IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, May 2022

A Predictive Approach to Estimate Software Defects Using Fuzzy Logic

[1*] T.Ravi Kumar, [2] Kalyana Kiran Kumar, , [3] Suneelgoutham Karudumpa,[4] Ch.Rajasekhara Rao, [5*] Balamurali Pydi

[1] Dept of CSE, Aditya Institute of Technology and Management, Tekkali, AP, India.
[2] Dept of EEE, Aditya Institute of Technology and Management, Tekkali, AP, India.
[3] Dept of EEE, Aditya Institute of Technology and Management, Tekkali, AP, India.
[4] Dept of ECE, Aditya Institute of Technology and Management, Tekkali, AP, India.
[5] Dept of EEE, Aditya Institute of Technology and Management, Tekkali, AP, India.

[*]Corresponding Author: Balamurali Pydi balu_p4@yahoo.com

Abstract:
Context: Software plays vital role in every day's life. Historical data shows as evidence that even a minor error in the software can make huge loss in terms of lives and economy. Software quality and reliability has been uncompromising attribute in the software life cycle. The software defect can be defined as 'small deviation in the process, improper requirement, or misrepresentation of the definition. The defect causes a lateral cost of budget as well as schedule over run. To avoid it, it's required to forecast software defects in the different phases of the software life cycle. The common approach to denote software quality is 'to reveal the number of defects present in it', and it is measured in terms of software defect density. The defect density is measured as a mathematical division of 'the total number of defects to its size'.

Objective: This paper emphasizes on the prediction of software defect density using fuzzy logic based approach.
Method: In the proposed model, defect density indicator is measured in the four phases of the SDLC (i.e. requirement, design, development, and testing). The defect density indicator of each phase is fed as one of the input metrics to the immediate next phase. Other metrics like, Requirement stability, Cyclomatic complexity, staff experience etc are fed to the corresponding phase.
Results: It's been found that the accuracy of the predicted defects is very close the actual defects. Validation results show this method is more accurate than the other fuzzy implementations.
Conclusion : The fuzzy logic approach to predict the software defect density is been verified on different real time data sets ranging from very low project sizes to the very high project size.

Keywords:

Software defects, Fuzzy logic, DDI, Cyclomatic complexity, Software metrics, SDLC, MMRE, BMRE etc.

Introduction:

These days, all humans are using software either directly or indirectly in their day to day life. In the current era of information technology, software quality is crucial to keep customer satisfaction as well as business growth. Software quality management[1][2] to ensure reliable software is the million dollar question for the software developers and companies. Even the industry standards like ISO, CMM and Six Sigma are emphasizing the needs of reliable software. Development of reliable software product as per the market trends is not an easy task at a given scope, time and budget[3]. It's prone to errors either knowingly or unknowingly at different phases of the software development life cycle (SDLC). So, software defect prediction at each phase is a value-add to the businesses. Software defect density is closely tied to the size and complexity of the software[4]. In each phase of SDLC there are so many metrics which contribute to the software defects[5]. The number of defects is directly proportional to the size of the software. Software defect data is not available in the beginning of the SDLC[6] It drives the need of fuzzy logic based approach to determine the software defect density. In this paper we use fuzzy logic to predict the defect density indicator which in turn provides the no of defects for the given size of the project. In this paper Section 2 describes about the related work followed by section 3 explains about the software metrics requirement, Section 4 describes the proposed model and validation results covered in sec 5.

2. Related work:

Software reliability has become the main criteria for the most of the organizations. Varied researches went into refining the software development process and develop reliable and efficient software, and continuing the refining the process day by day. Huge number of models has been proposed for estimation and prediction software reliability[7][8].

Review based models:

In the olden days, software reliability is closely associated with the continuous reviews in all phases of SDLC (i.e. requirement, design, coding and testing). These models conclusions related to software reliability are drawn based on reviews and re reviews [9].

Failure data based models:

Gaffney and Davis proposed [10,11], failure based model for predicting software reliability. It emphasizes on the fault statistics found during the review of various software development phases. Later on, the evolution of UML based models and Bayesian frame work models based on reviews and analysis are become inefficient and research lead towards the models which can be validated through coding.

Programmable models:

After evolution of Object oriented models to identify software reliability, it's been adopted bottoms up approach to find software defects. Defect prediction at early stages of process maturity and software metrics is proposed by Pandey and Goyal[12]. This model could not justify the usage of fuzzy profiles for different metrics. Yadav [13 et all enhanced and proposed a software defect prediction model using fuzzy basedapproach. It predicts the defect density in each phase and passes the DDI to the next phase to evaluate the total defect density. This paper leverages the fuzzy based approach.

Related work in Fuzzy Logic:

The software effort is estimated in terms of man-hours and it is of fuzzy nature. Therefore, defect-free software development is a challenging task. It is very crucial to ensure that the underlying software will perform its intended functions correctly. Therefore, there is a growing need to ensure reliability of software systems as early as possible.

Based on different literature survey and review, it is found that the software reliability is a function of number of residual defect in the software. Reliability relevant software metrics play a vital role in defect prediction and these metrics are of fuzzy nature. Therefore, in this research paper, a fuzzy logic based model for phase-wise software defects density prediction is developed using the reliability relevant software metrics.

Related work in defining software metrics:

There are different software metrics used in measuring software reliability. The important metric influencing the software defect density is software size. Majority of the model uses software size and complexity is the metrics. Zhang and Pham[23] suggested thirty-two factors which have impact on the software reliability in all stages of the software development process. Phase wise ranking for the software metrics which impacts the software reliability is provided by Li [14,15] et al.

Based on the different literature survey Fuzzy logic based approach can be leveraged to identify the software defect density and estimate the original defects. Next section will introduce the different software metrics used in the proposed model.

3. Software metrics:

Software metrics are the backbone of any model to decide the software reliability. There are so numerous metrics with effects the software quality in each phase of SDLC. In the current model we take different software metrics as inputs and are listed below. Initially we define the weights of this metrics based on the expertise, as the learning of this model is evolved weights will be adjusted to make the predicted defects equals to the original defects.
Following is the through discussion of the input software metrics.

i. Software size:

As the number of defects is directly proportional to the software size, the software size is most important metric in deciding the software reliability. This metric is measured in lines of code (LOC), and usually measured in KLOC. As the complexity of software increases there is a high chance that software size gets increases and proportionally the defects.

A well defined development process and clean coding techniques can reduce the possibility of defect occurrence, even the complexity and software size increases.

ii. Requirement phase software metrics:

Requirement phase is the first and important phase in the Software Lifecycle. A well defined and frozen requirement definition can highly impact the software quality [16]. At the requirement phase, input layer is populated with Requirement stability, Fault density and Review Information.

1. Requirement stability (RS): Frozen and Stable requirement is directly proportional to the software efficiency and reliability. Stable and frozen requirement gives a freehand to the designers and testers to concentrate on the reliability. If the change requests are more i.e. unfrozen requirements disrupt the process in SDLC which in turn cause the explosion of the software. Well defined requirements can avoid adverse effect on the cost, quality, reliability and schedule of thesoftware.
2. Fault density (FD): Fault density is inversely proportional to the software reliability. In requirement analysis phase fault density can range from a low priority issue to the high priority issue which can impact the software in different ways along with increasing the probability of the defects. Continuous requirement analysis and early requirement description may reduce the faultdensity.
3. Review Information (RI): Requirements review is directly proportional to the software reliability. Well reviewed requirement will eliminate any defects in the later phases. As found the issue in the

later stages of the SDLC will cost more in terms of cost and effort, it would be better to have a complete review of the requirements internally and externally with all the stakeholders. Reviews should happen based on the software requirement specifications outlined by the prima organisations.

ii. Design phase software metrics:

Design phase follows requirement phase. As requirement phase defects affect the design phase. The defect density of requirement phase is prime input to the design phase. Apart from it, software complexity and design review are considered as the software metrics.

i. Software complexity (SC): software defects are directly proportional to the Software complexity. Software design with more number of components can make software more complex. It is good to have a metric in the design phase for well executed complex [18] software. It can provide good indication for remaining software defects.

ii. Design Review (DR): Design review is done to identify the defects or faults occurred in the design phase. Design review should make sure the design document outlines each and every detail of requirement specifications and match with the intent of both. Whenever there is a requirement change it should be promptly conveyed to design phase and get it reviewed. After each review based on the modifications if necessary a re-review is suggested to make software quality assurance. Design review has a capability of eliminating defects from the laterphases.

iii. Coding phase metrics:

Coding phase follow the design phase. A matured design document will be a very good reference for the programmers. There is a practice in industry that most of the design reviews happens with the programmers.

i. Programmer capabilities (PC): A technically skilled and experienced programmer can write reliable software irrespective of the software size and complexity. Keeping this mind in program management will consider programmers from the background of good education from reputed institutes; intelligence and domain knowledge are the programmer capabilities, which are directly proportional to the software ere liability.

ii. Process maturity (PM): A well established process is very important in the software life cycle. A well defined process will reduce the gaps between the teams working globally to achieve reliable software. Process metrics can be utilized to gauge, screen and enhance the reliability and quality of software [19,20] . The process helps in the smooth development flow to achieve targets.

iv. Testing phase software metrics:

Verification or testing is the final and crucial phase in software life cycle. This phase is useful to find the defects/bugs before software goes to the end users. In most of the scenarios this phase contributes more towards the software reliability and quality. Staff experience and quality of tests are the prime metrics of this phase.

iii. Staff experience (SE): As verification phase requires a lot of resources for the good execution, a technically sound and experienced staff contribution to the testing phase can directly impact the software quality. Globally organizations follow a pyramid pattern to balance the experience and knowledge domains

4. Quality of testing (QT): A comprehensive test suite required for a well developed software product. As software testing is costly and time consuming effective tests cases will assure a quality software time to market. Well written tests will expose the software defects. Tests should cover all the scenarios of the requirement. Good documentation of the issues or tests will ensure the software ere liability.

5. Proposed Model:

The architecture design of the proposed modal is shown in figure 4.1. In the proposed design model, the defect density indicator measures in the initial for phases of SDLC. So the proposed model uses reliable and significantly best measurements of requirement analysis, design, coding and testing of SDLC. The metrics are shown using oval structures in the currently proposed mode. The stable requirement defect density review, inspection and walkthrough ahs been fed as input, in the requirement analysis phase, to find out the defects density index at the end of the analysis phase. In this requirement phase defect density indicator (RPDDI), Cyclomatic complexity, and design review and walkthrough metrics have been considered as input in design phase to predict the defect density index (DPDDI). The design phase defect density index or indicator (DPDDI), team capability and maturity of process have been considered as input to coding phase to forecast the defect density indicator towards the completion of coding phase. So, the coding phase defects density indicator (CPDDI), Team experience and skill, and the reliability of test cases have been considered as contribution to testing phase to forecast the defect density index towards the end of testing phase. The proposed model has been segregated into the following steps.

Selection of software metrics.

A. Define membership function of input and output metrics.
B. Design fuzzy rules.
C. Perform fuzzy interface and defuzzification.

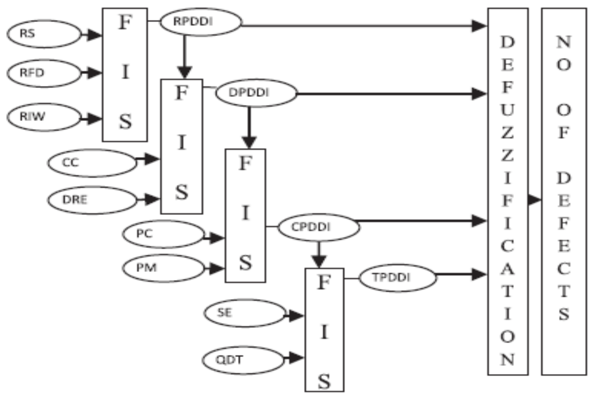


Figure 4.1: Architecture Design Model

RS: Requirement Strength

RFD: Requirement failing denseness

RIW: Review, importance and walk through

CC: Cyclomatic Complication

DRE: Design Revise Efficiency

PC: Programmer Capability

PM: Process Maturity

SE: Staff experience

QDT: Quality of documented test cases

RPDD: Requirement assessment point desert denseness indicator

DPDDI: Design point desert denseness indicator

CPDDI: Coding point desert denseness indicator

TPDDI: Test point desert denseness

FIS: Fuzzy interpretation scheme

In Fuzzy based approach each phase in the SDLC is taken as FIS instance and phase inputs are provided to the FIS. Along with the phase inputs previous phase defect density also provided to account the previous phase defects. With this approach if previous phase software metric directly influences on the current phase, then the defect density is compromised in the current phase.

Normalization Process:

A normalization process has been followed to have definite input and output range w.r.to the software size. The normalize inputs are defined with different levels based on the metric severity and impact on the software reliability. Following illustrations describe the normalization input software metrics.

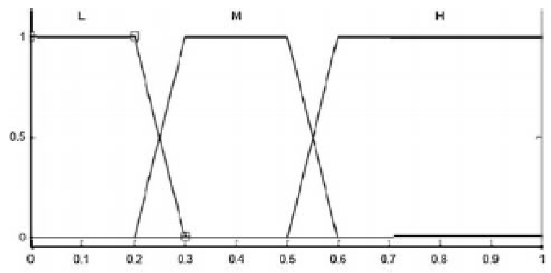


Figure2:RS

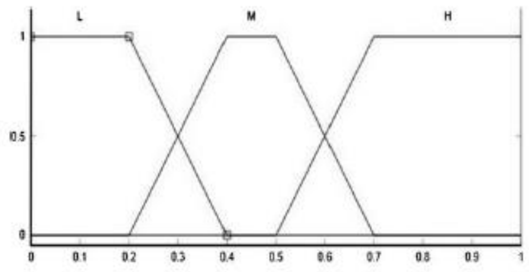


Figure 3:FD

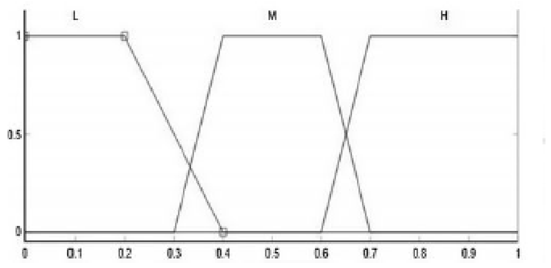


Figure4:RI

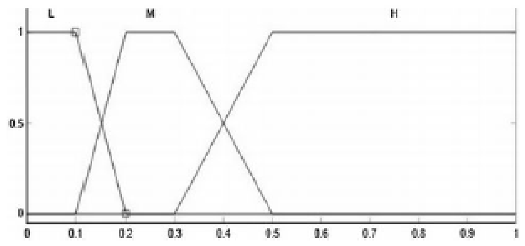


Figure5:CC b

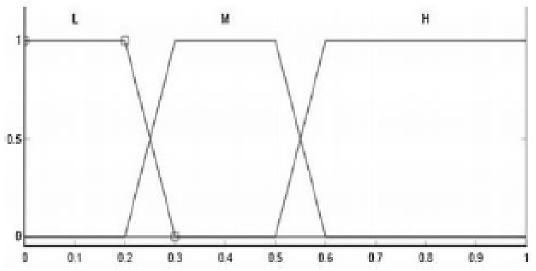


Figure 5:CC

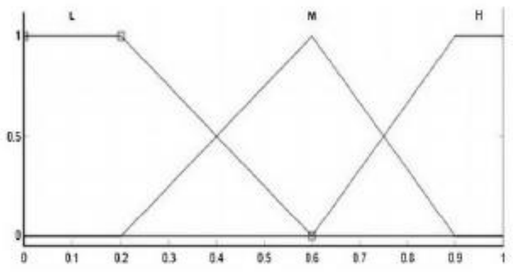
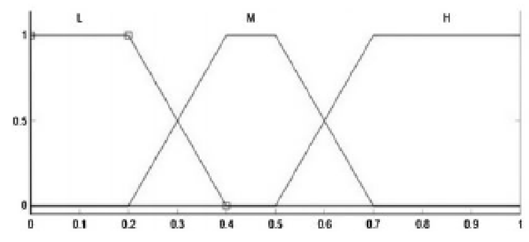


Figure 6: DR



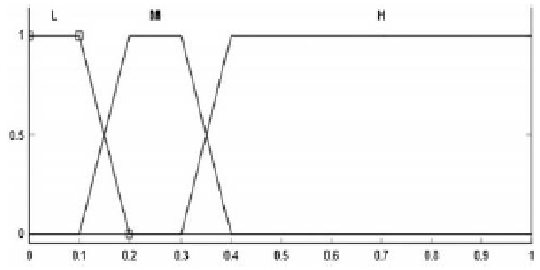


Figure 8:PM

Figure 9: SE

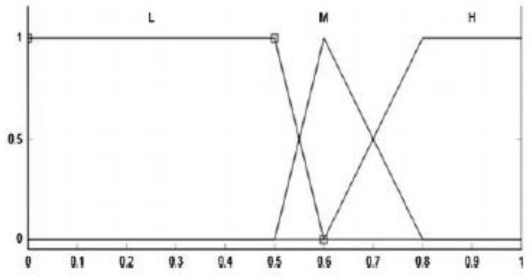


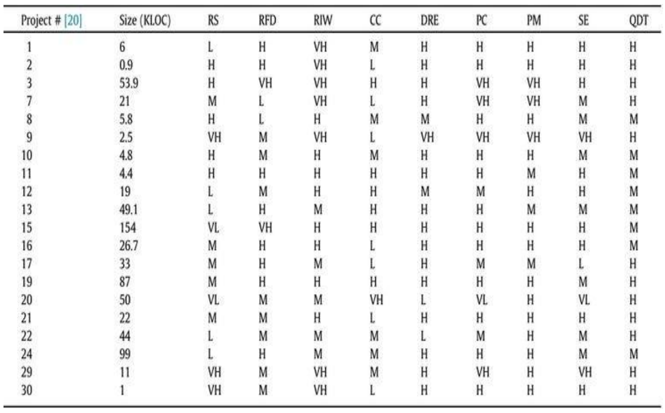
Figure 10 : QD

Training Module (TM):

Training module is very important part of the current fuzzy logic approach. This module helps in defining input data sets as well as identifying severity levels from Very Low (VL) to Very High (VH). The system is trained with around 20 real time data sets taken from the promise database [21]. Training module assigns severity levels to achieve local and global minima of the error delta. Where error delta is been defined as the difference between the original defects and the predicted defects. Local minima are corresponding to hitting the minimum error delta w.r.to SDLC phases and Global minima is the consolidated error delta of the system [22,23]

V. Results:

Table 1: Training inputs



Project # [20]	Size (KLOC)	RS	RFD	RIW	CC	DRE	PC	PM	SE	QDT
1	6	L	H	VH	M	H	H	H	H	H
2	0.9	H	H	VH	L	H	H	H	H	H
3	53.9	H	VH	VH	H	H	VH	VH	H	H
7	21	M	L	VH	L	H	VH	VH	M	H
8	5.8	H	L	H	M	M	H	H	M	M
9	2.5	VH	M	VH	L	VH	VH	VH	VH	H
10	4.8	H	M	H	M	H	H	H	M	M
11	4.4	H	H	H	H	H	H	M	H	M
12	19	L	M	H	H	M	M	H	H	M
13	49.1	L	H	M	H	H	H	M	M	M
15	154	VL	VH	H	H	H	H	H	H	M
16	26.7	M	H	H	L	H	H	H	H	M
17	33	M	H	M	L	H	M	M	L	H
19	87	M	H	H	H	H	H	H	M	H
20	50	VL	M	M	VH	L	VL	H	VL	H
21	22	M	M	H	L	H	H	H	H	H
22	44	L	M	M	M	L	M	H	M	H
24	99	L	H	M	M	H	H	H	M	M
29	11	VH	M	VH	M	H	VH	H	VH	H
30	1	VH	M	VH	L	H	H	H	H	H

Case study	RPDDI	DPDDI	CPDDI	TPDDI	Actual Defects	Defects prediction using fuzzy
1	0.0047	0.0391	0.0062	0.0783	89	93
2	0.0142	0.0168	0.0228	0.0265	100	106
3	0.0064	0.0357	0.0091	0.00737	51	49
4	0.0171	0.0228	0.028	0.0356	225	231
5	0.0036	0.009	0.0083	0.0066	230	240
6	0.0025	0.0078	0.0065	0.0055	400	393
7	0.0044	0.0085	0.0072	0.0047	1076	1052
8	0.0468	0.0333	0.0283	0.0126	536	528
9	0.00389	0.012	0.0145	0.0115	478	476
10	0.0084	0.0984	0.0133	0.0134	1893	1887
11	0.0598	0.0022	0.014	0.0558	746	739
12	0.00375	0.0534	0.0388	0.01289	121	115
13	0.021	0.0129	0.0175	0.0174	392	402
14	0.00531	0.1004	0.1039	0.0688	73	70
15	0.0903	0.0076	0.0077	0.01275	707	684
16	0.0474	0.0301	0.0589	0.03	654	638
17	0.0059	0.0502	0.0184	0.01431	18	15
18	0.0126	0.0122	0.1059	0.0993	1357	1343
19	0.189	0.0157	0.0131	0.0564	194	187
20	0.00971	0.0067	0.0356	0.0149	893	878

Table 2: Actual defect vs predicted defects using fuzzy logic.

As shown in figure normalized severity based system is predicting the no of defects very close to the actual defects compared to the most of the methods [24]. Above results are obtained using MatLab tool kit. Real time data sets have been taken from the promise database repository.

The table shows that, most of the defects are predicted very accurately in the range of ± 2% of outputs after the normalization is rounded to the nearest integer as there cannot be decimal. The data clearly depicts the relation between the software size and the number of associated defects. Following results justify the importance of the one stage defects driving the other stage defects.

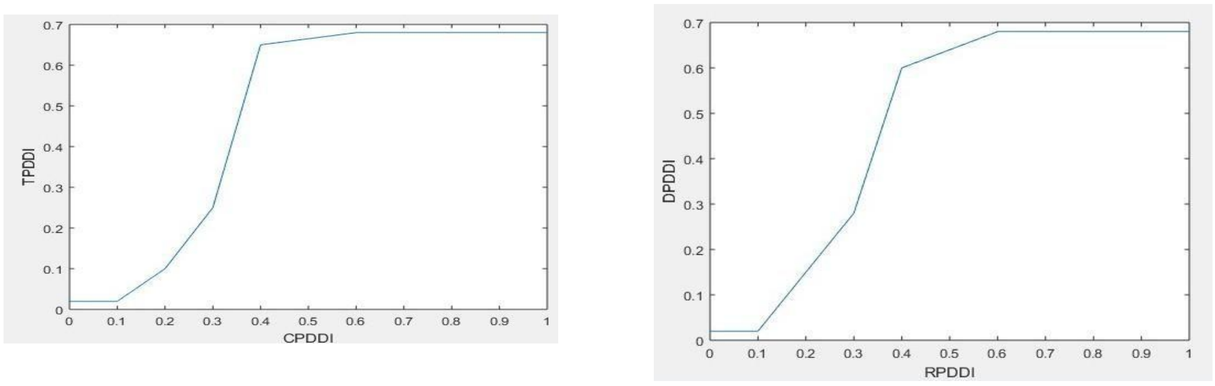


Figure 11: RFD vs Requirement phase defects

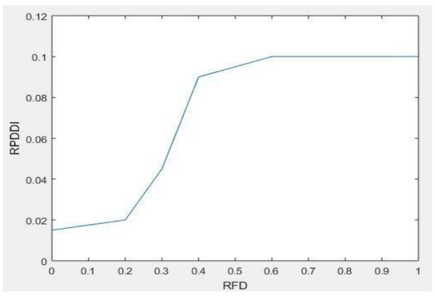


Figure 12: RPD vs Design phase Defects

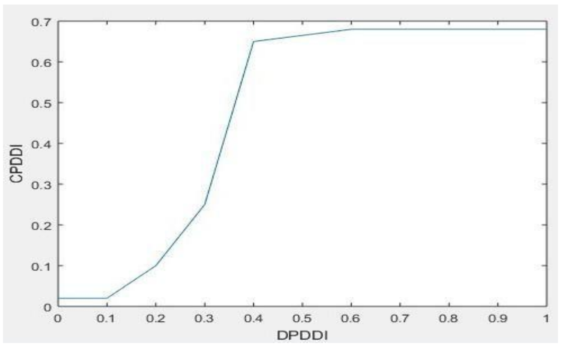


Figure 13: DPD vs Coding phase defects

Implementation of Fuzzy rules:

As per the current proposed model Fuzzy inference evaluates and combines the result of each fuzzy rule. A fuzzy max–min operator is used for conversion of one fuzzy set into another fuzzy set for crisp value as an output.
The following fuzzy rules have been implemented at each phase of the software development life cycle to get the precise results.

Requirement analysis phase fuzzy rules: There are 28 numbers of rules have been implemented at requirement phase. They are

Rule no.	Fuzzy rule
1	If RS is L and RFD is M and RIW is H then RPDDI is VL
2	If RS is L and RFD is L and RIW is L then RPDDI is VL
…	…………………………………………………………
27	If RS is H and RFD is M and RIW is L then RPDDI is L
28	If RS is H and RFD is M and RIW is M then RPDDI is M

Design phase fuzzy rules: There are 60 numbers of rules have been implemented at design phase. They are

Rule no.	Fuzzy rule
1	If CC is L and DRE is L and RPDDI is H then DPDDI is VL
2	If CC is L and DRE is M and RPDDI is H then DPDDI is VL
…	…………………………………………………………
59	If CC is M and DRE is H and RPDDI is H then DPDDI is VH
60	If CC is M and DRE is H and RPDDI is VH then DPDDI is VH

Coding phase fuzzy rules: There are 45 numbers of rules have been implemented at coding phase. They are

Rule no.	Fuzzy rule

1 If PC is L and PM is M and DPDDI is VL then CPDDI is VL
2 If PC is L and PM is L and DPDDI is H then CPDDI is H
… ………………………………………………………
44 If PC is H and PM is M and DPDDI is L then CPDDI is L
45 If PC is H and PM is H and DPDDI is M then CPDDI is M

Testing phase fuzzy rules: There are 50 numbers of rules have been implemented at testing phase. They are

Rule no.	Fuzzy rule
1	If SE is L and QDT is L and CPDDI is M then TPDDI is M
2	If SE is L and QDT is M and CPDDI is M then TPDDI is M
…	………………………………………………………
49	If SE is L and QDT is M and CPDDI is H then TPDDI is H
50	If SE is L and QDT is M and CPDDI is VH then TPDDI is VH

Model Validation:

To validate the accuracy of the predicted results of the proposed fuzzy model, commonly suggested measures[23] have been taken which are as follows.

i. Mean Magnitude of Relative Error (MMRE):

MMRE is the mean of total errors calculation and a measure of the spread of the variable Z, where Z = evaluate / real

MMRE= $\frac{1}{m}\sum_{j=1}^{m}\frac{|x_i - x_i|}{x_i}$

Where x_i is the actual value and ^x is the estimated value of a variable of interest.

ii. Balanced Mean Magnitude of Relative Error (BMMRE):

MMRE is unbalanced and assesses over rates in excess of underrates. Thus, a persistent or reliable mean magnitude of relative error measure is also considered which is as per the following:

BMMRE= $\frac{1}{m}\sum_{j=1}^{m}\frac{|x_i - \acute{x}_i|}{x_i}$

The minor value of MMRE and BMMRE specifies improved precision of prediction
For any proposed model BMMRE and MMRE values shows the model prediction accuracy. The results show that the lower the values the higher the accuracy of the model. The proposed Fuzzy based software defect density registers the BMMRE and MMRE values as follows.

Error rate	**Fuzzy Method**
MMRE	0.6761
BMRE	0.5178

Table 3: BMRE MMRE values

Conclusion:

In the current paper Fuzzy logic based approach is proposed to identify the software defects at each phase of the Software development life cycle. Input software metrics to this model is chosen based on the literature survey. Weights or severity levels assigned to each input value, which helps in tuning the system to get accurate no of software defects. Tuning of this model is based on the training module. Training module is fed with real time data from the promise database repository. Input and output normalization is done based on the software size (KLOC). It's been observed that defects predicted using this model is closer to the original defects and more accurate than other fuzzy models as gone through the literature survey. The validation results of BMMRE and MMRE shows very less value, which indicate the prediction results accuracy. The results emphasize that the proposed fuzzy logic approach with more and refined input metric rules for the prediction of software defect density and ensuring software reliability

Future work:

This model can be further extended to the artificial neural networks, Weighted artificial neural networks, and probabilistic neural networks and define the weights in more accurate way to get the defect rate close to the zero. In the similar way we can extend the model using feedback neural network by feeding error rate to adjust the weights and make error rate ~zero..

References:

1. Pentium FDIV bug "Statistical Analysis of Floating Point Flaw: Intel White Paper"(PDF). Intel. 9 July2004.p. 9. Solution ID CS-013007. Retrieved 5 April 2016.

2. The Helminthiasis of the Internethttps://tools.ietf.org/html/rfc1135

3.(1990) IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990,1,84

4.IEEE Guide for the use of IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE, New York, IEEE Std. 982.2-1988, 1988.

5. IEEE Standard lossary of Software Engineering Terminology. IEEE, New York, IEEE Std. 610.12–1990, pp. 1–84,1990.

6. C. Kaner, Software engineering metrics: what do they measure and how do we know?, in: 10th International Software Metrics Symposium, vol. 6,2004.

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

8. H. Pham, System Software Reliability, Reliability Engineering Series, Springer Verlag Publisher, London, 2006.

9. Methodology for Software Reliability Prediction and Assessment. TechRep RL-TR-92-95, Rome Laboratory, vol. 1–2, 1992.

10.J.E. Gaffney Jr., C.F. Davis, An approach to estimating software errors and availability, in: Proceedings of 11th Minnowbrook Workshop on Software Reliability, SPC-TR-88-007, version 1.0, July 26–29, 1988, Blue Mountain Lake,NY, 1988.

11. J.E. Gaffney Jr., J. Pietrolewiez, An automated model for software early error prediction (SWEEP), in: Proceedings of 13th Minnowbrook Workshop on Software Reliability, Blue Mountain Lake, NY, 1990.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 5, May 2022

12.A.K. Pandey, N.K. Goyal, Early Software Reliability Prediction, Springer, 2013.
13.D.K. Yadav, S.K. Charurvedi, R.B. Mishra, Early software defects prediction using fuzzy logic, Int. J. Performability Eng. 8 (4) (2012) 399–408.
14. X. Zhang, H. Pham, An analysis of factors affecting software reliability, J. Syst. Softw. 50 (1) (2000) 43–56.
15.M. Li, C. Smidts et al., Ranking software engineering measures related to reliability using expert opinion, in: Proceedings of 11th International Symposium on Software Reliability Engineering (ISSRE), October 08-1, 2000, SanJose, California, 2000, pp. 246–258.
16M. Li, C. Smidts, A ranking of software engineering measures based on expert opinion, IEEE Trans. Softw. Eng. 29 (9) (2003)811–824.
17.Boetticher, G.(2001) An Assessment of Metric Contribution in the Construction of a Neural Network-Based Effort Estimator. 2nd International Workshop on Soft Computing Applied to Software Engineering , Enschede, 8-9February2001,234-235
18.S.H. Kan, Metrics and Models in Software Quality Engineering, Addison Wesley Publications,2002.
19.M. Diaz, J. Sligo, How software process improvement helped Motorola, IEEE Softw. 14 (5) (1997)75–81.
20.M. Agrawal, K. Chari, Software effort, quality and cycle time: a study of CMM level 5 projects, IEEE Trans. Softw. Eng. 33 (2007)145–156.
21.D.K. Yadav, S.K. Charurvedi, R.B. Mishra, Early software defects prediction using fuzzy logic, Int. J. Performability Eng. 8 (4) (2012)399–408.
22.Harikesh Bahadur Yadav, Dilip Kumar Yadav, A fuzzy logic based approach for phase-wise software defects prediction using software metrics – http://dx.doi.org/10.1016/j.infsof.2015.03.001.
23.PromiseRepository <http://promise.site.uottawa.ca/SERepository/datasets-page.html>
24.N.E. Fenton, M. Neil, et al., On the effectiveness of early life cycle defect prediction with Bayesian Nets, Empirical Softw. Eng. 13 (5) (2008) 499–537.