International Journal of Research

International Journal of Research (IJR)

e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

Effective Assessment of Software Reliability by Using Neuro-Fuzzy System

Bonthu Kotaiah[1]& R.A. Khan[2]
[1] Assistant Professor, Maulana Azad National Urdu University, Hyderabad, India
[2] Professor, Babasaheb Bhimrao Ambedkar University, Lucknow, India
[1]kotaiah_bonthuklce@yahoo.com, [2]khanraees@yahoo.com

ABSTRACT
Software reliability is defined as the probability of software to deliver correct service over a period of time under a specified environment. This is becoming more and more important in various software organizations to discover the faults that occur commonly during development process. As the demand of the software application programs increases the quality becomes higher and higher and the reliability of these software becomes more essential. Hence Software reliability is mentioned to be as the one of the important factor during development. Many analytical models were being proposed over the years for assessing the reliability of a software system and for modeling the growth trends of software reliability with different capabilities of prediction at different testing phases. A Neuro Fuzzy based software reliability (SR) model is presented to estimate and assess the quality. Multiple datasets containing software failures are applied to the proposed model. These datasets are obtained from several software projects. Then it is observed that the results obtained indicate a significant improvement in performance by using neural fuzzy model over conventional statistical models (Fuzzy Model) based on non homogeneous Poisson process.

1. INTRODUCTION

1.1. BACKGROUND

Dependency on computer aided systems is increasing rapidly day by day and the software systems operating in it .However this quality of service by the system is degraded by some software failures or fails to meet the required level of performance this make many of the people to strike off these Softwares. This model attempt to match product properties with the software quality attributes. Hence if a company is to develop high quality software, it is important to employ some efforts on software reliability and usability. However, this thesis focuses only on software reliability based models.

1.2 Software Reliability

The American Institute of Aeronautics and Astronautics (AIAA) defines SRE as "the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems"[8].

Three kinds of identifiers for Software Reliability. They are a) Probability of failure free operation over a specified time interval. b) Mean time to failure (MTTF) the predicted elapsed time between inherent failures of a system during operation. c) Expected number of failures per unit time interval termed failure intensity.

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

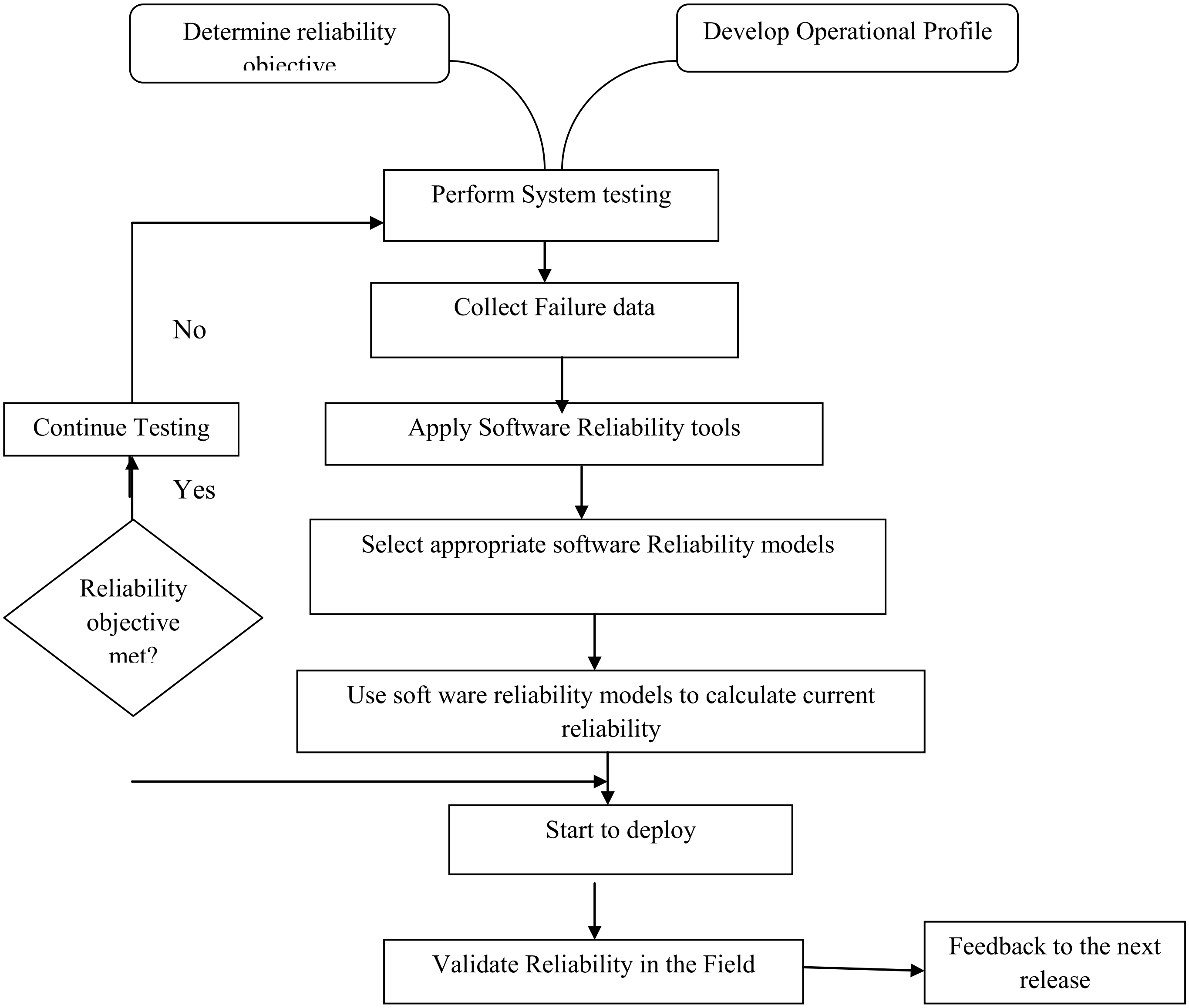

Figure 1: Software reliability engineering Process Overview [9]

Here in our work, a Neuro Fuzzy based SRGM is proposed. In order to test the accuracy of proposed model, real failure data of a software project is required. However, it is a very time consuming process to carryout software testing for a real project and could even take years. This is not feasible within the available time and thus secondary data which have already been collected and published.

1.3 Neuro Fuzzy Models

The idea of a Neuro Fuzzy system is to find the parameters of a fuzzy system by means of learning methods obtained from neural networks. In this chapter the basic properties of Neuro Fuzzy systems are discussed. The learning techniques that can be used to create fuzzy systems for data; a common way to apply a learning algorithm to a fuzzy system is to represent it in a special neural-network-like architecture. Then a learning algorithm – such as back propagation – is used to train the system. They cannot be applied directly to a fuzzy system, because the functions used in the inference process are usually not differentiable. There are two solutions to this problem:

a) Replace the functions used in the fuzzy system (like min and max) by differentiable functions, or b) Do not use a gradient-based neural learning algorithm but a better-suited procedure.

The Structure of Adaptive Neuro Fuzzy Inference System is shown in fig. 4.

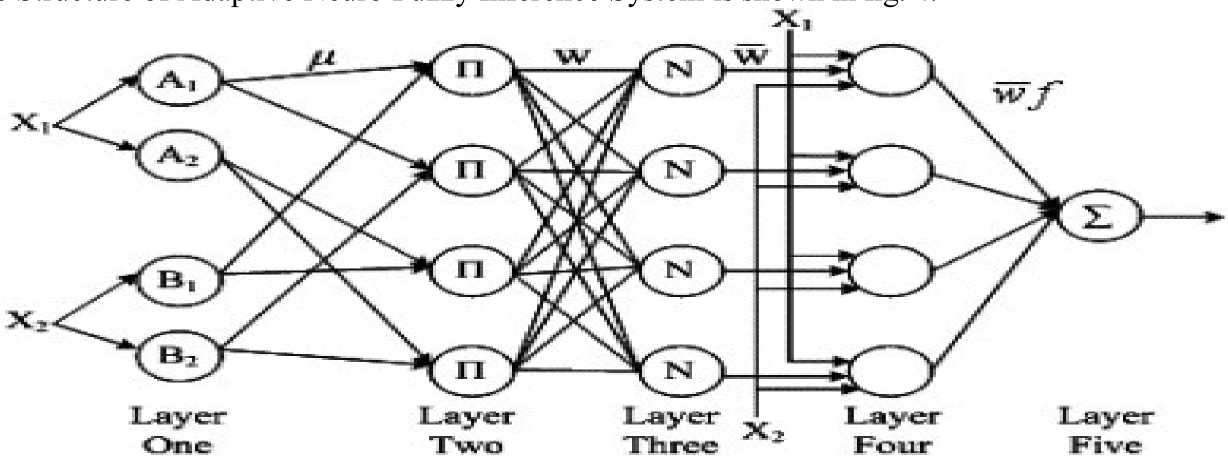

Figure 2: Structure of Adaptive Neuro Fuzzy Inference System.

1.4 OBJECTIVES

The following objectives are set in our research:

- To collect the set of dataset program from running software with the appropriate runtime errors that is useful for the assessment.
- To formulate a theoretical analysis for the evaluation of the metrics those are used for assessment and develop model.
- To identify how availability and MTBF relates with the software reliability
- Calculate the metrics with the given dataset both analytically and programmatically.
- To train the neural network with some collected software reliability parameters (at design phase of SDLC) mapped to numerical data and are loaded into neural network at input layer.
- Assess and evaluate the performance of the trained network for software reliability at the design level with some numerically approximated values by using fuzzy membership function (sigmoid).
- The approximated Software Reliability is compared against the expected reliability approximation.
- The Neuro Fuzzy model was to adjust at the input layer has to minimize the difference between actual and expected values of reliability.
- Our proposed model performance compared against conventional FIS (Fuzzy Inference system) models based on evaluation and validation metrics to prove that our proposed model is the promising one than the others.

2. Software Reliability Assessment using Neuro Fuzzy System

International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

2.1 Proposed Model

Figure below shows the proposed model of the research analysis, where the parameters concerned with the reliability assessment were given as the inputs for network. Based upon the outcome of validation the assessment will be finalized. A generalized block diagram is shown in the below figure, the parameters used were discussed in the next chapter. In this research work a Neuro Fuzzy interference model is designed for the assessment of reliability of a software growth model, the algorithm mainly focuses on MTBF and Availability which is analyzed and calculated theoretically and practically.

Fuzzy rules employed for the proposed model

- *If MTBF (Mean time Between Failure) >0.8 & availability >0.8 then reliability is very high*
- *If 0.7<MTBF <0.8 & 0.7< availability <0.8 then reliability is high*
- *If 0.6<MTBF <0.7 & 0.6<availability <0.7 then reliability is moderate*
- *If 0.5<MTBF <0.6 & 0.5< availability <0.6 then reliability is low*
- *If 0.4<MTBF <0.3 & 0.4<availability <0.3 then reliability is very low*

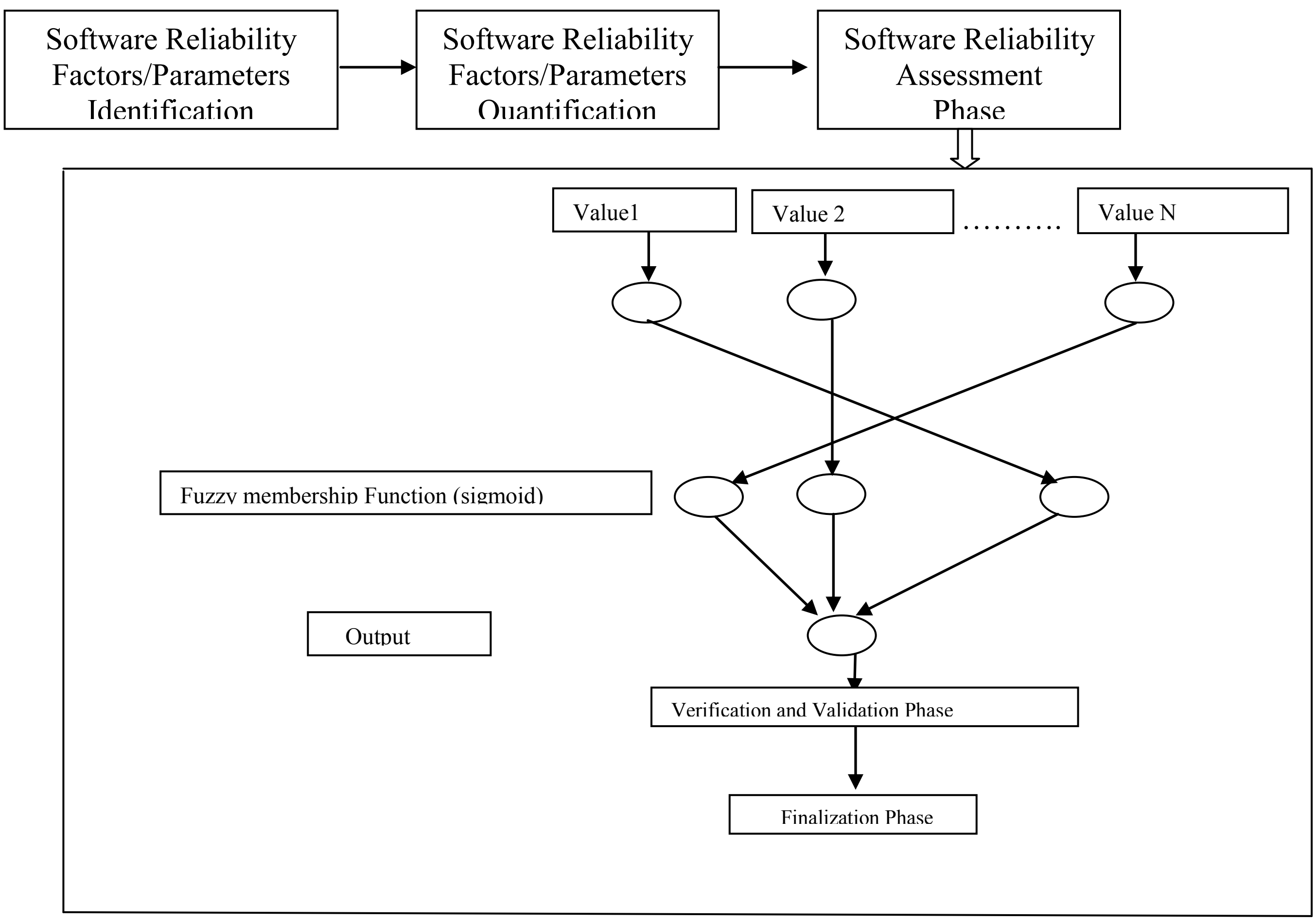

Figure 3: Proposed Model of software reliability estimation

2.2 Mathematical approximation of proposal model metrics

Formula: $Ca (x_i) = C (a) - h f(a)$

Where, C (a) = Set of Measured values. 'h' can be derived by,

$x_1 + x_0 n h$

International Journal of Research
International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 08, August 2015
Available at http://internationaljournalofresearch.org

Where, n= no. of values in the dataset. $x_0 = 0$ and $x_1 = 1$ (since the probability ranges from 0 to 1). Here 'x' is MTBF. f(a) can be function, denoted as
f(a)=MTBF/(1+MTBF)
Ca (x_i) is the set of values to be approximated.

Procedure for 'h' Calculation:
Let us take, $x_0 = 0$ and $x_1 = 1$ then, 1= 0 + 17 * h
h= 1/17 = 0.058
Iterations: Perform at least 5 to 10 iterations to arrive at good approximated software reliability value.
At every iteration, to calculate % of Reliability, use the following formula
% of Reliability = (Average of Approximated values)/ (Average of Measured values) * 100
At final iteration, if we got 99.99% or 99.8% or 99.7%, then we can say that it is good approximation.

2.3 REVIEW & REVISIONS

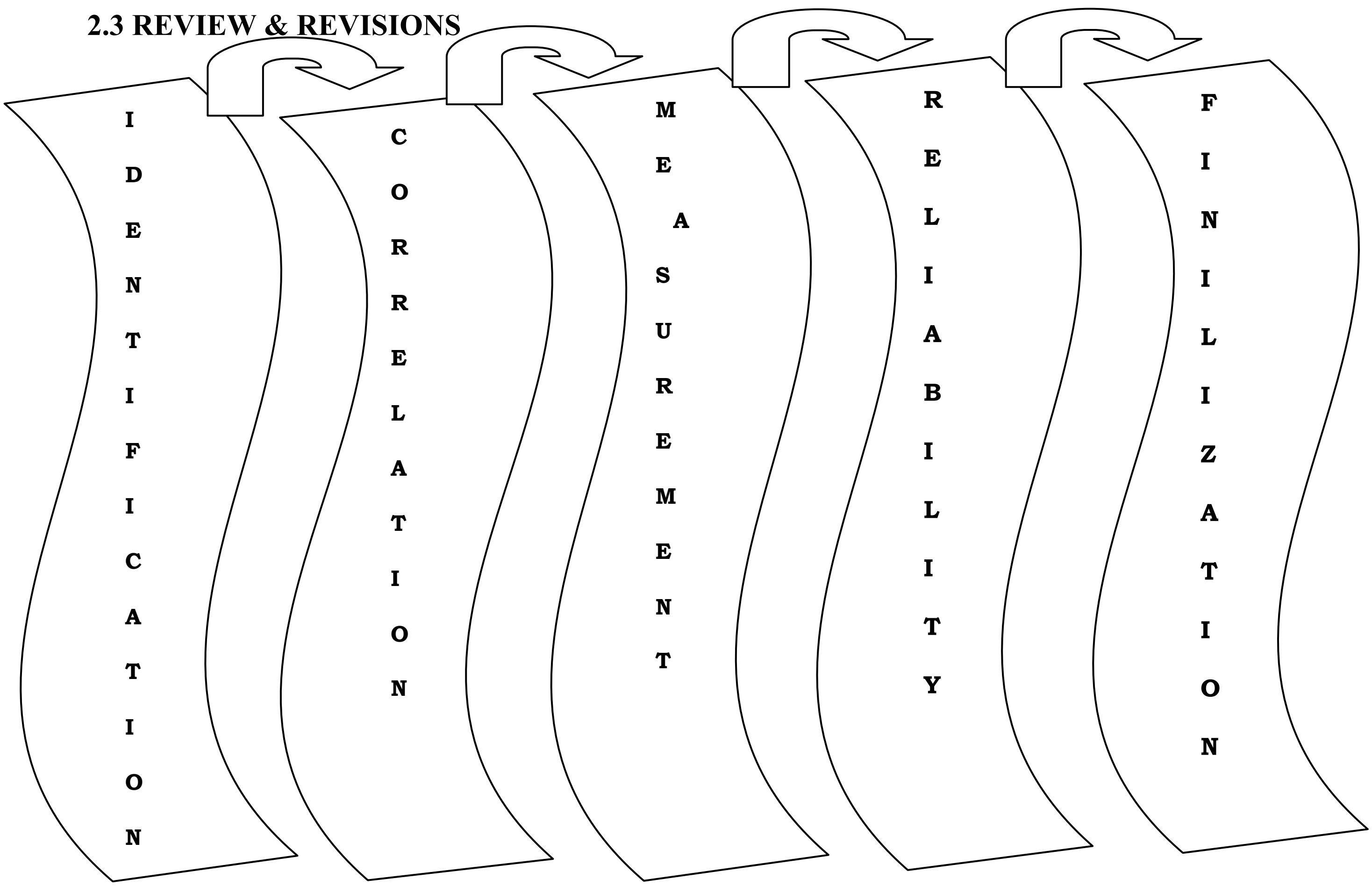

Figure 4: Process flow of the proposed approach

Recognition:
At this stage the objectives are
1. To identify the reliability factors.
2. To evaluate a mathematical analysis for the approximation constraints

Correlation:

At this stage the objectives are
1. To identify the reliability factors with availability and MTBF
2. To evaluate a mathematical analysis for the relationship

Measurement:
At this stage the objectives are
1. To assess the metrics for the estimation of reliability
2. To validate the metrics

Reliability:
At this stage the objectives are
1. To estimate reliability
2. To validate the reliability

Finalization:
At this stage the objectives are
1. To incorporate the changes and suggestions
2. To finalize the metrics for evaluation

3. EMPIRICAL VALIDATION

3.1 INTRODUCTION

Below figure the practical implementation of the FIS model in MATLAB software tool using FIS. The NF system is trained using a hybrid learning algorithm using both least squares method and back propagation algorithm. In the forward pass the consequent parameters are identified using least squares and in the backward pass the premise parameters are identified using back propagation [12]. The trained NF system is then tested for the fifteen inputs

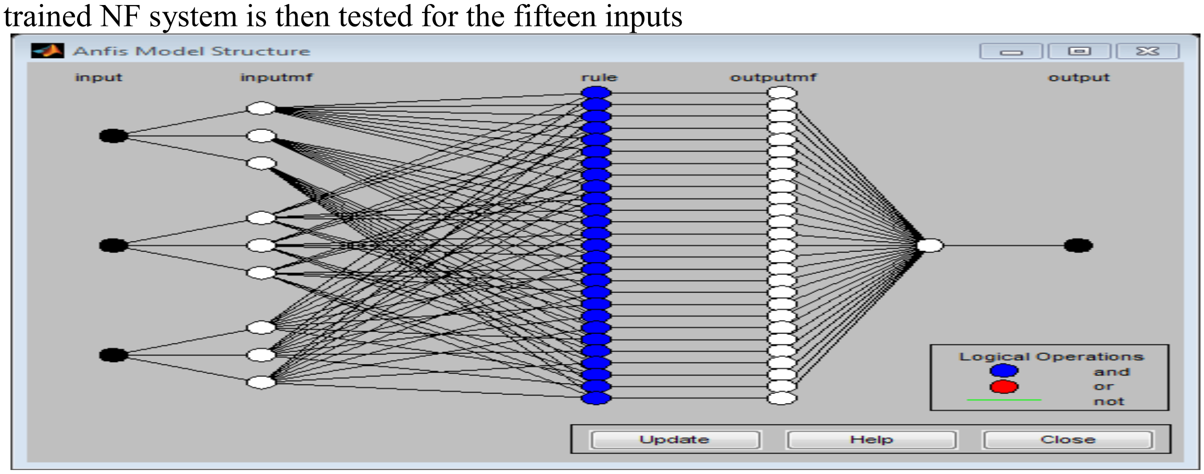

Figure 5: Real time design of Neuro Fuzzy structure

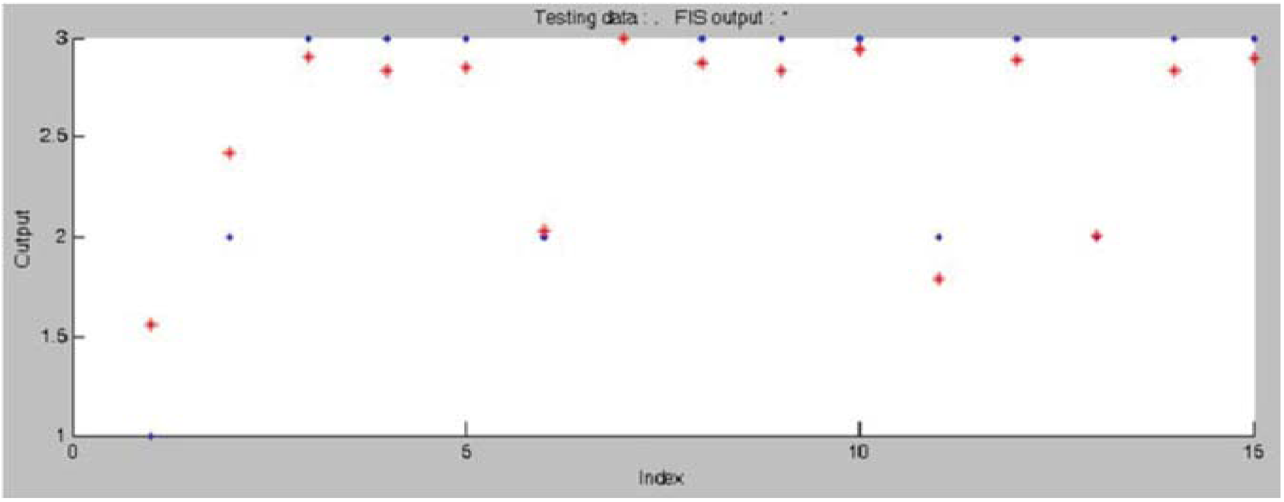

Figure 6: Test Data Vs FIS Output

And it shows 0.1571, 0.2140 as NRMSE, RMSE (equations can be found in parameters to be evaluated section) values respectively. The plot of the expected and the output of the NF system for the different inputs are shown.

3.2 DATASET

To validate our model, we had taken 17 programs of Glace EMR Medical Billing Software (on which I had worked previously as a Software Engineer at L Cube Innovative Solutions Pvt. Ltd.,) and find out the MTTF (Mean Time to Failure), MTTR (Mean Time to Repair) and MTBR (Mean Time Between Repair) and Software Reliability Approximated value based on the program execution observations. We input these 3 values as input to input layer of Neural Network and apply sigmoid fuzzy membership function at the hidden layer of neural network and try to find out the software reliability approximated value. The previous values assessed using conventional traditional software reliability growth models and our Neuro Fuzzy systems based model are compared and we found to be our model is the promising one.

Software reliability is measured in terms of mean time between failures(MTBF).MTBF consists of mean time to failure (MTTF) and mean time to repair(MTTR). MTTF is the difference of time between two consecutive failures and MTTR is the time required to fix the failure.

Let us take Software Reliability for good software is a number between 0 and 1. Reliability increases when errors or bugs from the program are removed or minimized. For example, if MTBF = 1000 hours for average software, then the software should work for 1000 hours for continuous operations. The dataset contains failure observations of 17 programs in Glace EMR Billing Software, in time series (i, Xi) and is used to predict the performance of the proposed model. Where, i = Program serial number.

3.3 Parameters used for Validation

Software Reliability: Software reliability is measured in terms of mean time between failures(MTBF).MTBF consists of mean time to failure (MTTF) and mean time to repair(MTTR). MTTF is the difference of time between two consecutive failures and MTTR is the time required to fix the failure.

Let us take Software Reliability for good software is a number between 0 and 1. Reliability increases when errors or bugs from the program are removed or minimized.

For example, if MTBF = 1000 hours for average software, then the software should work for 1000 hours for continuous operations.

MTBF = Avarage time between consecutive software system failures =MTTF+MTTR

MTTR = Average time taken to repair the system after the occurrence of failure.

Software Reliability = MTBF / (1+MTBF)

Availability = MTBF/(MTBF+MTTR) , is the likelihood that a software system will work at a given time.

3.4 Experimental results

Table 1: Production time analysis for the program dataset

S.No	Program #	Prod. time(Hrs.)	Uptime at x1(Hrs.)	Uptime at x2(Hrs.)	Downtime at x1(Hrs.)	Downtime at x2(Hrs.)	No. of breaks at x1(Hrs.)	No. of breaks at x2(Hrs.)
1	GE01	256	216	202	40	54	3	11
2	GE02	324	260	203	64	121	9	16
3	GE03	236	168	154	68	82	2	19
4	GE04	600	450	435	150	165	16	23
5	GE05	371	300	265	71	106	13	35
6	GE06	447	430	410	17	37	15	21
7	GE07	865	560	525	305	340	10	25
8	GE08	843	615	575	228	268	4	31
9	GE09	943	720	706	223	237	17	28
10	GE10	135	85	78	50	57	4	6
11	GE11	242	130	132	112	110	36	22
12	GE12	369	240	206	129	163	24	30
13	GE13	122	68	64	54	58	23	9
14	GE14	107	72	74	35	33	6	15
15	GE15	371	265	253	106	118	18	34
16	GE16	453	370	398	83	55	21	37
17	GE17	325	285	256	40	69	27	29

3.4.1. Calculations

Total Production time= Uptime+ down time

$$\textbf{MTBF} = \frac{\text{Total uptime (total time- total downtime)}}{\text{Number of Breakdowns}}$$

Where,

MTTF= Mean Time to Failure (in hours/minutes/seconds).
MTTR= Mean Time to Repair (in hours/minutes/seconds).
MTBF= Mean Time between Failures (in hours/minutes/seconds).

$$MTTR = \frac{\text{Total downtime}}{\text{Number of breakdowns}}$$

$$MTTF = \frac{(\text{Failure at obs.1} + \text{Failure at obs.2} + \ldots + \text{Failure at obs.N})}{\text{Number of software programs under test}}$$

$$\text{Availability (For Repairable software systems)} = \frac{MTBF}{(MTBF + MTTR)}$$

Table 2: calculation of MTBF & MTTR

S.No.	Program	MTTF	MTTR	MTBF
1	GE01	0	9.12	45.18
2	GE02	0	7.336	20.78
3	GE03	0	19.158	46.05
4	GE04	0	8.25	23.51
5	GE05	0	4.25	15.32
6	GE06	0	1.447	24.09
7	GE07	0	22.05	38.5
8	GE08	0	32.82	86.14
9	GE09	0	10.791	33.784
10	GE10	0	11	17.12
11	GE11	0	4.056	4.80
12	GE12	0	5.042	8.43
13	GE13	0	4.396	5.03
14	GE14	0	4.016	8.46
15	GE15	0	4.679	11.08
16	GE16	0	2.719	14.18
17	GE17	0	1.93	9.69

Table 3: calculation of Availability

S.No.	Program	MTTR	MTBF	Availability
1	GE01	9.12	45.18	0.832
2	GE02	7.336	20.78	0.739

3	GE03	19.158	46.05	0.706
4	GE04	8.25	23.51	0.739
5	GE05	4.25	15.32	0.783
6	GE06	1.447	24.09	0.943
7	GE07	22.05	38.5	0.635
8	GE08	32.82	86.14	0.724
9	GE09	10.791	33.784	0.757
10	GE10	11	17.12	0.608
11	GE11	4.056	4.80	0.543
12	GE12	5.042	8.43	0.609
13	GE13	4.396	5.03	0.533
14	GE14	4.016	8.46	0.678
15	GE15	4.679	11.08	0.703
16	GE16	2.719	14.18	0.839
17	GE17	1.93	9.69	0.833

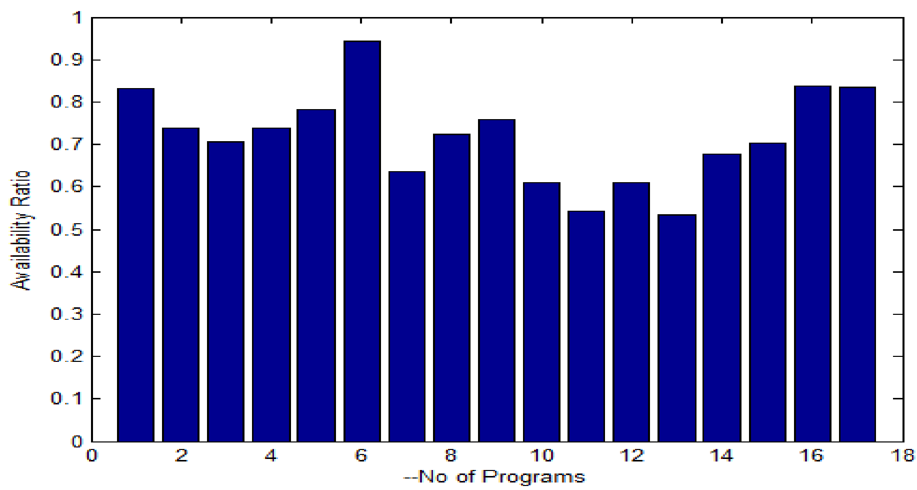

Figure 7: Analysis of availability ration w.r.t. number of programs

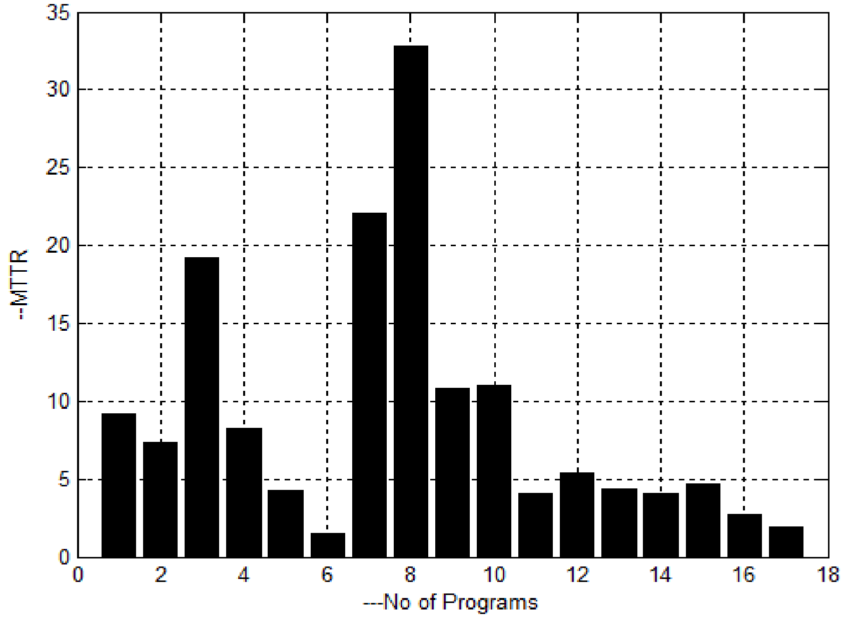

Figure 8: Analysis of MTTR ration w.r.t. number of programs

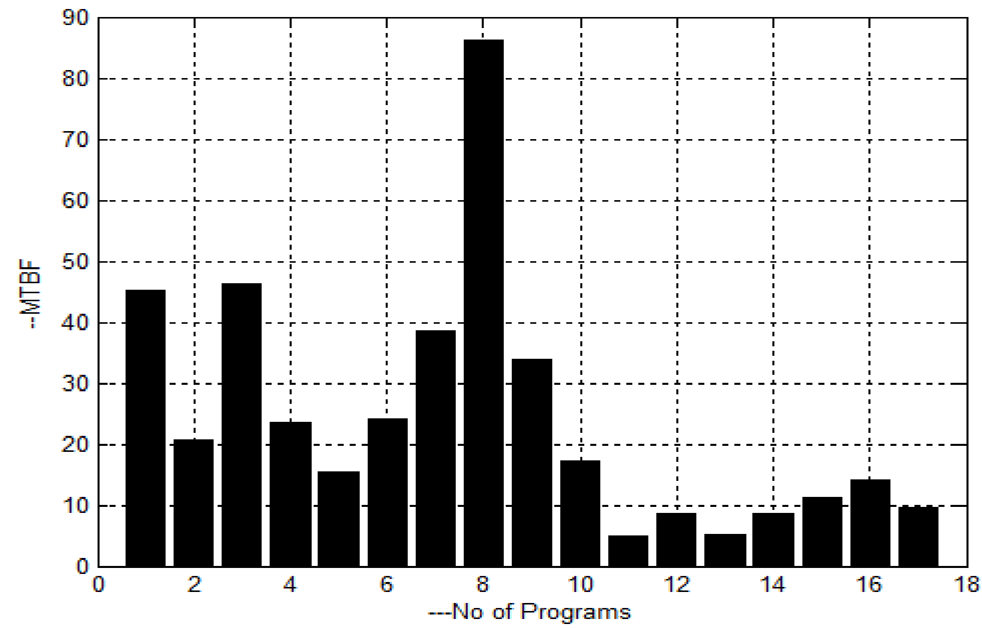

Figure 9: Analysis of MTBF ration w.r.t. number of programs

3.5 Theoretical Validation

From the above section 3.11 a theoretical valuation can be done with the formula mentioned in the context. For example at the 1st Iteration

Table 4: calculation of Reliability & its approximation at 1st iteration

x	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y + h * f(a)
1	45.18	0.97835	45.237
2	20.78	0.95409	20.835
3	46.05	0.97875	46.107
4	23.51	0.9592	23.566
5	15.32	0.93873	15.374
6	24.09	0.96014	24.146
7	38.5	0.97468	38.557
8	86.14	0.98852	86.197
9	33.784	0.97125	33.84
10	17.12	0.94481	17.175
11	4.80	0.82759	4.848
12	8.43	0.89396	8.4818
13	5.03	0.83416	5.0784
14	8.46	0.89429	8.5119
15	11.08	0.91722	11.133
16	14.18	0.93412	14.234
17	9.69	0.90645	9.7426

% Reliability after 1st iteration = (24.29/ 24.44)*100=99.38

At 2nd iteration

Table 5: calculation of Reliability & its approximation at 2nd iteration

X	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y +h* f(a)
1	45.237	0.97835	45.294
2	20.835	0.95409	20.89
3	46.107	0.97875	46.164

4	23.566	0.9592	23.622
5	15.374	0.93873	15.428
6	24.146	0.96014	24.202
7	38.557	0.97468	38.614
8	86.197	0.98852	86.254
9	33.84	0.97125	33.896
10	17.175	0.94481	17.23
11	4.848	0.82759	4.896
12	8.4818	0.89396	8.5336
13	5.0784	0.83416	5.1268
14	8.5119	0.89429	8.5638
15	11.133	0.91722	11.186
16	14.234	0.93412	14.288
17	9.7426	0.90645	9.7952

% Reliability after 2nd iteration = (24.35/ 24.44)*100=99.62

At 3rd Iteration

Table 6: calculation of Reliability & its approximation at 3rd iteration

X	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y +h* f(a)
1	45.294	0.97835	45.351
2	20.89	0.95409	20.945
3	46.164	0.97875	46.221
4	23.622	0.9592	23.678
5	15.428	0.93873	15.482
6	24.202	0.96014	24.258

7	38.614	0.97468	38.671
8	86.254	0.98852	86.311
9	33.896	0.97125	33.952
10	17.23	0.94481	17.285
11	4.896	0.82759	4.944
12	8.5336	0.89396	8.5854
13	5.1268	0.83416	5.1752
14	8.5638	0.89429	8.6157
15	11.186	0.91722	11.239
16	14.288	0.93412	14.342
17	9.7952	0.90645	9.8478

% Reliability after 3rd iteration = (24.40/ 24.44)*100=99.83

At 4th Iteration

Table 7: calculation of Reliability & its approximation at 4th iteration

X	y(Measured Value)	f(a)=MTBF/(1+MTBF)	Approximated value= y +h* f(a)
1	45.351	0.97835	45.408
2	20.945	0.95409	21
3	46.221	0.97875	46.278
4	23.678	0.9592	23.734
5	15.482	0.93873	15.536
6	24.258	0.96014	24.314
7	38.671	0.97468	38.728
8	86.311	0.98852	86.368
9	33.952	0.97125	34.008
10	17.285	0.94481	17.34
11	4.944	0.82759	4.992
12	8.5854	0.89396	8.6372
13	5.1752	0.83416	5.2236

14	8.6157	0.89429	8.6676
15	11.239	0.91722	11.292
16	14.342	0.93412	14.396
17	9.8478	0.90645	9.9004

% Reliability after 4th iteration = (24.439/ 24.44)*100=99.99

% Reliability= (Average of Approximated vales/ Average of observed Values) x 100

Overall percentage of Reliability= (24.28/ 24.44)*100=99.70

In 4th iteration, we got 99.99%, so we stop iteration process because we got good approximated % of reliability.

3.6 Practical Validation

The experiment was conducted with 17 programs of Glace EMR Medical Billing the analysis was done using FIS (fuzzy interference system) and the proposed Neuro Fuzzy model. The model structure and error tolerance graphs are depicted below.

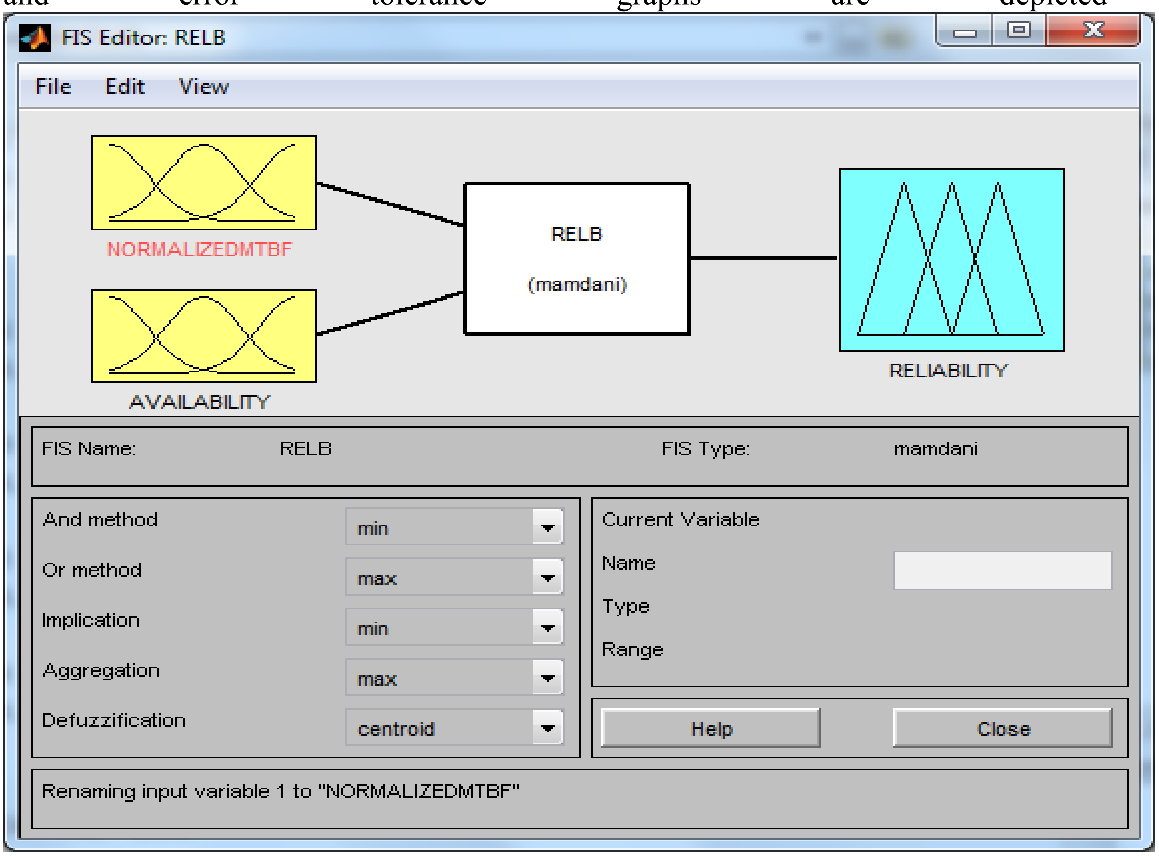

Figure 10: FIS system model

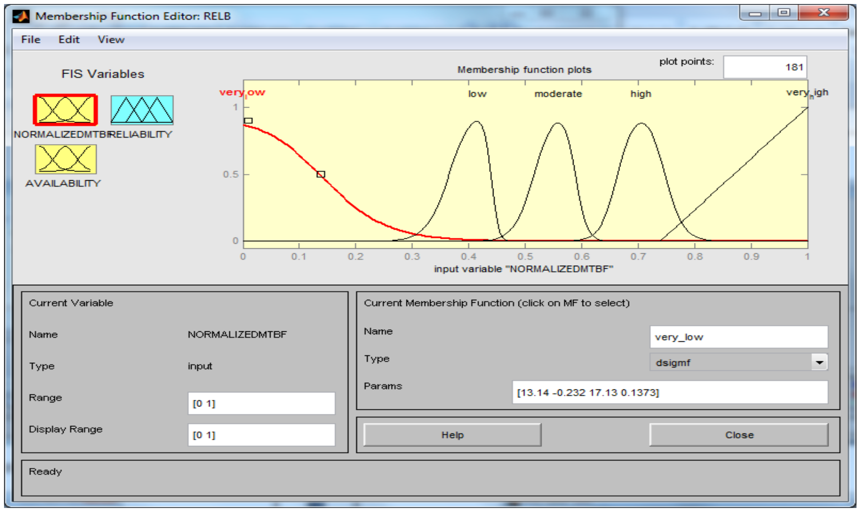

Figure 11: Membership function for MTBF and Availability

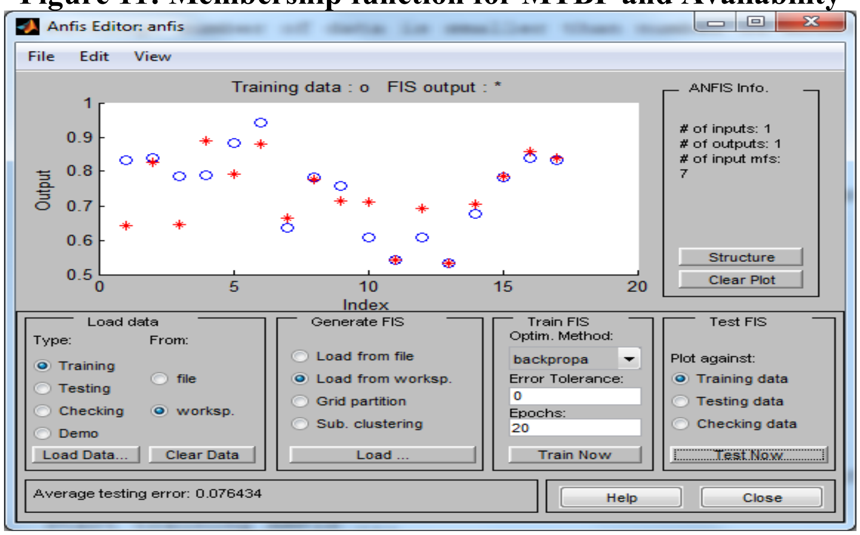

Figure 12: Neuro Fuzzy inference Model

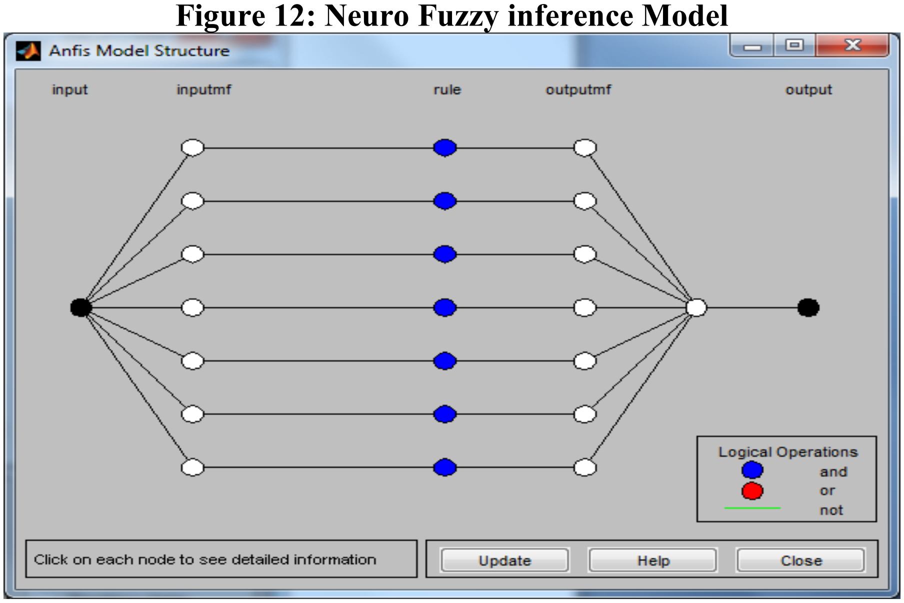

Figure 13: Neuro Fuzzy Structure

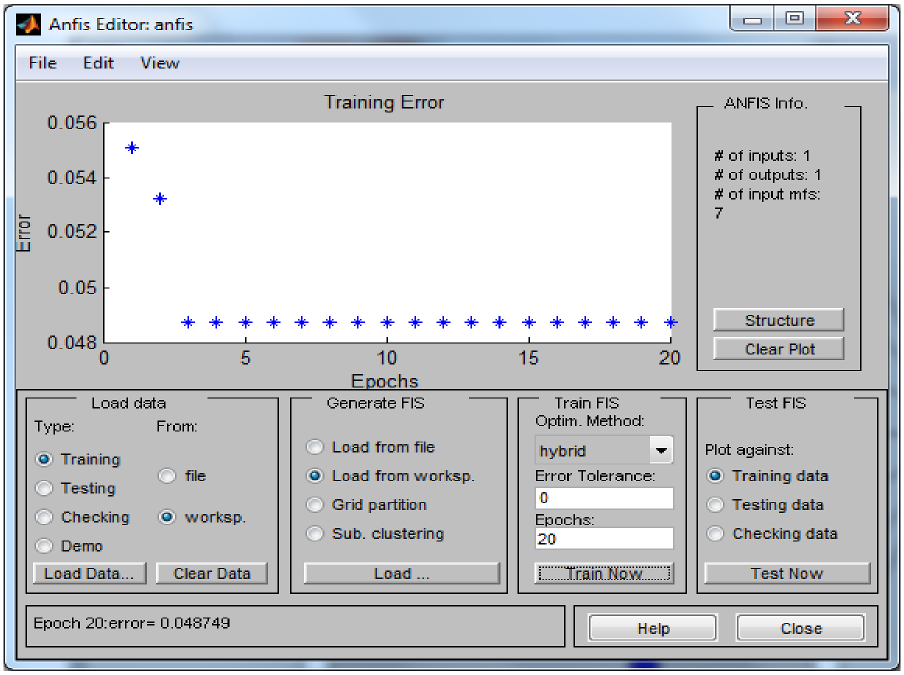

Figure 14: Error Tolerance

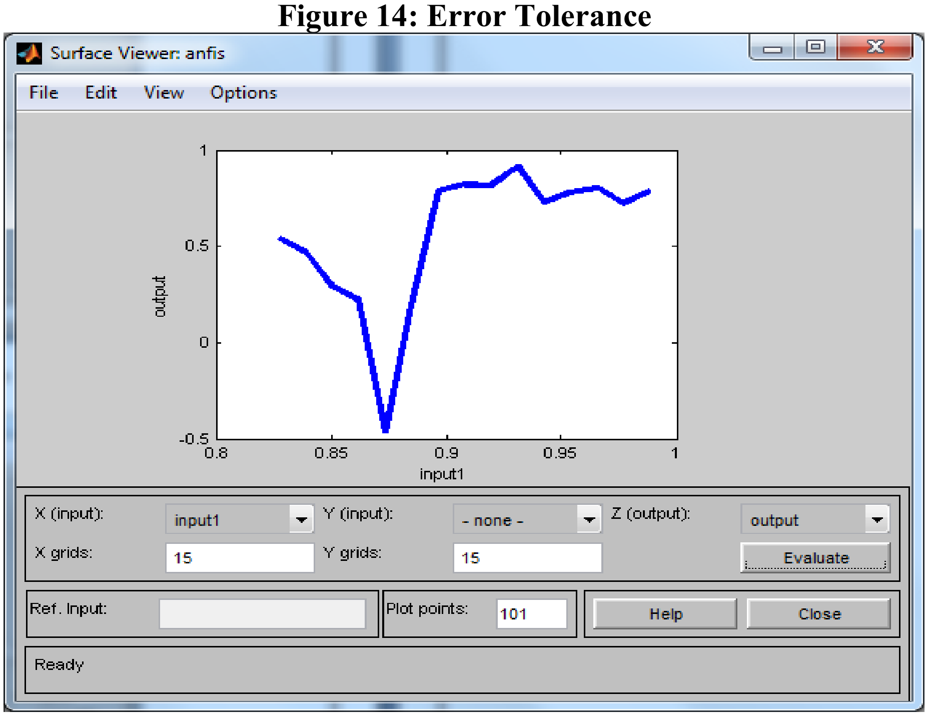

Figure 15: Performance analysis

3.7. Comparison of proposed approach with Conventional Fuzzy system

The inputs to the Neuro Fuzzy system are Normalized MTBF and availability which is show in the figure 23. The outcome of the conventional system is **84.5 %** and the proposed approach is **95.5 %** of reliability which is evaluated with MATLAB software tool, when we run the program in MATLAB environment(See Appendix-A). From the above the performance assessment for which an improvement of **11%** is achieved with the current proposal.

MSE= ((Theoretical validation- practical Validation)/total no of readings)^2

= **((99.7-95.5)./17).^2=0.061038**

Average error = ((Theoretical validation- practical Validation)/total no of readings)=0.247;

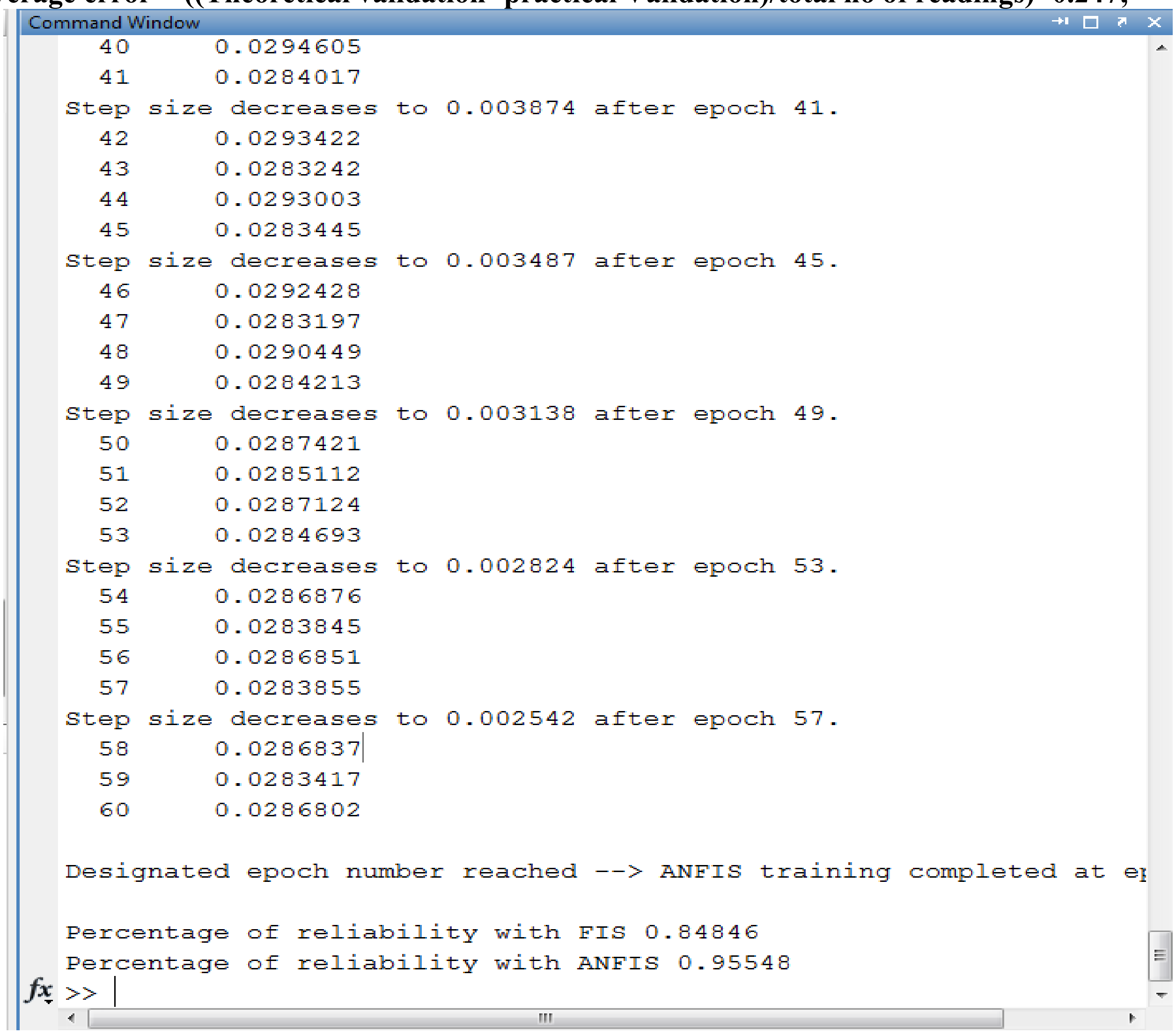

Figure 16: Practical validation of the reliability percentage obtained using MATLAB

Table 8: Performance comparison between FIS & ANFIS of SR estimation

Method	**MSE**	**AE**
FIS	0.799	0.894
ANFIS	0.061	0.247

Future Work and Suggestions

Research is a continuing activity as a future research plans the following tasks are to be completed:

- Software reliability can be predicted using hybrid intelligent system. In addition to neural network model genetic programming can be applied further.
- Novel recurrent architectures for Genetic Programming (GP) and Group Method of Data Handling (GMDH) to predict software reliability can be proposed.
- We can extend this work to the other machine learning techniques like Neuro Fuzzy systems approach, support vector machine approach, self-organizing maps approach, decision-region approach etc. for the better estimation of the software reliability at different stages of Software Development Life Cycle(SDLC) process. We can also incorporate recent evolutionary computational mechanisms for the purpose of assessing the software reliability.

CONCLUSION

From the research we found that Neuro Fuzzy model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do software reliability test. As the weights are randomly initialized, thus the model gives different results for the same datasets and thus the performance of the model varies. The usefulness of a Neuro Fuzzy model is dependent on the nature of dataset up to a greater extent.

The preliminary computational results in the MATLAB environment seem quite promising and give insight into the generalization capability of these models. The results of the fuzzy logic and neural networks models were very promising. The error difference between the actual and estimated response was small. This finding gives a good indication of prediction capabilities of the developed fuzzy model and neural networks for assessing the software reliability.

REFERENCES

[1] Hoyer, R. W. and Hoyer, B. B. Y., "What is quality?" Quality Progress, no. 7, pp. 52-62, 2001.

[2] Crosby P.B," Quality is Free the Art of Making Quality Certain NY, McGraw –Hill-1979

[3] Deming & Edwards W,"Out of Crisis", MIT Press-1986

[4] Feigenbaum A. V,"Total Quality Control", McGraw – Hill-1983

[5] IEEE Std 610.12 (1990). IEEE Standard Glossary of Software Engineering Terminology, NY.

[6] ISO/IEC Std 9126-1," Software Engineering – Product Quality ", Part 1: Quality Models, International Organization for Standards 2001.

[7] McCall J.A, Richards P.K & Walters G.F," Factors in Software Quality", Nat'1 Tech Information Services, - Vol 1-2, 1977

[8] AIAA/ANSI "Recommended Practice for Software Reliability, the American Institute of Aeronautics and Astronautics", Washington DC, Aerospace Center, R-013, 1992,-ISBN 1-56347-024-1.

[9] Michael R. Lyu," Handbook of software Reliability Engineering ", IEEE

[10] Wasserman, Gary . Reliability verification, Testing and analysis in

engineering design. Marcel Dekker incorporated, Newyork, USA, pp2-10, 2002

[11] Schneidewind, N. F., 2001, ‘Modeling the fault correction processes, Proceedings of the 12th International Symposium on Software Reliability Engineering, pp. 185-190.

[12] Myrtveit, I., Stensrud, E. and Shepperd, M., 2005, ‘Reliability and validity in comparative studies of software prediction models’, IEEE Transactions on Software Engineering, vol. 31, no. 5, pp. 380-391.

[13] Falcone, G. Hierachy-aware Software Metrics in Component Composition Hierarchies. PhD thesis, University of Mannheim.-2010

[14] Bass, L., Clements, P., and Kazman, R. “Software Architecture in Practice” , Second Edition. Addison-Wesley, Reading, MA, USA-2003

[15] Jain, R. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley.-1991

[16] Smith, C. U. and Williams, L. G. (2000). Software performance anti patterns. In Workshop on Software and Performance, pages 127-136.

[17] Immonen, A. and Niemel□a, E.,” Survey of reliability and availability prediction methods from the viewpoint of software architecture. Software and System Modeling-7(1):49{65)-2008

[18] AIAA/ANSI “Recommended Practice for Software Reliability, The American Institute of Aeronautics and Astronautics”, Washington DC, Aerospace Center, R-013, 1992,-ISBN 1-56347-024-1.

[19] Rosenberg Linda, Hammer Ted & Shaw Jack, “Software Matrices and Reliability”, ISSRE, 1998.

[20] N. E. Fenton and M. Neil. A critique of software defect prediction models. IEEE Transactions on Software Engineering, 25(5):675–689, 1999.

APPENDIX- A

MATLAB PROGRAM FOR PRACTICAL VALIDATION

```
clear
aa= VECTOR OF MTBF VALUES;
af=aa./mean(aa);
 b= VECTOR OF AVAILABILITY VALUES;
% Read the FIS structure named as RELB
F=readfis('RELB.fis');
% Evalate the input with the given fuzzy structre
 ff=evalfis([aa./max(aa)+.7,b+.7]',F)
 % this section is regarding ANFIS,train the data for it give MTBF and Availability as inputs
 trnData = [af , b];
numMFs = 7;
mfType = 'dsigmf';
epoch_n = 100;
% generate a new anfis with this training data
 in_fis = genfis1(trnData,numMFs,mfType);
```

```
 out_fis = anfis(trnData,in_fis,60);
ff'
mean(ff)
% evaluate the data with input anfis structure
oo=evalfis([b]',out_fis)'
mean(oo)
```

Mr. Bonthu Kotaiah obtained his Bachelor's degree in Computer Applications from Nagarjuna University in 2001 and M.C.A from Nagarjuna University in 2008. During the period from September, 2001 to 2011, he has been involved in various aspects of Information Technology - an engineer(L-Cube Innovative Solutions), a Corporate Trainer (SyncSoft & Datapro(Vijayawada), COSS(Hyd.)), a Computer Programmer(Acharya Nagarjuna University). Currently he wishes to conduct research in the area of Software Engineering and Data Mining and Artifiicial Neural Networks, Fuzzy Logic & Genetic Alogorithms. His research interests include software Engineering, Neural networks. Presently, he is working as a Full-Time Research Scholar in Babasaheb Bhimrao Ambedkar University (A Central University) Lucknow, UP in the Department of Information Technology.

Dr. R.A. Khan, Presently Working with Babasaheb Bhimrao Ambedkar University (A Central University) Lucknow, UP as a **Professor & Head** in the Department of Information Technology, School for Information Science & Technology since December 2006. He is having More than **Ten Years** of teaching experience at PG and UG Level. **He obtained Master of Computer Application (M.C.A.)** from PTU, Jalandhar securing **73%** marks (**2000**) and **Ph.D.** from Jamia Millia Islamia (A Central University) New Delhi (**2004**).