RESEARCH ARTICLE OPEN ACCESS

A predictive approach to estimate software defects density using Probabilistic Neural Networks for the given Software Metrics

T. Ravi Kumar, Dr. T. Srinivasa Rao,Dr.Ch.V.M.K.Hari
Sr.Asst professor, Department of CSE, AITAM Engineering College, Tekkali, (AP), India,
Associate professor, Department of CSE, GIT, GITAM University, Visakhapatnam, (AP).
Sr.Asst.Professor,Dr.V.S.Krishna Govt. College,Visakhapatnam,(AP),India
Corresponding Auther: T. Ravi Kumar

ABSTRACT:
Software plays a very important role in our everyday life.There are many instances which show that even a small defect in the software can cause huge loss to many lives.By evaluating the errors in software in various phases,we can reduce the lateral cost of software.Software quality prediction has been an important arena since the las two decades. Several models and techniques have been proposed and utilized in this gaze.We can recognize the areas which are prone to hazards with the help of logic of quality prediction. In the proposed model, defect density indicator in requirement analysis, design, coding and testing phase is predicted using ten software metrics of these four phases.at the end of each phase the defect density indicator will be taken as an input for the next phase.With the help of ANN and PNN strategies,we have extnded our work. The experimental results are compared with fuzzy, ANN and PNN.Incomparision with ANN and fuzzy,the number of defects can be discovered better with ANN strategy and with PNN strategy there is better prediction in the reliability of mertics of SDLC.Compared to the two implementation methods used on all datasets,ProbabilisticNueral Networks have better reliability.Rather than depending on a singlrtechnique,it would be better to use a scope of software defect prediction models.Experimental results will be performed by utilizing matlab tool.
Keywords: Software Defects,Software Metrics, fuzzy logic, Software error prediction, PNN, ANN, SDLC,MMRE,BMRE.

Date of Submission: 21-06-2018 Date of acceptance: 09-07-2018

I. INTRODUCTION

In the era of Information Technology,it is extrmely important to maintain the customer satisfaction.Maintaining the quality of software is an iportant task for software developers.The needs of reliable software are being emphasized by IEEE and ISO.We can define defects in various ways,though most defects can be considered as deviation from actual details.There are two types of defect data analysis.In order to remove models which show large defect data classes we can use classifictaion and prediction.Obtaining reliable software with in a given amout of time is very tough. A software defect is an error,bug, misstep or error in a program which might produce an inaccurate result.A quality software item with zero or little defects will be produced by the undertaking group.Software defects dependably bring about cost as far as quality and time. Besides distinguishing, redressing defects is a standout amongst the most tedious and costly software forms.At various levels of software design,some errors may or may not be introduced on purpose.However we can predict the defects in a product is released into the market.It is very hard to get the number of defects for the given software,but experience on the SDLC and getting the correct data points will help to predict the software defect density close to the original defects.software defect density is closely tied to the size of the software total number defects is directly proportional to the size of the software.In each phase of SDLC there are many metrics which contribute to the software defects.Software error information is not available in the early stage of the SDLC.Most of the software metrics are also uncertain in nature.Due to this a probablistic approach and expert knowledge are main driving factors in determining the defects.In this paper we use Probabilistic Nueral Network nodes and compute the defect density indictator which in turn provides th number 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 with analysis and results covered in Section 5.Section 6 explains conclusion and future work.

II. RELATED WORK

For many organizations delivering reliable software is an important task .A lot of research has

been done in order to improvethe process of software development and make reliable and efficient software. There are continuous efforts going in predicting the software defects and achieve high level of software system reliability. For estimation and prediction of software reliability a number of models were proposed[1,2].

Review based models :

Rome laboratory[3] model associated software reliability with the reviews some factors related to requirement, design, coding and verification. With process and product characteristics as inputs,a model has been proposed to predict the defect density by Agresti and Evanco [4].Based on reviews,conclusions related to software reliability are drawn in these models

Failure data based models :

Gaffney and Davis have proposedphase based model for predicting software reliability[5,6]. the fault statistics found during the review of various software development phases is the basis for this model.Smidts et al. [7] predicted the software defect density based on the failure data of the software and failure models.With the evolution of UML based models and Bayesian frame works models based on reviews and analysis are become inefficient and research progressed towards the models which can be validated through coding.

Programmable models :

Mohanta et al. [8,9] proposed object oriented model to identify software reliability based on the operation profile and reliabilities of classes. In contrast to other models this model adopted bottoms up approach to find software defects.Pandey and Goyal have proposed early fault prediction model using process maturity and software metrics [10].However this model could not justify the usage of fuzzy profiles for different metrics.Using fuzzy based approachYadav et al.proposed a software defect prediction model and enhanced the above model [11].Defect density in each stage is calculated by this model and passes this DDI to the next stage to evaluate the total defect density. This paper leverages the fuzzy based approach to use neural network approach.

Related work in neural network:

Use of neural networks in the area of predicting software defect density to make reliable software has been emphasized by the models based on the classification, cluster, hybrid, association techniques. Karunanithi et.al [12] explored the use of neural networks to identify the end to end software defect density. It is proved that neural systems are more reliable to find the endpoint software defect density based on distinctive systems like bolster forward NN, Jordan NN, intermittent neural systems. Anticipating software quality by utilizing neural systems proposed by Khoshgaftaar et.al [13]. This model characterized modules into blame or non-blame inclined. They contrasted the Artificial Neural Network show and a non-parametric discriminant model, and found that Neural Network demonstrate has better prescient precision.

Related work in defining software metrics :

For any model accessing software reliability,software metrics are backbone.Thesoftware size is the most important metric influencing the software defect density [14,15].Most of the model uses software size and complexity are the metrics[16].Thirty-two factors have been suggested by Zhang and Pham [17] which have impact on the software reliability in all stages of the software development process. Li et al.[18,19] has provided phase wise ranking for the software metrics which influences the software reliability .On the basis of their influence of given stage of SDLC,this model characterized the software metrics. Artificial neural networks can be leveraged to identify the software defect density and estimate the original defects,on the basis of above literature survey. Next section will introduce the different software metrics used in the proposed mode

III. SOFTWARE METRICS REQUIREMENT

Software metrics are the backbone of any model to decide the software reliability.There are so many metrics with effects the software quality,in each phase of SDLC. We take software metrics as inputs to the proposed model.For this proposed model,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 : Software size is a very important metric in deciding the software reliability. Because the size of the software is directly proportional to the number of defects. This metric can be measured in lines of code (LOC).Usually software sizes will be in KLOC.

ii. Requirement softwaremetrics :the initial stage in the SDLC is Requirement stage.Among so many metrics of the requirement stage, input layer consists of Requirement stability, Fault density and Review Information.

- Requirement stability (RS) : Stable requirement is directly proportional to the software efficiency and reliability. Stable requirement gives a freehand for the designers and testers to concentrate on the freezedrequirements.Unstable requirement disrupts the process in SDLC which in turn can cause the

explosion of the software,If the requirement change requests are more.

● 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. the fault density can be reduced by Continuous requirement analysis and early requirement description.

● Review Information (RI): Requirements review is inversely proportional to the software reliability. Well reviewed requirement will eliminate any defects in the later stages. 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 customers.

iii. Design phase software metrics : Design phase follows requirement definition.As requirement phase defects affects the coding phase need to consider overall defect density of requirement phase is prime input to the design phase. Based on the literature survey mentioned above software complexity and design review are considered as the software metrics.

● Software complexity (SC) : Software complexity is directly proportional to the software defects. As complexity of the software is derived from the tight design constraints.Larger number of decision points can make software program more complex .This is a good to have metric in the design phase because a well executed complex software attracts customers.

● Design Review (DR) In the design phase, design review is to identify the defects or faults occurred. The design document outlines each and every details of requirement specifications and match with the intent of both in the design review.The design phase should be intimated in case of changes.Design review has a capability of eliminating defects from the later stages.

Coding phase metrics : Coding phase will be dependent on the design document. A previous 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.

● Programmer capabilities (PC) : a programmer who is technically well experienced can write a reliable software irrespective of the software complexity. Keeping the above in mind program management will consider programmers with a background of good education from reputed institutes, intelligence and domain knowledge. programmer capabilities are directly proportional to the software reliability.

● Process maturity (PM) : In the SDLC, well established process is very important.A well defined process will reduce the gaps between the teams which are working globally to achieve a reliable software. In order to gauge, screen and enhance the reliability and quality of software process metrics can be utilized [20,21].The process helps in the smooth development flow to achieve targets.

Verification phase software metrics : In SDLC Verification or testing is the critical phase.efore software goes to the customers This phase is useful to find the defects/bug. In most of the cases this phase contributes more towards the software reliability and quality. Staff experience and quality of tests are the prime metrics of this phase.

● Staff experience (SE): For the good execution verification phase requires a lot of resources. A technically well experienced staff contribution to the testing phase can directly impact the software quality. Most organizations around the globe follow a pyramid pattern to balance the experience and knowledge domains

● Quality of testing (QT): To verify the software the tests have to be designed well. As software testing is costly and time consuming effective tests cases will assure a quality software time to market.The software defects will be exposed by well written tests . Tests should cover all the scenarios of the requirement.To ensure the software reliability Good documentation of the issues or tests is required.

IV. PROPOSED MODEL

The proposed model is the probablisitic nueral network model. From "PROMISE Repository" [24] the required data is collected .The weka tool is used to classify this data .In Mat lab 2015a this classified data is given as input. Fuzzy logic is implemented towards the Architecture Design Model shown in fig 1. It predicts the reliability of the software metrics .The model is proposing that the results which are obtained by PNN are more accurate though the results which are obtained by fuzzy logic and ANN are more reliable .The accuracy which is required is obtained and the results are show in results section.

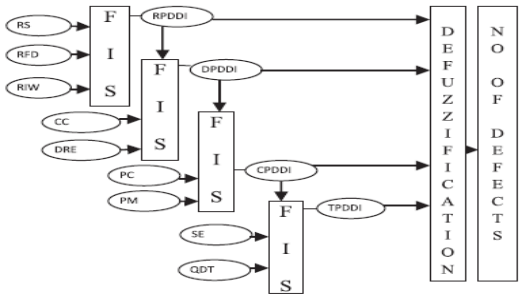

Fig. 1. Architecture Design Model

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

Here PNN defects prediction model is implemented based on software metrics. PNN is implemented towards the architecture design model shown in fig 2. Architectural view of network for requirement gathering as shown in Fig 3.

In PNN there are four layers available. The input will be given to input layer. The four layers of PNN are input layer, pattern layer, summation layer, output layer. We have given RS, RFD, RIW, CC, DRE, PC, PM, SE, QDT, LOC as input to perform the PNN operations. These inputs are processed and given to Pattern layer.

In the identification of number of number of defects process by finding the values of the inputs. The network in the pattern layer will design characters of the given input. The identification of the probability of getting defects will be done in the pattern layer and will be passed to summation layer. In summation layer The Euclidian distances for the original defects will be calculated in the summation layer and the number of defects will be identified.The values of MMRE and BMRE will be calculated on the basis of the identification of number of number of defects. The nuerons present in pattern layer change time to time because they are not characterized specifically in that layer.everytime they take an input they have a different behaviour.the rate of prediction will be increased by doing the above process.The minute defects also can be identified by probability check.In this way ,in the output layer we obtain the outputs.The original defects will be compared in the output layer and the actual number of defects will be finalized.

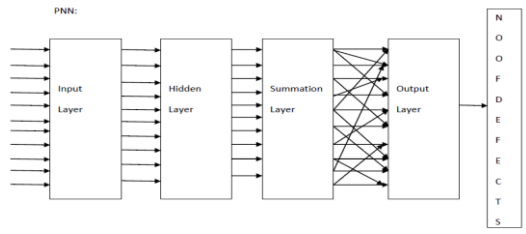

Fig 2. Proposed Architecture Design

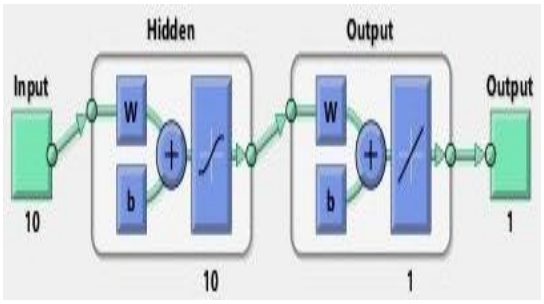

Fig 3. Architectural view of network for requirement gathering phase

V. ANALYSIS AND RESULTS

Analysis

Normalized fuzzy Range

$$= \left[\frac{\text{Minimum value} - \text{Minimum value}}{\text{Maximum value} - \text{Minimum Value}}, \frac{\text{Maximum Value} - \text{Minimum Value}}{\text{Maximum Value} - \text{Minimum Value}}\right]$$

Evaluation measures

To approve the prediction exactness of the proposed display regularly utilized and recommended assessment measures have been taken which are as per the following.[22,23]

i. Mean Magnitude of Relative Error (MMRE): MMRE is the mean of complete calculation errors and a measure of the spread of the variable Z, where Z = estimate/actual

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

Where x_i is the actual value and $\hat{x}_i$ is the estimated value of a variable of interest

ii. Balanced Mean Magnitude of Relative Error (BMMRE): MMRE is unbalanced and assesses overrates in excess of underrates.

For this reason, a balanced mean magnitude of relative error measure is also considered which is as follows:

$$\text{BMMRE} = \frac{1}{m}\sum_{j=1}^{m}\frac{|x_i - \bar{x_i}|}{\min(x_i, \bar{x_i})}$$

The minor value of MMRE and BMMRE specifies improved precision of prediction.

Performing Fuzzy interface and PNN

Case study:
Project no. #
RPDDI: 0.0047
DPDDI: 0.0391
CPDDI: 0.0062
TPDDI: 0.0783

Case study	RPDDI	DPDDI	CPDDI	TPDDI	Actual Defects	Defects prediction using fuzzy	Defects prediction using ANN	Defects prediction using PNN
1	0.0047	0.0391	0.0062	0.0783	89	93	91	89
2	0.0142	0.0168	0.0228	0.0265	100	106	107	107
3	0.0064	0.0357	0.0091	0.00737	51	49	50	51
4	0.0171	0.0228	0.028	0.0356	225	231	225	225
5	0.0036	0.009	0.0083	0.0066	230	240	228	231
6	0.0025	0.0078	0.0065	0.0055	400	393	398	401
7	0.0044	0.0085	0.0072	0.0047	1076	1052	1073	1080
8	0.0468	0.0333	0.0283	0.0126	536	528	537	537
9	0.00389	0.012	0.0145	0.0115	478	476	478	480
10	0.0084	0.0984	0.0133	0.0134	1893	1887	1895	1894
11	0.0598	0.0022	0.014	0.0558	746	739	750	749
12	0.00375	0.0534	0.0388	0.01289	121	115	119	122
13	0.021	0.0129	0.0175	0.0174	392	402	398	389
14	0.00531	0.1004	0.1039	0.0688	73	70	74	73
15	0.0903	0.0076	0.0077	0.01275	707	684	699	702
16	0.0474	0.0301	0.0589	0.03	654	638	657	651
17	0.0059	0.0502	0.0184	0.01431	18	15	17	18
18	0.0126	0.0122	0.1059	0.0993	1357	1343	1355	1358
19	0.189	0.0157	0.0131	0.0564	194	187	188	196
20	0.00971	0.0067	0.0356	0.0149	893	878	881	890

Table 1 Actual Defects and Prediction of Defects using Fuzzy, ANN&PNN

Error rate	Fuzzy Method	ANN method	PNN Method
MMRE	0.37343	0.016397	0.000039
BMRE	0.37534	0.008258	0.000031

Table 2. Error rate of Mean modules

Simulation Results

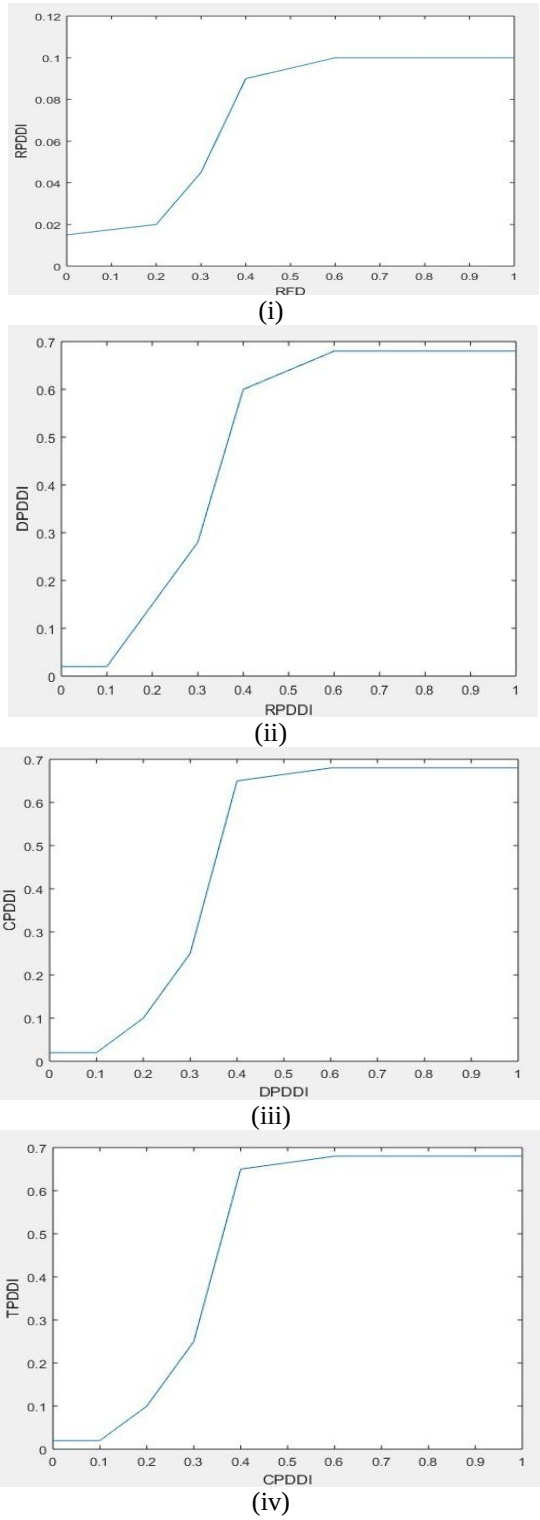

Fig 4. Input metrics towards defect density (i–iv).

Table.1 shows the input metrics,actual outputs and predicted outputs.Here predicted defects are very close to actual defects.As shown in Table.2, PNN based system is predicting the number of defects very close to the actual defects compared to the most of the methods.

Fig.4 shows the sensitivity of requirements fault density ,requirement phase defects density indicator,design phase defect density indicator,and coding phase defect density on the defect density indicator predicted at the end of requirement analysis,design phase,coding phase and testing phase respectively.

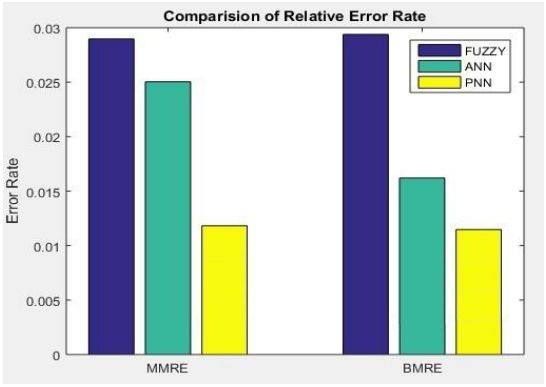

Fig 5. Finding out the defects using BMRE and MMRE.

With respective Fig.5 MMRE and BMRE has more probability of predicting the errors than ANN and Fuzzy.As each and every probable will be calculated so the results will be almost same sometimes.

VI. CONCLUSION:

In this paper Probabilistic Nueral Networks are proposed to identify the software defects in software development life cycle. Input software metrics to this model is chosen based on the literature survey.In order to predict the software defects density indicator during each time frame in SDLC a PNN based model is proposed in this paper. we can compare the experimental results which are obtained by using fuzzy and ANN. The predicted defect density indicators are extremely useful to analyze the defects difficulty in various phases of SDLC of a software project. Defects predicted using this model is more close to the original defects than any other model. ANN has proved to be best with low error rate when errors rates such as MMRE and BMMRE calculated using fuzzy and ANN.PNN has proved to be the best when the same error rates are further calculated by using PNN. Two neural networks, Artificial Neural Network (ANN) and Probabilistic Neural Network (PNN) based software error prediction models utilize software metrics. They analyze the

consequences of these two neural network models with factual techniques. This study shows that PNN is better than ANN and it is robust in nature.

FUTURE WORK:

Further we can extend our work using optimization techniques.By which we can try to reduce the functioning time and reduce the cost of implementation.The number of actual defects identification may also further increase

REFERENCES

[1]. J.D. Musa, A. Iannino, K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill Publishers, New York, 1987.
[2]. H. Pham, System Software Reliability, Reliability Engineering Series, Springer Verlag Publisher, London, 2006.
[3]. Methodology for Software Reliability Prediction and Assessment. TechRep RL-TR-92-95, Rome Laboratory, vol. 1–2, 1992.
[4]. W.W. Agresti, W.M. Evanco, Projecting software defects form analyzing ada design, IEEE Trans. Softw. Eng. 18 (11) (1992) 988–997.
[5]. .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.
[6]. 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.
[7]. C. Smidts, M. Stutzke, R.W. Stoddard, Software reliability modeling: an approach to early reliability prediction, IEEE Trans. Reliab. 47 (3) (1998) 268–278.
[8]. S. Mohanta, G. Vinod, A.K. Ghosh, R. Mall, An approach for early prediction of software reliability, ACM SIGSOFT Softw. Eng. Notes 35 (6) (2010) 1–9.
[9]. . S. Mohanta, G. Vinod, R. Mall, A technique for early prediction of software reliability based on design metrics, Int. J. Syst. Assurance Eng. Manage. 2 (4) (2011) 261–281.
[10]. .A.K. Pandey, N.K. Goyal, Early Software Reliability Prediction, Springer, 2013.
[11]. .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.
[12]. Methodology for Software Reliability Prediction and Assessment. TechRep RL-TR-92-95, Rome aboratory, vol. 1–2, 1992.
[13]. T.M. Khoshgoftaar, J.C. Musson, Predicting software development errors using software complexity metrics, IEEE J. Sel. Areas Commun. 8 (2) (1990) 253–261.
[14]. .J.E. Gaffney Jr., Estimating the number of faults in code, IEEE Trans. Softw. Eng.10 (4) (1984) 141–152.
[15]. M. Lipow, Number of faults per line of code, IEEE Trans. Softw. Eng. Se-8 (4) (1982) 437–439.
[16]. N.E. Fenton, M. Neil, A critique of software defect prediction models, IEEE Trans. Soft. Eng. 25 (5) (1999) 675–689.
[17]. X. Zhang, H. Pham, An analysis of factors affecting software reliability, J. Syst. Softw. 50 (1) (2000) 43–56.
[18]. 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.
[19]. M. Li, C. Smidts, A ranking of software engineering measures based on expert opinion, IEEE Trans. Softw. Eng. 29 (9) (2003) 811–824.
[20]. M. Diaz, J. Sligo, How software process improvement helped Motorola, IEEE Softw. 14 (5) (1997) 75–81.
[21]. . 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.
[22]. D.K. Yadav, S.K. Charurvedi, R.B. Mishra, Early software defects predictionusing fuzzy logic, Int. J. Performability Eng. 8 (4) (2012) 399–408.
[23]. HarikeshBahadurYadav, 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.
[24]. PromiseRepository<http://promise.site.uottawa.ca/SERepository/datasets-page.html>.

Authors

T. Ravi Kumar received M.Tech degree in Computer Science and Engineering from Jawaharlal Nehru technological University Hyderabad, A.P., India. And B.tech in computer Science and Information

Technology from Jawaharlal Nehru Technological University Hyderabad, A.P., India. He is having 10 years of experience in teaching and presently working as Sr Assistant Professor in the Department of Computer Science and Engineering at Aditya Institute of Technology and Management, Tekkali [AITAM], A.P., India. His area of research includes Software Engineering, Fuzzy logic and Software Testing Methodologies.

Dr.T.Srinivasa Rao received B.Tech degree from

GITAM, Andhra University, Visakhapatnam A.P., India. Received M.Tech degree from Andhra University, Visakhapatnam, A.P., India. Received Ph.D. degree from Andhra University, Visakhapatnam, A.P., India. Presently he is working as Associate professor, department of CSE, Gitam Institute of Technology, GITAM University, Visakhapatnam. He is having 17 years of Teaching Experience. His research interest includes wireless communication (WiFi, WiMax), Mobile Ad hoc Networks, Sensor Networks, Neural Networks and fuzzy logic, Communication networks, Data mining, software engineering, Machine Learning.

T. Ravi Kumar "A predictive approach to estimate software defects density using Probabilistic Neural Networks for the given Software Metrics"International Journal of Engineering Research and Applications (IJERA) , vol. 8, no.7, 2018, pp.08-15