International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8509-8517

Software Defects Prediction based on ANN and Fuzzy logic using Software Metrics

T. Ravi Kumar [1]
Sr. Assistant Professor, Department of Computer Science and Engineering,
AITAM Engineering College, Tekkali, Andhra Pradesh, India.
Orcid Id: 0000-0002-0856-3719

Dr. T. Srinivasa Rao [2]
Associate professor, Department of Computer Science and Engineering, GIT,
GITAM University, Visakhapatnam, Andhra Pradesh, India.
Orcid Id: 0000-0002-6263-2666

Abstract

Software Quality analysis is one of the significant criteria required to explore the software life and additionally software reliability. Software quality is been characterized under various parameters. Software risk analysis is one such basis required to distinguish the software reliability. At the point when software is arranged or being created by the sort of software and in addition the endeavors required to build up the software by and large characterizes the software hazard. For example, the accessibility of the required software, equipment, man power all are the prescient hazard factors. In this work, these all hazard factors are characterized under the fuzzy outing the demonstration. In light of this fuzzy estimation to the some weightage is been allotted to these all hazard factors. However conventional metrics approaches, numerous predictable methodologies are inadequate in this regard as well as on a very basic level conflicting. Other than this the paper additionally legitimizes Neural Networks as a superior contrasting option to formal techniques in initiating times of software improvement lifecycle. A fuzzy logic reputable paradigm is proposed for predicting software defect density on individual phases of the SDLC. The perceptive precision of the proposed model is applicable utilizing five real software project data, Approval comes about are appealing and Measures in view of the MMRE and BMMRE fundamentally as the software project estimate growths.

Keywords: Software quality, Design Metrics, Software fault prediction, SDLC, ANN, Fuzzy logic.

INTRODUCTION

A software metric is a standard to assess a computation to which a software framework or process has some possessions. Software metrics is a necessary part of the condition of the-hone in software advancement process. It gives a computable approach to the advancement and approval of models of the software improvement process. Software metrics can be utilized to expand software profitability and quality. Presently a-days clients are indicating software as well as quality metrics scope as a major aspect of their prerequisites. Global guidelines like ISO 9000 [1] and industry models like the Software Engineering Institute's Capability Maturity Model Integrated incorporate quality estimation. The term software metrics implies diverse things to various individuals. Software metrics can differ from extend cost and exertion forecast and displaying, to abscond mapping and main driver analysis, to a particular test scope metric, to PC execution demonstrating. The significance of software metrics to a software advancement process and to a created software Product is an intricate errand that requires study and teach, which passes on learning of the status of the procedure and/or result of software with respect to the objectives to accomplish arrange/stage based imperfection evacuation design. The essential point of software designing is to deliver great effective software requiring little to no effort. With development in size and multifaceted nature of software, administration issues started ruling. An ideal plan system with no bargains e.g. cost and time, for the framework does not build up an ideal plan. [2] The explanation behind this is the adjustments in prerequisites that may happen in later advancement cycles. Such changes may cause plan choices taken before to be less ideal. [3] Design disintegration is unavoidable with the present method for creating software. Refined strategies just contribute by postponing the minute that a framework should be pulled back or resigned. These methodologies don't address the essential issues that reason Design disintegration and makes framework inconsistent [4]. Part based plan is relied upon to strongly affect the quality of software advancement: Due to the effortlessness, the software improvement accelerates. The shorter improvement time brings about decreased expenses. The extensibility and resolvability of software frameworks is enhanced, on the grounds that segments can adaptably be substituted by another segment that fulfills the necessities. Software segments upgrade the reliability of the software, as they are enhanced,

tried and repaired over years [5]. Software Reliability Engineering (SRE) is a training that encourages one create software that is more solid, and create it speedier and substantially less expensive. It is a demonstrated standard and best practice that is for the most part pertinent to frameworks that incorporate software [6]. Software Reliability Engineering works by quantitatively portraying and applying two things about the Product: i) the anticipated relative utilization of its capacities and ii) its required significant quality attributes. The real quality attributes are reliability, accessibility, conveyance date and life-cycle cost. In applying software reliability designing, one can shift the relative accentuation on these elements. [7]

RELATED WORK

The developing complexities of software and expanding interest of dependable software have prompted the advance of persistent research in the territories of viable software reliability evaluation. Software imperfection prediction is not another thing in software building area. In this segment, some imperative commitments around there are displayed.

A. Software Defect Prediction Based on Classification Techniques

Karunanithi et.al [8] exhibited the neural system display for software reliability prediction and found that neural system models are preferable at endpoint prediction over investigative models. They utilized distinctive systems like bolster forward NN, Jordan NN, intermittent neural systems.

Khoshgaftaar et.al [9] presented the utilization of the neural system as an instrument for anticipating software quality of a substantial media transmission system, characterizing 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.

Kanmani et.al [10] presented two neural system based blame prediction models utilizing object arranged metrics and contrasted the outcomes and two measurable models utilizing five quality traits and inferred that neural systems improve.

B. Software Defect Prediction Based on Clustering Techniques

In the paper [11], creators proposed a novel software defect prediction technique in view of useful groups of projects to enhance the execution. Until at that point, most techniques proposed toward this path anticipate absconds by class or record. Tests completed presumed that bunch based models can altogether enhance the review from 31.6 % to 99.2% and accuracy from 73.8 % to 91.6%.

In the paper [12], k-implies based bunching approach has been utilized for finding the blame inclination of the Object arranged systems and found that k-implies based grouping techniques indicates 62.4% exactness. It likewise demonstrated high estimation of likelihood of location and low estimation of likelihood of false cautions. This investigation affirms the achievability and value of k implies based software blame prediction models.

C. Software Defect Prediction Based on Association control mining

In [13], analysts proposed prediction of imperfection affiliation and defect adjustment technique in light of affiliation control mining strategies. The proposed strategies were connected to abscond data comprising of more than 200 undertakings more than 15 years. It was finished up from trial comes about that exactness accomplished is high for both defect affiliation prediction and imperfection adjustment prediction. The outcomes got were likewise contrasted and PART, C4.5 and Naive Bayes technique and demonstrated the precision change by 23 percent.

This proposed demonstrate was assessed on open source datasets and contrasted with comparable existing methodologies and found that this model over performed for the majority of the current machine learning based techniques for imperfection prediction.

D. Software Defect Prediction Based on Hybrid Approach

In the paper[14], a cross breed approach in light of K-Means Clustering and bolster forward neural system has been proposed and it was discovered that execution is better if there should be an occurrence of this half and half approach as contrasted and the current methodologies as far as exactness , mean total blunder and root mean square mistake esteems.

Mixture blame inclined module prediction technique was presented that consolidates affiliation control mining with calculated relapse investigation [15], [16]. On the off chance that a module fulfills the preface of one of the chose rules, the module is arranged by run as either blame inclined or not. Something else, the module is characterized by the strategic relapse. The prediction execution of this model was assessed and contrasted and three other blame inclined modules in view of strategic relapse demonstrate, straight discriminant model and characterization tree. The trial comes about indicated change in execution when contrasted with regular strategies. [17, 18]

SOFTWARE RELIABILITY

Reliability might be characterized as the likelihood of a thing to play out a required capacity under expressed conditions for

a predefined timeframe. Software Reliability is characterized as the likelihood of the disappointment free software operation for a predefined timeframe in a predetermined situation. Unreliability of any Product comes because of the disappointments or nearness of issues in the framework. The unreliability of software is essentially because of bugs or configuration Faults in the software. It happens just when framework is being used and are not gone before by notices. [19, 20]

Software Reliability Measurement Techniques

The decision of which metric is to be utilized relies on the sort of framework to which it applies and the necessities of the application area. Measuring the software reliability is a troublesome issue since we don't have a decent comprehension about the idea of software. It is hard to locate a reasonable approach to gauge software reliability, and the vast majority of the perspectives identified with software reliability. Indeed, even the software sizes have no uniform definition. On the off chance that we can't quantify the reliability straightforwardly, something can be measured that mirrors the qualities identified with reliability. Some reliability metrics which can be utilized to measure the reliability of the software Product are:-

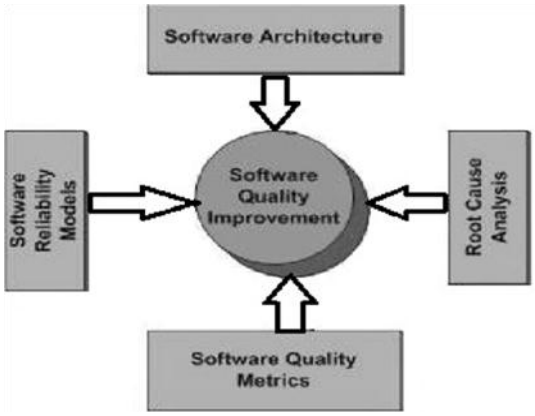


Figure 1: Software Quality Improvement Factors

Product Metrics

Product metrics are those which are utilized to construct the ancient rarities i.e. prerequisite detail archives, framework configuration reports and so forth. These metrics help in evaluation if the Product is sufficient through reports on characteristics like ease of use, reliability, practicality and convenience. In this estimations are taken from the genuine body of the source code.

▪ Software estimate is believed to be intelligent of multifaceted nature, improvement exertion and reliability. Lines of Code (LOC), or LOC in thousands (KLOC), is an instinctive starting way to deal with measuring software estimate. The premise of LOC is that program length can be utilized as an indicator of program qualities, for example, exertion &ease of upkeep. It is a measure of the useful multifaceted nature of the program and is free of the programming dialect.

- Function point metric is a strategy to quantify the usefulness of a proposed software improvement in view of the tally of data sources, yields, ace records, asks, and interfaces.
- Test scope metric gauge Fault and reliability by performing tests on software Products, expecting that software reliability is a component of the bit of software that is effectively checked or tried.
- Complexity is straightforwardly identified with software reliability, so speaking to multifaceted nature is critical. Many-sided quality arranged metrics is a technique for deciding the multifaceted nature of a program's control structure, by disentangling the code into a graphical portrayal. Agent metric is McCabe's Complexity Metric.
- Quality metrics measures the quality at different phases of software Product advancement. An essential quality metric is deformity evacuation effectiveness (DRE). DRE gives a measure of quality due to different quality confirmation and control exercises connected all through the advancement procedure. [21, 22]

Project Management Metrics

Project metrics portray the project attributes and execution. In the event that there is great administration of project by the software engineer then this assistance us to accomplish better Products. Relationship exists between the advancement procedure and the capacity to finish extends on time and inside the coveted quality destinations. Cost increment when engineers utilize deficient procedures. Higher reliability can be accomplished by utilizing better advancement process, chance administration process, arrangement administration process. These metrics tells about:-

- Number of software designers
- Staffing design over the life-cycle of the software
- Cost and timetable
- Productivity

Process Metrics

Process metrics evaluate helpful traits of the software improvement process and its condition. They tell if the procedure is working ideally as they provide details regarding qualities like process duration and revise time. The objective of process metric is to do the correct employment on first time through the procedure. The quality of the Product is an immediate capacity of the procedure. So process metrics can be utilized to gauge, screen and enhance the reliability and

quality of software. Process metrics portray the adequacy and quality of the procedures that deliver the software Product. Illustrations are:

- Effort required simultaneously
- Time to deliver the Product
- Effectiveness of deformity evacuation amid improvement
- Number of deformities found amid testing
- Development of the procedure

Fault and Failure Metrics

A Fault is a deformity in a program which emerges when software engineer makes a mistake and causes disappointment when executed under specific conditions. These metrics are utilized to decide the disappointment free execution software.

To accomplish this objective, number of shortcomings found amid testing and the disappointments or different issues which are accounted for by the client after conveyance are gathered, condensed and dissected. Disappointment metrics depend on client data in regards to disappointments found after arrival of the software. The disappointment information gathered is in this manner used to compute disappointment thickness, Mean Time between Failures (MTBF) or different parameters to quantify or anticipate software reliability. [23, 24]

The Metrics are utilized to enhance the reliability of the framework by distinguishing the ranges of prerequisites. The diverse sorts of software metrics that are utilized are:-

Prerequisite Reliability Metric

Prerequisites demonstrate what highlights the software must contain. It determine the usefulness that must be incorporated into the software. The necessities must be composed with the end goal that is no misconception between the designer and the customer. The necessities must contain legitimate structure to stay away from the loss of important data. The prerequisites ought to be careful and in a nitty gritty way so it is simple for the plan stage. The necessities ought not contain deficient data. Necessity Reliability metrics assesses the above said quality elements of the required record.

Design and Code Reliability Metric

The quality calculates that exists Design and coding design are intricacy, size and seclusion. Complex modules are hard to comprehend and there is high likelihood of happening errors. The reliability will diminish if modules have a blend of high multifaceted nature and expansive size or high many-sided quality and little size. These metrics are likewise relevant to question arranged code, however in this, extra metrics are required to assess the quality.

Testing Reliability Metric

These metrics utilize two ways to deal with assess the reliability. In the first place it guarantees that the framework is outfitted with the capacities that are determined in the prerequisites. Along these lines, the mistakes because of the absence of usefulness diminishes. Second approach is assessing the code, finding the errors and settling them. To guarantee that the framework contains the usefulness determined, test designs are composed that contain different experiments. Each experiment depends on one framework state and tests a few capacities that depend on a related arrangement of necessities The target of a powerful check program is to guarantee that each prerequisite is tried, the suggestion being that if the framework breezes through the test, the necessity's usefulness in incorporated into the conveyed framework.

PROPOSED WORK

In this dataset is collected from "PROMISE Repository" [25] and done the classification using weka tool and finally that classified data is used as dataset in Mat lab 2015a. Below there are few Metrics shown in Figure 3,4,5,6,7,8,9,10,11 taken as inputs. Fuzzy logic is implemented towards the Architecture Design Model shown in Figure 1 it predict the reliability of the software metrics. The Results which we got while using Fuzzy logic is reliable but we are proposing ANN to get the more Accuracy. Finally achieved the required accuracy and those all are show in results section.

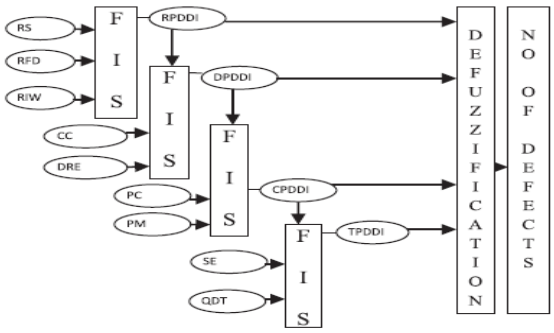


Figure 1: Architecture Design Model [24]

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

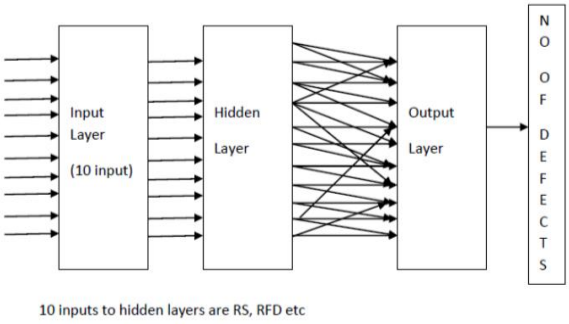


Figure 2: Proposed Architecture Design

Guaranteeing the reliability of a software extend is imperative to all gatherings included including Managers, Marketing, Programmers, and Customers. Untrustworthy frameworks can affect software designers and customers by basically being an inconvenience, by costing time and cash, or most dire outcome imaginable, by costing single or numerous lives. Everyone engaged with the software procedure has explanations behind craving dependable software.

Utilization of ANN enhanced affiliation mining to anticipate software reliability has been proposed. Software reliability appraisal has been a crucial factor to portray the quality of any software item quantitatively amid testing stage. The work depends on the software disappointments or the imperfections and on which the diagnostic choice will be drawn utilizing ANN. It just takes disappointment history as information and Predicts future disappointments. The contribution to the proposed technique is software execution time, while yield of the framework is anticipated as number of disappointments. The disappointments or the errors will be characterized with various weights. In this way, here we investigate the materialness of ANN for better expectation of reliability in a practical situation and present an evaluation strategy for software reliability development utilizing connectionist demonstrate.

Testing stage software metrics

i. Staff Experience (SE): Testing staff having a sound specialized foundation and experience greatly affects the test quality. Staffs engaged with software testing are damaging in nature and attempt their best to discover software deformity. In this way, aptitudes and experience of test group greatly affects the software quality.

ii. Quality of Documented Test Cases (QDT): Software testing is exorbitant and tedious, in this manner powerful experiments are should have been produced. Software test cases are particulars of the contributions to the test and the normal yield from the framework in addition to an announcement of what is being tried. The experiments are intended to uncover deserts. A decent experiment is one that has a high likelihood to uncover software abandons. [23,24]

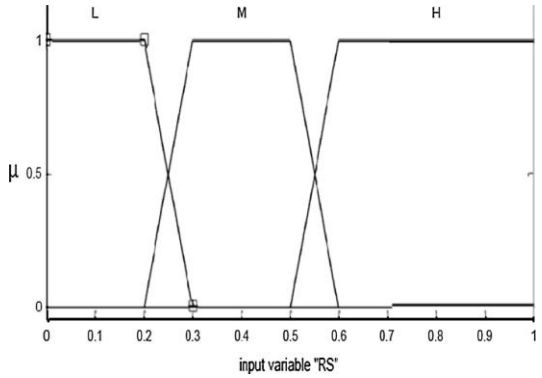


Figure 2: Input Metrics 1

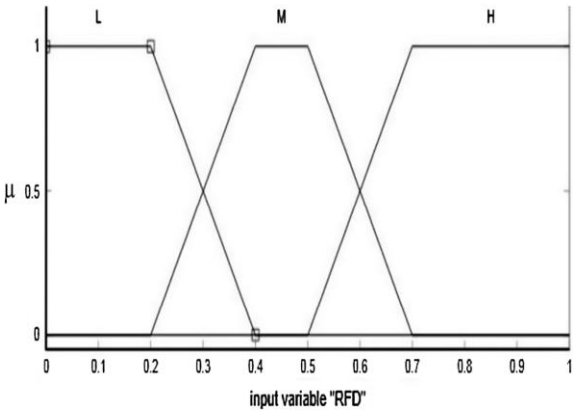


Figure 3: Input Metrics 2

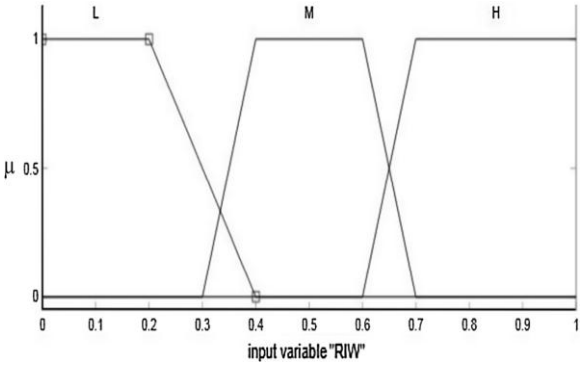


Figure 4: Input Metrics 3

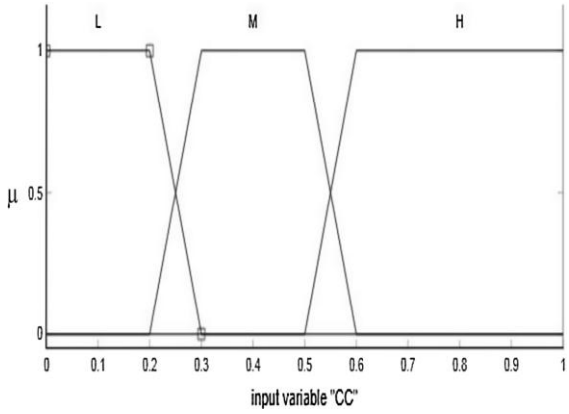


Figure 5: Input Metrics 4

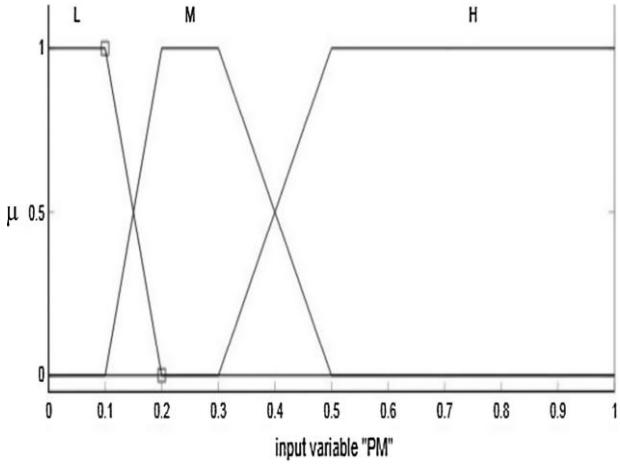


Figure 8: Input Metrics 7

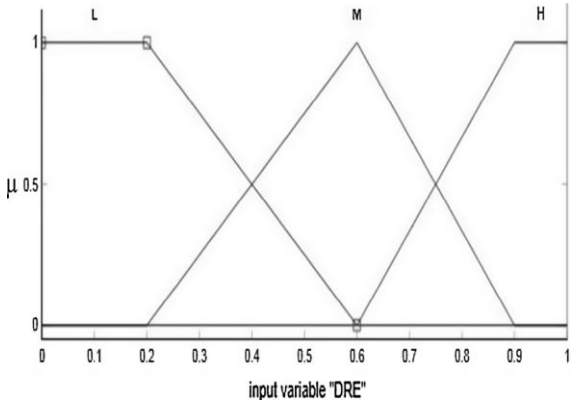


Figure 6: Input Metrics 5

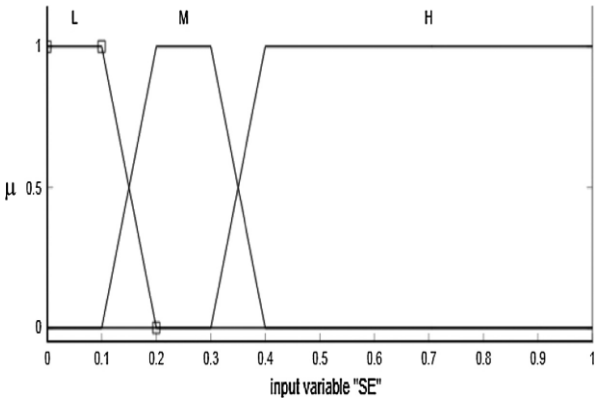


Figure 9: Input Metrics 8

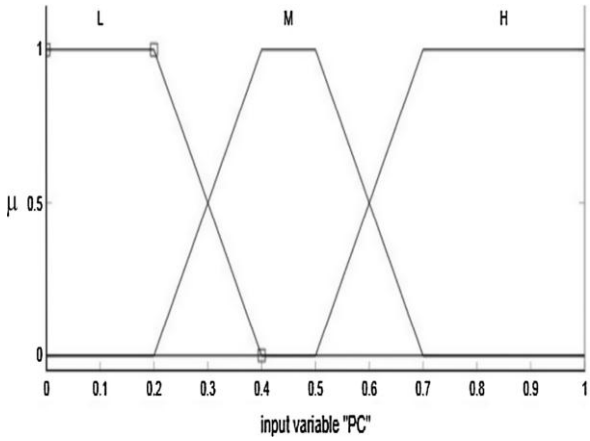


Figure 7: Input Metrics 6

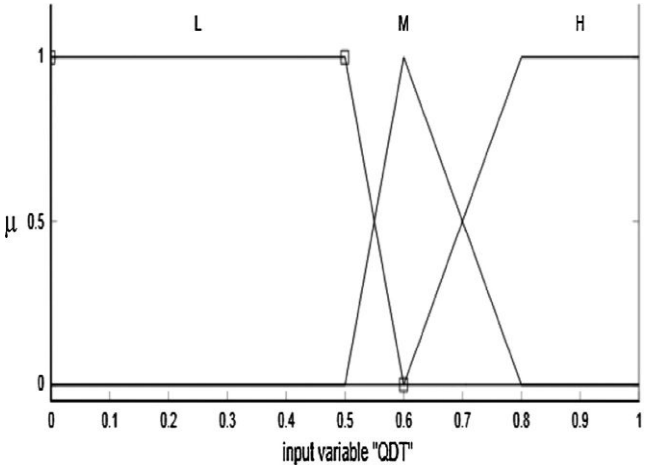


Figure 10: Input Metrics 9

ANALYSIS AND RESULTS

Analysis

$$Normalized\ fuzzy\ Range = [\frac{Minimum\ value - Minimum\ value}{Maximum\ value - Minimum\ Value},$$

$$\frac{Maximum\ Value - Minimum\ Value}{Maximum\ Value - Minimum\ Value}]$$

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.[23,24]

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

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

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:

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

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

Performing Fuzzy interface and ANN

Case study no.: 1

Project no. # [5]

RPDDI: 0.0142

DPDDI: 0.0168

CPDDI: 0.0228

TPDDI: 0.0265

Table 1: Actual Defects and Prediction of Defects using Fuzzy, ANN

Case study	RPDDI	DPDDI	CPDDI	TPDDI	Actual Defects	Defects prediction using fuzzy	Defects prediction using ANN
1	0.0142	0.0168	0.0228	0.0265	100	106.699	107.077
2	0.0171	0.0228	0.0280	0.0356	225	231.1171	225.0249
3	0.0036	0.0090	0.0083	0.0066	230	240.000	229.987
4	0.0025	0.0078	0.0065	0.0055	400	390.7233	400.001
5	0.0044	0.0085	0.0072	0.0047	1000	980.000	994.079

Table 2: Error rate of Mean modules

Error rate	Fuzzy Method	ANN method
MMRE	0.36343	0.015397
BMRE	0.36534	0.007258

Simulation Results

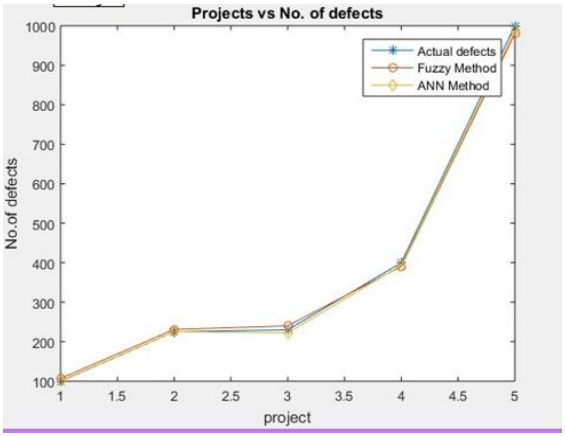
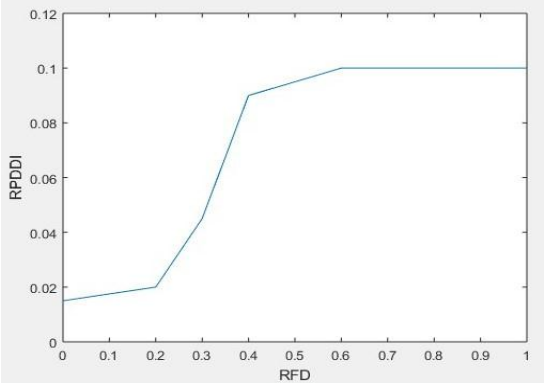
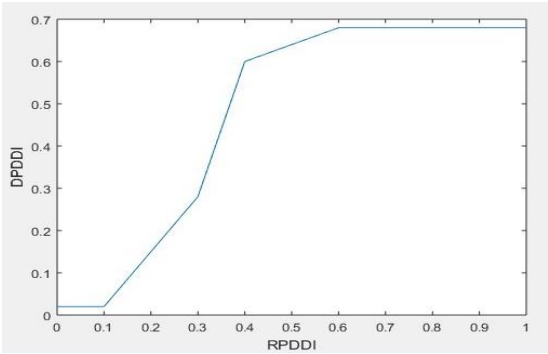


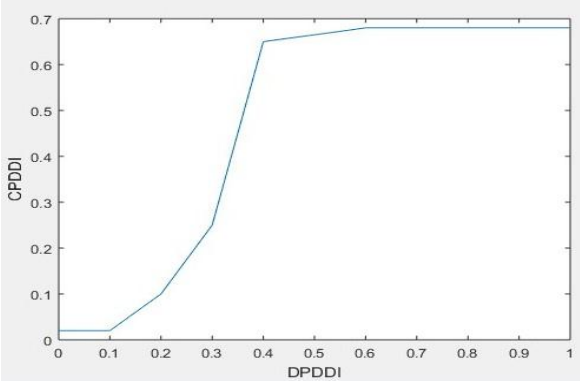
Figure 11: Results on Project size and No. of defects



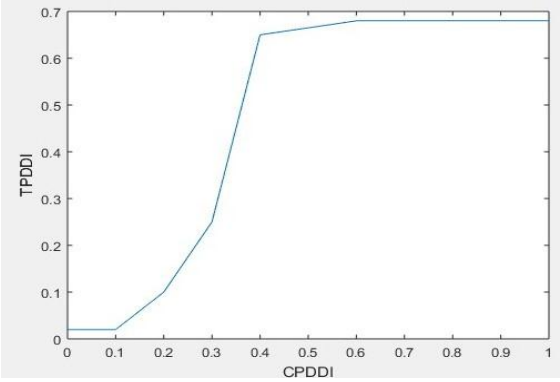
(i)



(ii)



(iii)



(iv)

Figure 12: Input metrics towards defect density (i–iv).

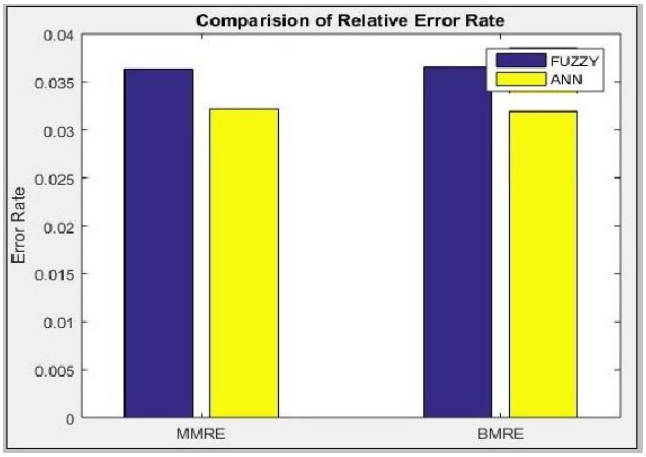


Figure 13: Procreation of defects using BMRE and MMRE.

CONCLUSION

In this paper, a fuzzy logic based model is proposed for predicting software defects density indicator at each phase of SDLC. The fuzzy rules are applied for the software metric data. Defuzzification is performed to obtain the total number of defects based on the given number of actual defects. The proposed model considers only reliability relevant software metrics of each phase of SDLC. The error values are calculate and the prediction of defects is been obtain using fuzzy technique. In order to increase the rate of prediction of defects we further implement our work using artificial neural networks. The experimental results using fuzzy and ANN are been compared using matlab tool. By the obtained results it is shown that the prediction rate is more accurate by using ANN method. The predicted defect density indicators are very helpful to analyze the defects severity in different artifacts of SDLC of a software project. Software developer may easily detect the errors in representation of information to take correct decisions regarding the software development work. The error rate such as MMRE and BMMRE calculated using fuzzy and ANN for which ANN proved to be best with low error rate.

FUTURE WORK

Further we can extend our work using convolution neural networks. By which we can try to reduce the functioning time and reduce the cost of implementation. The no of actual defects identification may also further increase.

REFERENCES

[1] M.R. Lyu, Handbook of Software Reliability Engineering, vol. 222, IEEE Computer Society Press, CA, 1996.

[2] IEEE Guide for the use of IEEE Standard Dictionary

of Measures to Produce Reliable Software. IEEE, New York, IEEE Std. 982.2-1988, 1988.

[3] IEEE Standard Glossary of Software Engineering Terminology. IEEE, New York, IEEE Std. 610.12–1990, pp. 1–84, 1990.

[4] K.Y. Cai, C.Y. Wem, M.L. Zhang, A critical review on software reliability modeling, Reliab. Eng. Syst. safety 32 (3) (1991) 357–371.

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

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

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

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

[9] W.W. Agresti, W.M. Evanco, Projecting software defects form analyzing ada design, IEEE Trans. Softw. Eng. 18 (11) (1992) 988–997.

[10] C. Wholin, P. Runeson, Defect content estimations from review data, in: Proceedings of 20th International Conference on software Engineering, 1998, pp. 400–409.

[11] C. Smidts, M. Stutzke, R.W. Stoddard, Software reliability modeling: an approach to early reliability prediction, IEEE Trans. Reliab. 47 (3) (1998) 268–278.

[12] V. Cortellesa, H. Singh, B. Cukic, Early reliability assessment of UML based software models, in: Proceedings of the 3rd International Workshop on Software and Performance, 2002, pp. 302–309.

[13] 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.

[14] 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.

[15] J.E. Gaffney Jr., Estimating the number of faults in code, IEEE Trans. Softw. Eng. 10 (4) (1984) 141–152.

[16] M. Lipow, Number of faults per line of code, IEEE Trans. Softw. Eng. Se-8 (4) (1982) 437–439.

[17] T.M. Khoshgoftaar, J.C. Musson, Predicting software development errors using software complexity metrics, IEEE J. Sel. Areas Commun. 8 (2) (1990) 253–261.

[18] N.E. Fenton, M. Neil, A critique of software defect prediction models, IEEE Trans. Softw. Eng. 25 (5) (1999) 675–689.

[19] N.E. Fenton, M. Neil, et al., Predicting software defects in varying development lifecycles using Bayesian nets, Inf. Softw. Technol. 49 (1) (2007) 32–43.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Promise Repository <http://promise.site.uottawa.ca/SERepository/datasets-page.html>.