects for project#1 available, the maximum value of the output metric would have been set to the maximum number of defects detected in the similar projects. Since the information of similar project(s) is not available, therefore, maximum value of the output metric for project is taken 117.

Step 2 Out of 27 rules formulated, some of the AND fuzzy rules are as follows:

1. If ERT is L and RDD is L and RS is L then Expected total number of defects is H
2. If ERT is L and RDD is L and RS is M then Expected total number of defects is M
3. If ERT is L and RDD is L and RS is H then Expected total number of defects is L

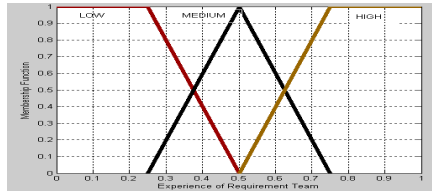


Figure 2: Experience of Requirement Team

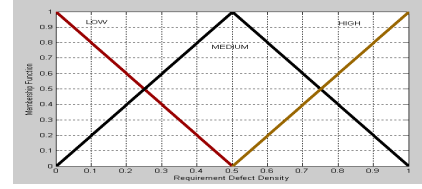


Figure 3: Requirements Defect Density

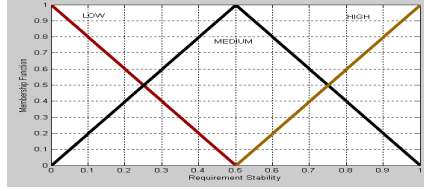


Figure 4: Requirements Stability

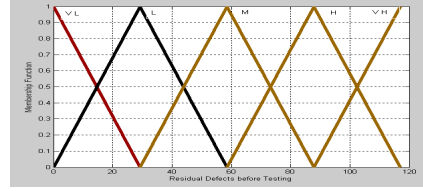


Figure 5: Total Number of Residual Defects

25. If ERT is H and RDD is H and RS is L then Expected total number of defects is VH
26. If ERT is H and RDD is H and RS is M then Expected total number of defects is H
27. If ERT is H and RDD is H and RS is H then Expected total number of defects is M

Step 3 Using the Matlab® fuzzy tool box, the aggregated output fuzzy set is shown in Fig. 6. The centroid defuzzification provides a total number of residual defects as 88.

### 3.3 Results of Prediction

The total numbers of software defects predicted by using the proposed model and recently proposed model by Fenton *et al.* [10] for the 20 projects are provided in the Table 2. The results have been ordered as per the accuracy provide by the proposed model. It is evident that the model performs better than the model proposed method by Fenton [10]. Fig. 7 shows the graphical view of the results presented in Table 2. It can seen from Fig. 7 that the prediction accuracy of the proposed model is much better than the Fenton *et al.* [10] model for twenty software projects of varying KLOC from 0.9 to 154.

## 4. Model Validation

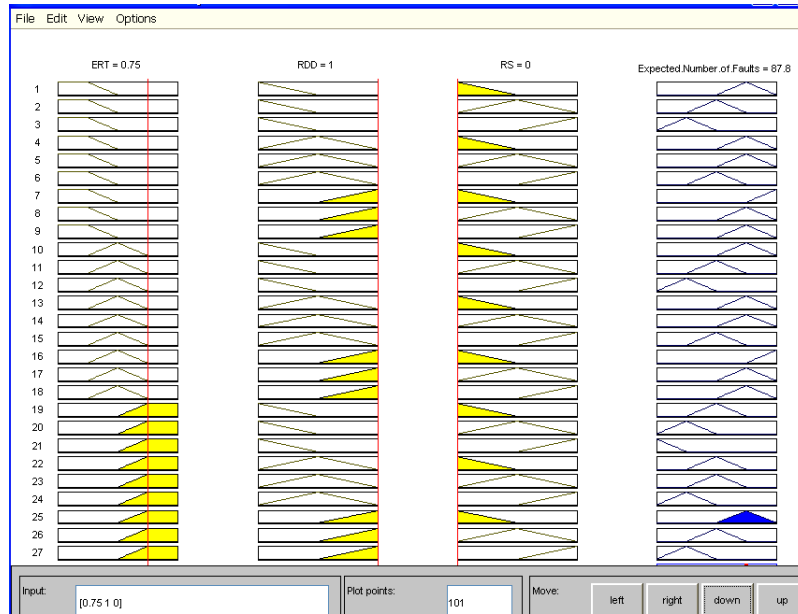
### 4.1 Evaluation Measures

To validate the proposed approach following, commonly used and suggested evaluation [10, 12-14], measures have been taken:

1. **Mean Magnitude of Relative Error (MMRE):** MMRE is the mean of absolute percentage errors and a measure of the spread of the variable  $z$ , where  $z = \text{estimate} / \text{actual}$ .

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i},$$

where  $y_i$  is the actual value and  $\hat{y}_i$  is the estimated value of a variable of interest.



**Figure 6:** Aggregation of Fuzzy Rules and the Defuzzification for Project 1

**Table 2:** Actual and predicted number of defects before testing

SL. No.	Project # [9]	Actual defects	Defects predicted by	
			Fenton <i>et al.</i> [10]	Proposed Model
1	7	204	262	204
2	30	5	46	5
3	12	90	347	92
4	20	928	986	892
5	8	53	48	56
6	21	196	259	214
7	24	1597	1514	1440
8	11	71	51	64
9	19	476	581	422
10	17	688	444	589
11	15	1768	1526	1490
12	22	184	501	213
13	29	91	116	107
14	16	109	145	130
15	3	209	254	261
16	1	148	75	88
17	9	17	57	24
18	2	31	52	9
19	10	29	203	70
20	13	129	516	476

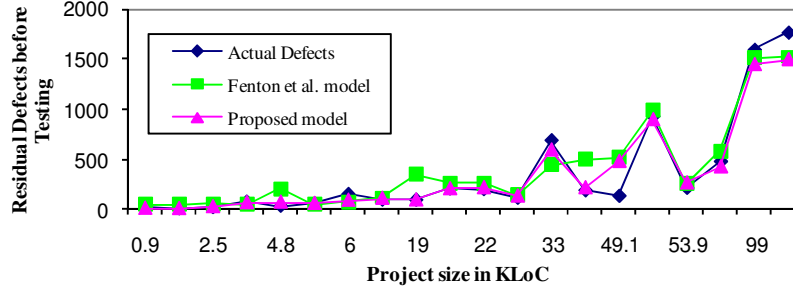


Figure 7: Number of Residual Defects before Testing for 20 Projects

## 5. Model Validation

### 5.1 Evaluation Measures

To validate the proposed approach following, commonly used and suggested evaluation [10, 12-14], measures have been taken:

1. **Mean Magnitude of Relative Error (MMRE):** MMRE is the mean of absolute percentage errors and a measure of the spread of the variable  $z$ , where  $z = \text{estimate} / \text{actual}$ .

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i}$$

where  $y_i$  is the actual value and  $\hat{y}_i$  is the estimated value of a variable of interest.

2. **Balanced Mean Magnitude of Relative Error (BMMRE):** MMRE is unbalanced and penalizes overestimates more than underestimates. For this reason, a balanced mean magnitude of relative error measure is also considered which is as follows:

$$\text{BMMRE} = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{\text{Min}(y_i, \hat{y}_i)}$$

The lesser value of MMRE and BMMRE indicates better accuracy of prediction.

### 5.2 Validation Results

Fenton *et al.* [10] predicted the software defects for these 20 projects by employing Bayesian nets and early software development life-cycle process metrics. The comparisons of these measures have been given in Table 3. It can be observed in Table 3 that the MMRE and BMMRE for the proposed model are 0.3613 and 0.4185, respectively whereas for Fenton *et al.* [10] model their value are 1.3965 and 1.4375, respectively. Clearly, the MMRE and BMMRE of the proposed model come out to be much lesser than that of the Fenton *et al.* [10] model. It can also be observed that the predictive accuracy of the model expressed by different measures increases with the size of the project for both models. The measures based on relative error (MMRE, BMMRE) decrease significantly, as project size increases for both models.



**Table 3:** Values of model evaluation measures

Project Size	MMRE		BMMRE	
	Fenton <i>et al.</i> [10]	Proposed	Fenton <i>et al.</i> [10]	Proposed
Projects < 5 KLoC ( $n=5$ )	3.5024	<b>0.5268</b>	3.5245	<b>0.6862</b>
5KLoC $\leq$ Projects < 50 KLoC ( $n=10$ )	0.9731	<b>0.3936</b>	1.0416	<b>0.4237</b>
Projects $\geq$ 50KLoC ( $n=5$ )	0.1375	<b>0.1313</b>	0.1424	<b>0.1404</b>
All projects ( $n=20$ )	1.3965	<b>0.3613</b>	1.4375	<b>0.4185</b>

$n$  is the number of projects

## 6. Conclusions

At the early stage of software development life cycle, the software metrics are difficult to measure due to their associated uncertainties. These uncertainties in the inputs (measured qualitatively or quantitatively) could be dealt by the use of fuzzy set theory. In this paper, a fuzzy logic based early software defects prediction model has been proposed, which would be very useful for early software defects prediction where sufficient objective data are not generally available. The model takes into account the uncertainty associated over the assessment of software size metric and three metrics of requirement analysis phase, *viz.*, ERT, RDD and RS, to predict the expected number of total defects in the software. The model has employed both quantitative and qualitative measures of metrics for defects prediction using fuzzy approach thereby making it different from the other defect prediction models by explicitly combining these factors. Furthermore, 20 real software project data sets have been employed to illustrate the applicability and usability of the proposed approach. Through comparison with the recently proposed method by Fenton *et al.* [10] in 2008, it has been shown that the prediction accuracy of the proposed model is much better.

## References

- [1]. McCall, J. A., W. Randell, and J. Dunham. *Software Reliability, Measurement, and Testing*. Rome Laboratory 1992; RL-TR-92-52
- [2]. Agresti W. W., and W. M. Evancho. *Projecting Software Defects from Analyzing Ada Designs*. IEEE Transactions on Software Engineering 1992; 18 (11): 988-997
- [3]. Smidts C., M. Stutzke, and R. W. Stoddard. *Software Reliability Modeling: An Approach to Early Reliability Prediction*. IEEE Transactions on Reliability 1998; 47(3): 268-278
- [4]. Yin M-L, E. J. Lawrence, S. Keene, R. R. Arellano, J. Peterson, and C. A. Fullerton. *Adaptive Software Reliability Prediction*. Proceedings of the Raytheon Systems Company Software Technology Symposium 1999 Sep. 21-23
- [5]. Cortellessa V., H. Singh, and B. Cukic. *Early Reliability Assessment of UML based Software Models*. Workshop on Software and Performance 2002 : 302-309
- [6]. Khoshgoftaar T. M., and J. C. Munson. *Predicting Software Development Errors Using Software Complexity Metrics*. IEEE Journal on Selected Areas in Communications 1990; 8(2) :253-261
- [7]. Fenton N. E., and M. Neil. *A Critique of Software Defect Prediction Models*. IEEE Transaction on Software Engineering 1999; 25(5) : 675-689
- [8]. Jiang Y., B. Cukic, and T. Menzies. *Fault Prediction using Early Lifecycle Data*. Proceedings of 18th IEEE International Symposium on Software Reliability Engineering (ISSRE) 2007: 237-246
- [9]. Fenton N., M. Neil M., W. Marsh, and P. Hearty, L. Radlinski, and P. Krause. *Project Data Incorporating Qualitative Factors for Improved Software Defect Prediction*.

- Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering 2007
- [10]. Fenton N., M. Neil, W. Marsh, and P. Hearty, L. Radliński, and P. Krause. *On the Effectiveness of Early Life Cycle Defect Prediction with Bayesian Nets*. Empirical Software Engineering 2008; 13(5): 499-537
  - [11]. Gaffney J. E. (Jr.), *Estimating the Number of Faults in Code*. IEEE Trans. Software Eng. 1984; 10(4):141-152
  - [12]. Chulani S., B. Boehm, and B. Steece. *Bayesian Analysis of Empirical Software Engineering Cost Models*. IEEE Trans Software Engineering 1999; 25(4): 573-583
  - [13]. Kitchenham B. A., L. M. Pickard, S. G. MacDonell, and M. J. Shepperd. *What Accuracy Statistics Really Measure*. IEE Proc. Software 2001; 148(3): 81-85
  - [14]. Stensrud E., T. Foss, B. Kitchenham, and I. Myrtveit. *An Empirical Validation of the Relationship between the Magnitude of Relative Error and Project Size*. Proc. of 8th IEEE Symposium on Software Metrics 2002: 3-12
  - [15]. Li M, and C. Smidts. *A Ranking of Software Engineering Measures Based on Expert Opinion*. IEEE Transaction on Software Engineering 2003; 29(9): 811-824
  - [16]. Misra K. B., and G. G. Weber. *Use of Fuzzy Set Theory for Level-I Studies in Probabilistic Risk Assessment*. Fuzzy Sets and Systems 1990; 37(2): 139-160
  - [17]. Verma A. K., A. Srividya, and Gaonkar R. S. P. *Fuzzy Reliability Engineering: Concepts and Applications*. Narosa Publishing House, Delhi, India, 2007
  - [18]. Maisseu A., and Xingquan W. *Fuzzy Parameterization of Bulgaria's Nuclear Policy Decision*. International Journal of Performability Engineering 2005; 1(2): 167-178
  - [19]. Ross T. J., *Fuzzy Logic with Engineering Applications*, John Wiley publications, 2nd Edition, 2004

**Dilip Kumar Yadav** is a research scholar (QIP) in the Reliability Engineering Centre, Indian Institute of Technology Kharagpur (West Bengal), India. His current research interests include software reliability, software safety, fuzzy systems, and software metrics.

**S. K. Chaturvedi** is currently working as an Associate Professor at Reliability Engineering Centre, Indian Institute of Technology, Kharagpur (W.B.) India. He received his B. E. (Electrical, 1988), M. E. (SEOR, 1990) from IIT Roorkee and Ph. D. from Reliability Engineering Centre, IIT-Kharagpur, in year 2003. He has research interest in the area of Reliability Modeling and Prediction, FMECA, FTA, network reliability, life-data analysis, and optimization. He has published papers in several international journals such as IEEE, Performability Engineering, Quality and Reliability Management, Quality, Reliability and Safety Engineering, Uncertain Systems, Quality Technology and Quantitative Management *etc.* He is an editorial board member to IJPE and also reviewer for IEEE, Int J of System Science, Int J of Failure Analysis, Defense Science Journal and IJQRM.

**Ravindra B. Misra** received his B.Eng. and M. Tech. degrees in electrical engineering and power systems respectively and received his Ph.D. degree in reliability engineering from IIT, Roorkee, in 1980. He carried out Post Doctoral research at University of Ottawa, Canada. He has been a professor at IIT Kharagpur, India and has published more than 150 papers on system reliability, software reliability and probabilistic power system planning. He has successfully completed many R & D and consultancy projects in these areas. He is a reliability consultant to private as well as public sector industries in India. He is a fellow of Institution of Engineers, India and senior member of IEEE. He is member of editorial board of Journal of Mathematics & System Sciences. His research interests include reliability modeling of engineering systems, reliability testing and demonstration, power system reliability, software reliability and software safety.