Fault Prediction Model by Fuzzy Profile Development of Reliability Relevant Software Metrics

Ajeet Kumar Pandey
Reliability Engineering Centre
IIT Kharagpur, Kharagpur

Neeraj Kumar Goyal
Reliability Engineering Centre
IIT Kharagpur, Kharagpur

ABSTRACT
This paper presents a fault prediction model using reliability relevant software metrics and fuzzy inference system. For this a new approach is discussed to develop fuzzy profile of software metrics which are more relevant for software fault prediction. The proposed model predicts the fault density at the end of each phase of software development using relevant software metrics. On the basis of fault density at the end of testing phase, total number of faults in the software is predicted. The model seems to useful for both software engineer as well as project manager to optimally allocate resources and achieve more reliable software within the time and cost constraints. To validate the prediction accuracy, the model results are validated using PROMISE Software Engineering Repository Data set.

Keywords
Reliability Relevant Software Metrics, Software Fault Prediction, Fault Density, Fuzzy profile, Fuzzy Inference System (FIS)

1. INTRODUCTION
Software has become so essential to human in their daily lives that today it is difficult to imagine living without devices controlled by software. Software reliability and quality has become the primary concern during the software development. It is difficult to produce fault-free software due to the problem complexity, complexity of human behaviors, and the resource constrains. Failure is an unavoidable phenomenon in all technological products and systems. System failures due to the software failure are very common and results undesirables consequences which can adversely affect both reliability and safety of the system. Now a day's most of the software development activity is performed in labor-intensive way. This may introduce various faults across the development, causing failures in near future. Therefore, there is a growing need ensure the reliability of these software systems by fixing the faults as early as possible. Moreover, it is well known that earlier an error is identified, the better and more cost effectively it can be fixed. Therefore, there is a need to predict these software faults across the stages of software development process.

IEEE defines software reliability as "the probability of a software system or component to perform its intended function under the specified operating conditions over the specified period of time" [1]. In other words it can also be defined as "the probability of failure-free software operation for a specified period of time in a specified environment". Software reliability is generally accepted as the key factor of software quality since it quantifies software failures that make the system inoperative or risky [2]. A software failure is defined as "the departure of external result of program operation from requirements", whereas a fault is defined as "the defect in the program that, when executed under particular conditions, causes a failure" [3]. To further elaborate, a software fault is a defective, missing, or extra instruction or set of related instructions that may cause of one or more actual or potential failures when executed.

Software reliability has roots in each step of the software development process and can be improved by inspection and review of these steps [4]. Generally, the faults are introduced in each phase of software life cycle and keep on propagating to the subsequent phases unless they are detected through testing or review process. Finally, undetected and/or uncorrected faults are delivered with software, causing failures. In order to achieve high software reliability, the number of faults in delivered code should be at minimum level.

Software reliability can be estimated or predicted through various software reliability models [3, 5]. These models uses failure data collected during testing and/or field operation to estimate or predict the reliability. This becomes late and sometimes infeasible for taking corrective actions. One solution to this problem may be predicting the faults across the stages of development process so that appropriate actions can be taken to mitigate or prevent these faults. Moreover, since the failure data are not available during the early phases of software life cycle, we have to dependent on the information such as reliability relevant software metrics (RRSM), expert opinions and similar or earlier project data. This paper proposes a multistage fault prediction model using reliability relevant software metrics (RRSM) that predicts the total number of faults at the end of each phase of software life cycle.

The remainder of this paper is organized as follows: Section 2 presents literature survey of the problem. Section 3 describes the proposed model. Section 4 provides implementation of model using fuzzy inference system. Section 5 contains the case studies and results whereas conclusions are presented in Section 6.

2. RELATED WORK
A lot of efforts have been made for software reliability prediction and assessment using various models [3, 5]. Gaffney and Davis [6, 7] of the Software Productivity Consortium developed a phase-based model, which makes the use of fault statistics obtained during the technical review of requirements, design, and the coding to predict reliability.

One of the earliest and well known efforts to predict software reliability during the early phase was the work initiated by the Air Force's Rome Laboratory [8]. They developed a prediction

model of fault density that can be transformed into failure rates. To do this the researchers selected a number of factors that they felt could be related to fault density at the earlier phases. In a similar study, Agresti and Evanco [9] presented a model to predict defect density based on the product and process characteristics for Ada program. Moreover, there are many papers advocating statistical models and software metrics [10, 11]. Most of them are based on size and complexity metrics.

A study was conducted by Zhang and Pham [12] to find the factors affecting software reliability. The study found thirty two potential factors involved in various stages of the software life cycle. In another study conducted by Li and Smidt [13], thirty reliability relevant software engineering measures have been identified. They have developed a set of ranking criteria and their levels for various reliability relevant software metrics, present in the first four phases of software life cycle. Kumar and Misra [14] made an effort for early software reliability prediction considering the six top ranked measures given by [13] and software operational profile. Sometimes, it may happen that some of these top ranked measures are not available, making the prediction unrealistic. Also they have considered only product metrics and ignored process metrics that influence software reliability. Recently, Pandey and Goyal [15] have developed an early fault prediction model using process maturity and software metrics. In their model they have developed the fuzzy profiles of various metrics in different scale and have not explained the criteria used for developing these fuzzy profiles.

Following are the general observations from the literature review:

- Software reliability is a function of number of faults present in the software.
- Software metrics plays a vital role in fault prediction in the absence of failure data.
- Early phase software metrics are of fuzzy nature.
- Software metrics follow either linear or logarithmic nature.

From the review of literature, we found that earlier fault prediction models have not considered these issues altogether. Also, no study has discussed the criteria for developing the fuzzy profile of software metrics. Keeping these points in mind, a fault prediction model is proposed which is able to predict the total number of faults present in the software, by systematic development of fuzzy profiles of software metrics on the basis of their nature and fuzzy inference system.

3. PROPOSED MODEL

Prediction of faults is desirable for any industry and attracts both engineers as well as managements. For software industry it provides an opportunity for the early identification of software quality, cost overrun and optimal development strategies. During earlier phases of software development; predicting the number of faults can reduce the efforts for additional reviews and more extensive testing [7].

The model architecture is shown in Figure1. Stages present in the proposed structure are similar to the waterfall model, a well known software development process model. It divides the structure into four consecutive phase I, II, III, and IV i.e. requirement, design, coding, and testing phase respectively. Phase-I predicts the fault density at the end of requirement phase using relevant requirement metrics. Phase-II predicts the fault density at the end of design phase using design metrics as well as output of the requirements phase. Similarly at phase-III besides the using coding metrics, output of phase-II is also considered as input to predict the fault density at the end of coding phase. Finally, the phase-IV predicts the fault density using testing metrics as well as output of the coding phase.

The proposed model considers three requirements metrics (RM): a) requirements complexity (RC), b) requirements stability (RS), and c) review, inspection and walk-through (RIW) as input. Similarly, at design phase two design metrics (DM): a) design team experience (DTE) and b) process maturity (PM) are considered as input. Two coding phase metrics (CM): a) coding team experience (CTE) and b) defined process followed (DPF) are taken as input. Finally, testing phase metrics (TM): a) testing team experience (TTE), b) stake-holders involvement (SI) and c) size of the software (KLOC) are taken as input. The outputs of the model are fault density indicator at the end of requirements phase (FDR), design phase (FDD), coding phase (FDC) and testing phase (FDT). It is important to mention here that these metrics may fallow either linear or logarithmic scale based on their nature.

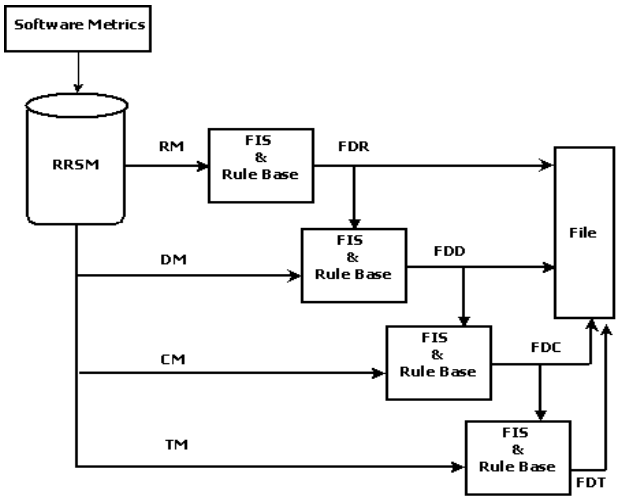

Figure 1. Software fault prediction model

Table 1. Phase-wise input/output variable

Phase	Input Variables	Output Variables
Requirement	RC, RS, RIW	FDR
Design	FDR, DTE, PM	FDD
Coding	FDD, CTE, DPF	FDC
Testing	FDT, TTE, SI, KLOC	FDT

4. IMPLIMENTATION

The model is based on the fuzzy logic and implemented in MATLAB. The basic steps of the model are identification of

RRSMs as input/output variables, development of fuzzy profile of these input/output variables defining relationship between input and output variables and fault prediction at the end of software life cycle using fuzzy inference system (FIS). These basic steps can be grouped into three broad phases as follows: a) information gathering phase, b) information processing phase, and c) fault prediction phase.

4.1 Information gathering phase

The quality of fuzzy approximation depends mainly on the quality of information collected and expert opinion. The information gathering phase is often considered the most vital step in developing a fuzzy inference system and includes the following steps:

4.1.1 Selection of input / output variables

A list of RRSMs applicable to various phase of software life cycle is given in Appendix A [19]. Ten input and four output variables are identified, as shown in Table 1, for the purpose of predicting number of faults in the software. Input variables are the RRSMs relevant to each phase of software life cycle and output variables are the fault densities at the end of each phase.

We found that metrics RC, RS, and RIW are more suitable for requirement phase and influence to the requirements faults. If RC is more, the number of faults will be more, but this is not true for RS. Similarly, if there are more reviews and inspection, more faults will be detected and corrected leaving fewer faults in the requirements phase. For design phase, two metrics DTE and PM are considered because both of these metrics are responsible for error free software design. For higher value of DTE and PM, there will be lower number of design faults in the software. At coding phase, CTE and DPF metrics is found more suitable to affect the coding faults. In general it is found that if CTE and DPF are more, the number of faults will be less. Lastly, for testing phase, three metrics TTE, SI and KLOC are taken which can influence the fault density at this phase.

4.1.2 Fuzzy profile development

Input /output variables identified at the previous stage are fuzzy in nature and characterized by fuzzy numbers. Fuzzy numbers are a subset from the real numbers set, representing the uncertain values. All fuzzy numbers are related to degrees of membership, which state how true it is to say if something belongs or not to a determined set.

There are various types of fuzzy numbers and its nomenclature is, in general, associated with its format, such as: sine numbers, bell shape, polygonal, trapezoids, triangular, and so on. Triangular fuzzy numbers (TFN) are convenient to work with because in real space, adding two TFN involves adding the corresponding vertices used to define the TFNs. Similar simple formulas can be used to subtract or find the image of TFNs. Also TFNs are well suited to modeling and design because their arithmetic operators and functions are developed, which allow fast operation on equations. Because of these properties we have considered TFNs, for all input/output variables. We have divided input variables into five linguistic categories as: i.e. very low (VL), low (L), moderate (M), high (H) and very high (VH), and for output variables we have considered seven linguistic categories i.e. very very low (VVL), very low (VL), low (L), moderate (M), high (H), very high (VH), and very very High (VVH).

The data, which may be useful for selecting appropriate linguistic variable, is generally available in one or more forms such as expert's opinion, software requirements, user's expectations, record of existing field data from previous release or similar system, etc. [22]. Fuzzy membership functions are generated utilizing the linguistic categories such as identified by a human expert to express his/her assessment and the nature of the metrics. As stated earlier, that software metrics may be either linear or logarithmic nature. On the basis of this information, fuzzy profiles (FP) of each software metrics are developed using the following formula,

(a) For logarithmic nature software metrics,

$$FP = \left[1 - \frac{\log_{10} 1{:}5}{\log_{10} 5} \right]$$

The profiles may take the following values:

VL (0; 0; 0.14), L (0; 0.14; 0.32), M (0.14; 0.32; 0.57), H (0.32; 0.57; 1.00), and VH (0.57; 1.00; 1.00)

(b) For linear nature software metrics,

$$FP = \left[\frac{(0:4)}{4} \right]$$

The profiles may take the following values:

VL (0; 0; 0.25), L (0; 0.25; 0.50), M (0.25; 0.50; 0.75), H (0.50; 0.75; 1.00), VH (0.75; 1.00; 1.00)

Outputs are considered on logarithmic scale, and divided in seven linguistic categories as:

Fuzzy profile range = [1-{log10 (1:7)} / {[log10 (7)}]; the profiles may take the following values:

VVL (0; 0; 0.08), VL (0; 0.08; 0.17), L (0.08; 0.17; 0.29), M (0.14; 0.32; 0.57), H (0.17; 0.29; 0.44), VH (0.44; 0.64; 1.00), VVH (0.64; 1.00; 1.00)

It is assumed that out of these ten input variables, only three variables (RIW, PM and DPF) follow linear nature and remaining variables follow logarithmic nature. All output variables are assumed to be following logarithmic nature. Fig. 2 to Fig. 15 shows membership functions and fuzzy profiles of all the selected input/output variables for visualization purpose.

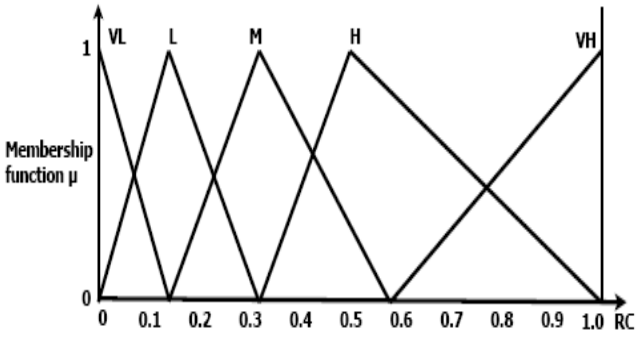

Figure 2. Fuzzy profile of RC

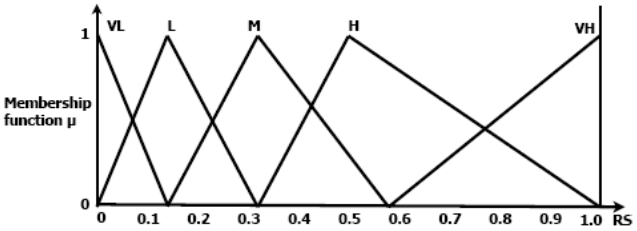

Figure 3. Fuzzy profile of RS

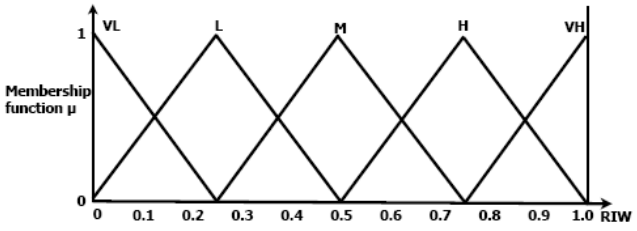

Figure 4. Fuzzy profile of RIW

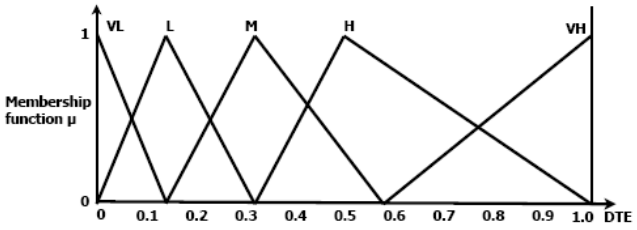

Figure 5. Fuzzy profile of DTE

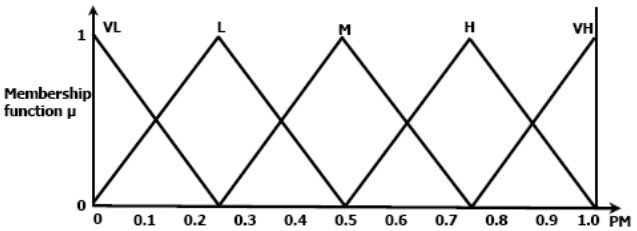

Figure 6. Fuzzy profile of PM

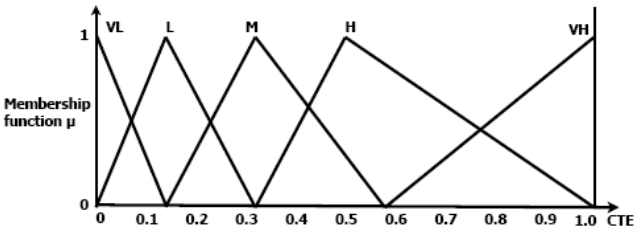

Figure 7. Fuzzy profile of CTE

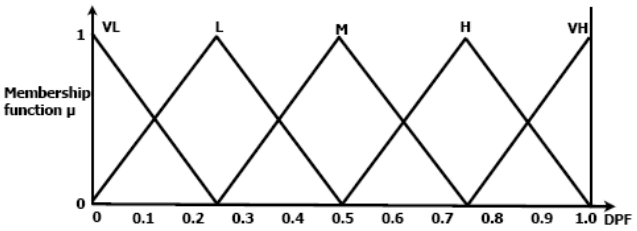

Figure 8. Fuzzy profile of DPF

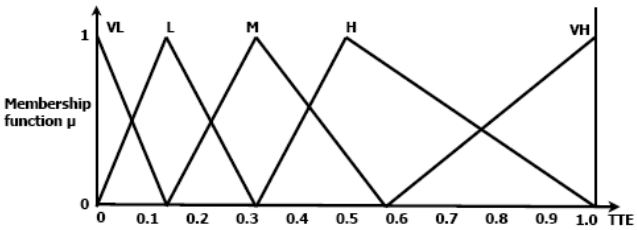

Figure 9. Fuzzy profile of TTE

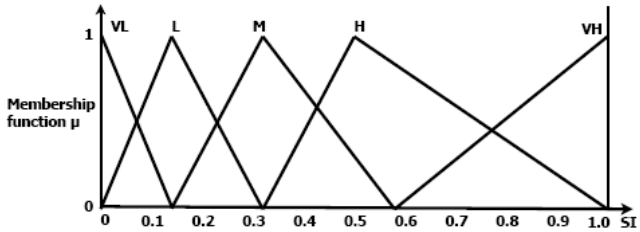

Figure 10. Fuzzy profile of SI

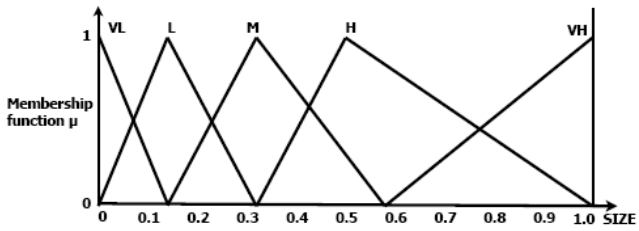

Figure 11. Fuzzy profile of SIZE

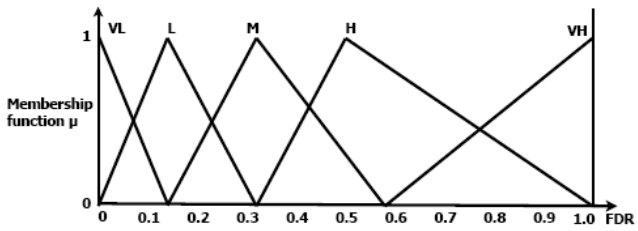

Figure 12. Fuzzy profile of FDR

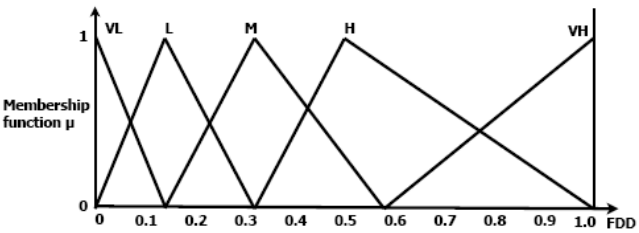

Figure 13. Fuzzy profile of FDD

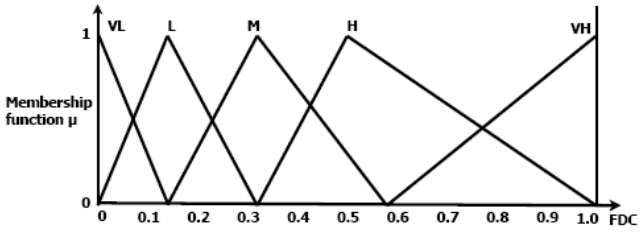

Figure 14. Fuzzy profile of FDC

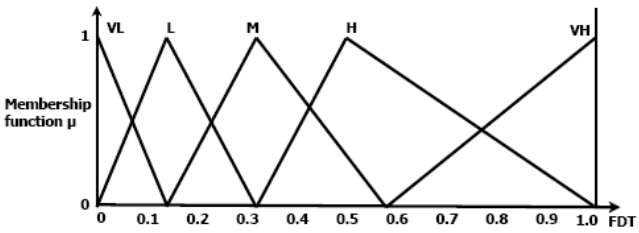

Figure 15. Fuzzy profile of FDT

4.1.3 Development of fuzzy rule base

The most important part of any inference system is the rules, and how they interact with each other to generate results. In general, rules are formed using various domain experts so that the system can emulate the inference of an actual expert. To develop fuzzy rule base, we can acquire knowledge from different sources such as domain experts, historical data analysis of similar or earlier system, and engineering knowledge from existing literature's [25, 12]. In our experiments, we generated some rules from the software engineering point of view and some from project management view points. All the rules take the form of `If A then B'. Table 2 to Table 5 show the fuzzy if-then rules required for each phase of software life cycle.

Table 2. Fuzzy rules at requirements phase

Rule	RC	RS	RIW	FDR
1	L	L	L	VL
2	L	L	M	L
3	L	L	H	M
.	.	.	.	.
.	.	.	.	.

Table 3. Fuzzy rules at design phase

Rule	FRP	DTE	PM	FDD
1	VL	L	L	VL
2	VL	L	M	VL
3	VL	L	H	L
.	.	.	.	.

Table 4. Fuzzy rules at coding phase

Rule	FDP	CTE	DPF	FDC
1	VL	L	L	VL
2	VL	L	M	VL
3	VL	L	H	L
.	.	.	.	.
.	.	.	.	.

Table 5. Fuzzy rules at testing phase

Rule	FCP	TTE	SI	K	FDT
1	VL	VL	L	L	VL
2	VL	VL	L	M	VL
3	VL	VL	L	H	L
.	.	.	.	.	.
.	.	.	.	.	.

4.2 Information Processing Phase

In this phase, the fuzzy system maps all inputs on to an output. This process of mapping is known as fuzzy inference process or fuzzy reasoning [26, 24]. Basis for this mapping is the number of fuzzy IF-THEN rules, each of which describes the local behavior of the mapping. The Mamdani fuzzy inference system [27] is considered here for all the information processing. We have taken the centroid value of each fuzzy profile for computation purpose. Centroid value of each fuzzy profile can be computed from its fuzzy number and these are shown in the Table 6. Another vital part of information processing is the defuzzification process, which derives a crisp value from a number of fuzzy values. Various defuzzification techniques are Centroid, Bisector, Middle of maximum, Largest of maximum, Smallest of maximum, etc. [23]. The most commonly used method is the centroid method, which returns the centre of area under the curve, is considered here for defuzzification.

4.3 Fault Prediction Phase

On the basis of fault density at the end of testing phase, total number of faults in the software is predicted. The faults are predicted from FTP value, which is an indicator of fault density at the end of testing phase. The number of faults detected is proportional to the both size of the software and the fault density indicator value and can be found as:

$$FaultP = C1 * LOC * FTP \quad (1)$$

However, fault detection process is not exactly linear with size. As size of a software increases, portion of faults detected decreases due to saturation, time and efforts requirements, increased possibility due to interactions among the variables and instructions. Therefore, the FTP value is divided by (1+exp (-LOCi /C2)), where the value of C2 scales the effect of LOC value. Thus (1) becomes,

$$FaultP_i = C1 * LOC_i * FTP_i / (1 + \exp(-LOC_i / C2)) \quad (2)$$

Where FaultPi, is the total number of predicted faults in the ith software project, LOC is the size of ith project, FTPi, is the fault density indicator at the end of testing phase of project i, and C1 and C2 are two constants obtained through learning. Value of C1 and C2 are obtained from available project data. The values of C1 and C2 for current project are obtained as 0.04 and 107 respectively. The proposed fuzzy inference model is generic in nature to capture variation in faults present in the software.

5. RESULTS

Fifteen different software projects data are taken form Appendix A, to analyze the prediction accuracy of the proposed model. For this, mean absolute percent error (MAPE) is calculated to find the absolute percentage error for each predicted faults and computing the average of these values. The predicted MAPE value is found to be 9.86, which shows good prediction accuracy.

Table 6. Centroid value of fuzzy profiles

	RC	RS	RIW	DTE	PM	CTE	DPF	TTE	SI	Size
VL	0.05	0.05	0.08	0.05	0.08	0.05	0.08	0.05	0.05	0.05
L	0.15	0.15	0.25	0.15	0.25	0.15	0.25	0.15	0.15	0.15
M	0.34	0.34	0.50	0.34	0.50	0.34	0.50	0.34	0.34	0.34
H	0.63	0.63	0.75	0.63	0.75	0.63	0.75	0.63	0.63	0.63
VH	0.86	0.86	0.92	0.86	0.92	0.86	0.92	0.86	0.86	0.86

Table 7. Number of faults at the end of each phase

#	FRP	FDP	FCP	FTP	Size	Efforts	No. of faults predicted by proposed model	Faults observed by [19]	Percentage errors
1	0.0925	0.0834	0.0739	0.3008	1000	1976	6.02	5	-20.33
2	0.1827	0.1378	0.1332	0.2703	4840	4410	26.17	29	9.77
3	0.1827	0.3002	0.1802	0.4645	5790	8822	53.81	53	-1.52
4	0.2087	0.1898	0.142	0.5	11000	3764	110.06	91	-20.95
5	0.1872	0.1504	0.1361	0.2395	26670	7121	127.94	109	-17.37
6	0.2892	0.2498	0.1523	0.1406	49100	25450	138.41	129	-7.29
7	0.7829	0.7947	0.8577	0.8692	50000	52660	871.41	928	6.1
8	0.4778	0.5383	0.1817	0.3838	52000	14602	400.17	412	2.87
9	0.2908	0.2554	0.1552	0.195	53860	18171	210.61	209	-0.77
10	0.458	0.3002	0.2322	0.5319	58300	33472	622.01	672	7.44
11	0.4778	0.4944	0.3929	0.564	61000	53995	690.17	680	-1.5
12	0.6306	0.305	0.2499	0.3281	87000	12388	573.44	476	-20.47
13	0.5653	0.5	0.3932	0.6528	99000	24895	1298.84	1597	18.67
14	0.4941	0.3006	0.2338	0.5302	154000	34893	1645.64	1768	6.92
15	0.607	0.6541	0.457	0.6458	155200	32366	2020.16	1906	-5.99

6. CONCLUSION AND FUTURE SCOPE

This paper has developed a fault prediction model using early phase software metrics and fuzzy inference system. Fuzzy profiles of metrics were developed by a human expert to express his/her assessment and the nature of the metrics. For this, ten software metrics are taken from PROMISE data set and fault density at the end of each phase is predicted using fuzzy inference system. For software professionals, this model provides an insight towards software metrics and its impact on software fault during the development process. For software project mangers, the model provides a methodology for allocating the resources for developing reliable and cost-effective software. There are various models for fault prediction and each one has its own strengths, and weakness. An interesting open problem for the future work is to find the failure rate form these fault densities, and predict the software reliability, safety, availability from these values.

Appendix A: PROMISE database: Twenty six software project data

#	Project	RCF1	RSS7	RIWS3	DTED1	PMD4	CTED2	DPFD3	TTET2	SIP5	EffortsE	SizeK	FaultsTD
1	1	M	L	VH	L	H	H	H	H	H	7108.82	6.02	148
2	2	L	H	VH	L	H	H	H	H	H	1308.08	0.90	31
3	3	H	H	VH	H	VH	VH	H	H	VH	18170.00	53.86	209
4	5	H	M	H	L	H	M	H	M	M	9434.00	14.00	373
5	7	L	M	VH	M	H	VH	H	M	VH	13888.27	21.00	204
6	8	M	H	H	H	M	H	M	M	H	8822.00	5.79	53
7	10	M	H	H	H	H	H	H	M	H	4410.00	4.84	29
8	11	H	H	H	H	H	H	H	H	H	14196.00	4.37	71
9	12	H	L	H	VH	H	M	M	H	H	13387.50	19.00	90
10	13	H	L	M	H	H	H	H	M	H	25449.60	49.10	129
11	14	VH	H	H	H	H	H	H	H	H	33472.00	58.30	672
12	15	H	VL	H	H	H	H	H	H	VH	34892.65	154.00	1768
13	16	L	M	H	H	H	H	H	H	VH	7121.00	26.67	109
14	17	L	M	M	M	H	M	H	L	M	13680.00	33.00	688
15	18	VH	VL	H	M	H	H	H	H	VH	32365.98	155.20	1906
16	19	H	M	H	H	H	H	H	M	H	12387.65	87.00	476
17	20	VH	VL	M	VL	H	VL	L	VL	H	52660.00	50.00	928
18	21	L	M	H	H	H	H	H	H	H	18748.00	22.00	196
19	22	M	L	M	H	H	M	L	M	H	28206.00	44.00	184
20	23	H	M	VH	L	H	H	H	H	H	53995.00	61.00	680
21	24	M	L	M	M	M	H	H	M	M	24895.00	99.00	1597
22	27	H	M	VH	M	H	L	M	M	M	14602.00	52.00	412
23	28	VH	L	VH	M	H	L	H	M	M	8581.00	36.00	881
24	29	M	VH	VH	VH	H	VH	H	VH	VH	3764.00	11.00	91
25	30	L	VH	VH	H	H	H	H	H	VH	1976.00	1.00	5
26	31	M	M	H	H	H	H	H	H	VH	15691.00	33.00	653

REFERENCES

[1] ANSI/IEEE Standard Glossary of Software Engineering Terminology, IEEE STD-729, 1991.

[2] M. Agrawal, K. Chari, Software Effort, Quality and Cycle Time: A Study of CMM Level 5 Projects, *IEEE Transaction on Software Engineering*, vol. 33, no. 2, pp. 145-156, 2007.

[3] J. D. Musa, A. Iannino, K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill Publishers, New York, 1987.

[4] C. Kaner, Software Engineering Metrics: What Do They Measure and How Do We Know, *10th International Software Metrics Symposium 2004*.

[5] H. Pham, *System Software Reliability*, Reliability Engineering Series, Springer-Verlag Publisher, London, 2006.

[6] J. E. Gaffney, C. F. Davis, An Approach to Estimating Software Errors and Availability, *Proceedings of 11th Minnow brook Workshop on Software Reliability 1988*.

[7] J. E. Gaffney, J. Pietrolewiez, An Automated Model for Software Early Error Prediction (SWEEP), *Proceedings of 13th Minnow brook Workshop on Software Reliability1990*.

[8] Technical Report, *Report Methodology for Software Reliability Prediction and Assessment*, Rome Laboratory (RL) RL-TR-92-52, vol. 1 & 2, 1992

[9] W. W. Agresti, W. M. Evanco, Projecting Software Defects form Analyzing Ada Design, *IEEE Transaction on Software Engineering*, vol. 18, no. 11, pp. 988-997, 1992.

[10] T. J. Yu, V. Y. Shen, H. E. Dunsmore, (1988), An Analysis of Several Software Defect Models, *IEEE Transaction on Software Engineering*, vol. 14, no. 9, pp. 261-270, 1988.

[11] T. M. Khoshgoftaar, J. C. Munson, Predicting Software Development Errors Using Complexity Metrics, *IEEE Journal on Selected Areas in Communication*, vol. 8, no. 2, pp. 253-261, 1990.

[12] X. Zhang, H. Pham, An Analysis of Factors Affecting Software Reliability, The Journal of Systems and Software, vol. 50, no. 1, pp. 43-56, 2000.

[13] M. Li, C. Smidts, A Ranking of Software Engineering Measures Based on Expert Opinion, *IEEE Transaction on Software Engineering*, vol. 29, no. 9, pp. 811-824, 2003.

[14] K. S. Kumar, R. B. Misra, An Enhanced Model for Early Software Reliability Prediction using Software Engineering Metrics, *Proceedings of 2nd Int. Conf. on Secure System Integration and Reliability Improvement*, pp. 177-178, 2008.

[15] A. K. Pandey, N. K. Goyal, A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics, *International Journal of Electronics Engineering*, vol. 1, no. 2, pp. 239-245, 2009.

[16] M. S. Krishnan, M. I. Kellner, Measuring Process Consistency: Implications Reducing Software Defects, *IEEE Transaction on Software Engineering*, vol. 25, no. 6, pp. 800-815, 1999.

[17] M. Diaz, J. Sligo, How Software Process Improvement Helped Motorola, *IEEE Software*, vol. 14, no. 5, pp. 75-81, 1997.

[18] D. E. Harter, M. S. Krishnan, S. A. Slaughter, Effects of Process Maturity on Quality, Cycle Time and Effort in Software Product Development, *Management Science*, vol. 46, pp. 451-466, 2000.

[19] http://promisedata.org/

[20] IEEE Standard, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software, IEEE Standard 982.2, New York, 1988.

[21] D. C. Montgomery, *Design and Analysis of Experiments*, Wiley-India, New Delhi, 2005.

[22] K. S. Saravana, R. B. Misra, N. K. Goyal, Development of Fuzzy Software Operational Profile, *International Journal of Reliability, Quality and Safety Engineering*, vol. 15, no. 6, 581-597, 2008.

[23] T. Ross, *Fuzzy Logic with Engineering Applications*, Wiley-India, New Delhi 2005.

[24] L. A. Zadeh, Knowledge representation in fuzzy logic, *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, pp. 89-100, 1989.

[25] M. Xie, G. Y. Hong, C. Wohlin, Software reliability prediction incorporating information from a similar project, *The Journal of Systems and Software*, vol. 49, pp. 43-48, 1999.

[26] J. B. Bowles, C. E. Pelaez, Application of fuzzy logic to reliability engineering, *IEEE Proceedings*, vol. 83, no. 3, pp. 435-449, 1995.

[27] E. H. Mamdani, Applications of fuzzy logic to approximate reasoning using linguistic synthesis, *IEEE Transactions on Computers*, vol. 26, no. 12, pp.1182-1191, 1977.