DRCE Maintainability Model for Component Based Systems using Soft Computing Techniques

Kiran Narang, Puneet Goswami

Abstract: ***Effective software maintainability is one of the most significant and challenging activity in the field of component based software. Several maintainability models are proposed by the researchers to reduce the maintenance cost, to improve the quality and life span of the software product. The proposed model will assist the software designers to develop maintainable softwares. This paper discusses a maintainability model, which selects four crucial factors that highly affect maintainability of component based software system. Soft computing techniques are employed to demonstrate strong correlation of these factors with maintainability. MATLAB's Fuzzy logic toolbox is used for predicting the maintainability level of component (such as Excellent, Fair, Good, Bad and worst). Data generated by fuzzy model are provided as input to artificial neural network model. Experimental results shows mean absolute error (MAE) to be .028 and Relative Error (RE) to be .045.To further improve the performance of the model; neuro-fuzzy tool was employed. With the use of self learning capability of this tool, MAE and RE are now improved to the value .0029 and .039. It means that the model was sound enough to provide satisfactory outcomes in comparison to neural network.***

Index Terms: Component Based System, Coupling, Document Quality, Extensibility, Maintainability, MATLAB Fuzzy Logic, Quality models, Reusability.

I. INTRODUCTION

Component Based Software Engineering (CBSE) is the methodology of developing software by the reuse of already available softwares or components.

Software maintenance is considered as the most important phase of software development life cycle. The alterations done, after delivery of the software comes under the process of maintenance (Swanson, 1999). Maintenance usually becomes essential due to change in client's requirement, current conditions of the market, capability enhancement and host modifications (Anda, 2007). Maintainability of CBSE can be described as the simplicity by which the software can be understood, modified, enhanced and corrected according to the user's current requirement. (Muthanna, Kontogiannis, Ponnambalaml, & Stacey, 2000).

A study done on "software maintenance estimation" discovered that the cost of maintenance can be 65% of the total cost of the software process cycle (Rizvi & Khan, 2010). Lifespan of the development phase of software is approximately only two to five years whereas maintenance phase may extent up to twenty years. However, this phase is not perfectly managed. Software maintenance can be divided into four sections i.e. adaptive maintenance, perfective maintenance, corrective maintenance and preventive maintenance (Kumar & Dhanda, 2015).

Adaptive maintenance is concerned with amendments required by the software, so as to adapt the new environment such as new operating system or hardware. Corrective maintenance is concerned with repair of errors, bugs and faults that are discovered by the users when the software is in use. Perfective maintenance includes the implementation of new functionalities, features, enhancing performance and reliability of the software. Preventive maintenance is concerned with amendments that are needed to be done to deal with forthcoming problems so that the product may remain operational for an extended time. These problems may not appear significant today but in future they may cause troubles (Lenarduzzi, Sillitti, & Taibi, 2017).

There are several factors that influence maintainability of component based software such as modularity, understandability, modifiability, granularity, analyzability, testing, usability, coupling, cohesion, complexity etc as discussed in Table 1. Here, we have proposed a maintainability model that relies on four imperative factors, which are: Document Quality, Reusability, Coupling and Extensibility (DRCE). This model can be used to estimate the maintainability of the components before they became the part of component based system.

This paper consists of six sections. Second section discusses the literature review of related work. Third section discusses proposed methodology using fuzzy approach and results from the model. Fourth section discusses neural network approach of the model and the experimental results of the model. Fifth section illustrates the neuro fuzzy approach and its outcomes. Sixth section concludes the papers. At the end, list of references shows the referred papers.

II. LITERARTURE REVIEW

This section describes the work done by various researchers in the field of maintenance that motivate us to build up a maintainability model for CBSE. The main complexity in maintaining the Component Based System (CBS) is the non -availability of the source code, as it remains available in traditional software. Literature survey showed that maintainability of Component Based Systems cannot be measured directly.

Revised Manuscript Received on July 10, 2019.

Kiran Narang , Research scholar, Department of Computer science and Engineering, SRM University, Haryana, India

Puneet Goswami ,Department of Computer science and Engineering, SRM University, Haryana, India .

Researchers and academicians are trying to find out the factors that largely affect the maintenance thus reducing the maintenance cost.

A conceptual framework was proposed by Peercy (1981), for software maintainability evaluation and it shows that maintainability depends on modularity, descriptiveness, consistency, simplicity, expandability, and instrumentation. Sneed & Mary (1985) proposed a model that considered design attributes for calculating maintainability.

Don Coleman (1994) in his research demonstrated how automated software maintainability analysis tool can be useful for process selection decision. This model determined the error prone and unorganized components. The researcher calculated maintainability through Hierarchical Multidimensional Assessment Model. Aggarwal (2005) proposed a fuzzy model that measure maintainability with the use of four factors i.e. average number of Live Variables (LV), average Life Span (LS) of variables, the Average Cyclomatic Complexity (ACC) and the Comments Ratio (CR).

Jyothi and Aggarwal (2012) proposed an Aspect J model to determine perfective maintenance of Component based software system. They provided a process model and calculated inter and intra level component interaction.

Punia and Kaur (2014) proposed a model in which maintainability is defined as software characteristics' measure such as source code, readability, document quality and cohesion.

Almugrin, Albattah & Melton (2016) developed maintainability metrics using Indirect Responsibility (IR) and Martin's principles. Lenarduzzi, Sillitti, & Taibi (2017) in their research paper analyzed forty years of maintenance models.

Alsolai, Roper & Nassar (2018) presented a comparison of two machine learning models to forecast software maintainability that is individual models (regression tree, multilayer perception, k-nearest neighbors), and an ensemble model (bagging) method.

Table 1 demonstrates a few factors given by various researchers that affect the maintainability of component based system using various soft computing techniques (Kumar, 2012; Peercy, 1981; Aggarwal, 2005; Punia and Kaur, 2014; Saini et al., 2011). With the advancement in the field of maintainability techniques, more factors will continue to be the part of this table.

Table 1 A few Factor influencing maintainability

Factors that affect Maintainability	Brief Description
Modularity	Competence of a software package or a component being break down into proficient units.
Understandability	It is the ability of a software component to be easily and effortlessly understood.
Modifiability	It is the aptitude to allow amendments (appending and detaching) in functionalities of a software system.
Granularity	Granularity is the capability to split bigger tasks into the minor tasks.
Analyzability	Analyzability engrosses the challenge required to identify the insufficiency, sources of breakdown and identifying the components that may need modifications.
Testability	It is the capability of a software package to be efficiently checked for faults, errors and bugs.
Complexity	The complexity of software refers to the metric which informs how complex the code is and it is inversely proportional to its maintainability.
Cohesiveness	It is the amount or strength of belongingness within the modules in the software component.

From literature surveys we observed that diverse researcher give importance to diverse factors for computation of maintainability. No paper seems to provide an exact and successful description of factors that affect the maintainability of component based software. Maintainability is considered as a quality factor however researcher community has not concluded the major influencing factors for it. In our research work, rather than depending on one paper we have collected factors from a lot of papers, elected those factors that are supported by many researchers, applied soft computing techniques to get precise and significant solution in a time efficient manner. Moreover, this model can be practically used as opposed to other models that are difficult to be utilized in real life scenario, as the selected features for DRCE model can be fetched out easily by using automated tools.

DRCE model is capable of determining all types of maintainability, as compared to various models that calculate only a particular kind of maintainability. We have extended the model implemented by Punia and Kaur (2014). This model has taken into consideration Documentation Quality, Modifiability, integrability, testability and coupling. We have reduced complexity of this model by selecting only four factors higher impact and recommended by various researchers. The experimental results shows mean absolute error (MAE) to be .028 and Relative Error (RE) to be .045 in case of neural network. Further improvement in this model is done by employing neuro–fuzzy technique. In this case Absolute error (MAE) is .0029 and Relative Error (RE) is .039, which is better than the neural network and the previous model.

Predicting maintainability through the proposed DRCE model will facilitate the developer of component based software, in following ways:

- It will assist the developer in analyzing whether to develop a new software or maintain the already existing one.
- Developer and maintenance engineer may use this model to ensure maintainability of their own product.
- Provide cost and time benefit to the software user as well as developer.
- One can discover the maintainable components to form a Component based system and reject the components with less maintainability.
- After determining the maintainability, amount of effort and time to maintain a system can be reduced to great extent.

III. PROPOSED DRCE METHODOLOGY USING FUZZY APPROACH

Figure 1 conceptualizes the functionality of proposed maintainability model. A brief discussion on all the major parts of the proposed model is as follows:

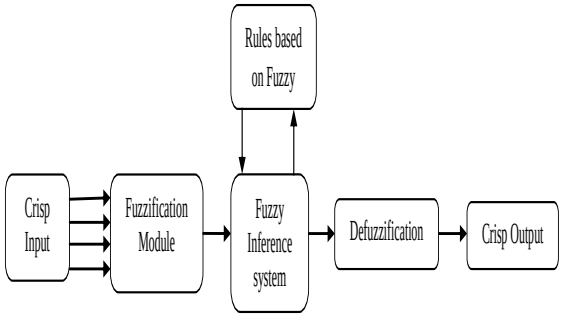


Figure 1. Proposed Maintainability Model

A. Crisp Input

The proposed DRCE model analyzes that the following four factors are highly correlated with the maintainability of a component based system in comparison to many factors as conferred in Table1, so these factors are exerted as crisp input to the fuzzy model (Chen, Alfayez, Srisopha, Boehm, & Shi, 2017).

1. **Documentations quality :**
   An excellent documentation of a component increases its probability of being selected as an associate module in component based system (D. Jain, A. Jain, & Pandey, 2018).
2. **Reusability:**
   Reusability of a component reflects its probability of being selected as a module component in various softwares. (Sharma & Baliyan, 2011).
3. **Coupling :**
   Coupling is the extent of interconnection between the software components. It illustrates the power of the interaction amongst components or modules. Coupling is inversely proportional to maintenance (Siddhi & Rajpoot, 2012).
4. **Extensibility:**

It is the software system's ability to allow considerable extension of new functionalities with no major amendment in its basic architecture for future growth (Koscianski, Candido, & Costa, 1999).

Using fuzzy, one can make use of English words as input and output instead of numeric data. These words are also called crisp input or linguistic variable (Elshoff, 1978). Figure 2 demonstrates the input and output of the proposed maintainability model.

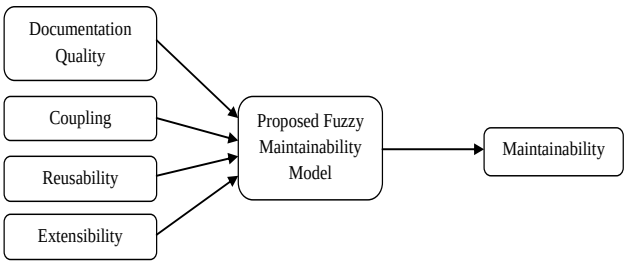


Figure 2. Inputs and Output of DRCE Maintainability Model

B. Fuzzification Module

Fuzzification is described as the process of mapping crisp input into fuzzy value. It analyzes the extent up to which the input is related to the fuzzy set through the membership function (Punia & Kaur, 2014). It can be done by using various kinds of fuzzifiers (membership functions). MATLAB's fuzzy toolbox supports a variety of membership functions for its inputs and outputs such as Gaussian, Trapezoidal, Singleton, S Function and Triangular fuzzy number (TFN). However, the most conventional is TFN due to its simplicity and therefore the same is utilized in proposed model for input and output ("Math Works", n.d.). Figure 3 demonstrates TFN with lower limit "a", an upper limit "b" and a centre value "m" (where a < m < b).

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{m-a}, & a < x \le m \\ \frac{b-x}{b-m}, & m < x < b \\ 0, & x \ge b \end{cases}$$

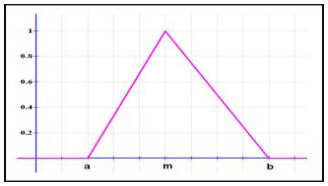


Figure 3. Triangular membership function (TFN) in fuzzy toolbox of MATLAB.

For the inputs of this model, three membership functions are taken into consideration i.e. Minimum, Average and Maximum (TFN) and for the output five membership functions are considered such as excellent, fair, good, bad and worst. For example greater value of document quality will be denoted by maximum in fuzzy and lower value of document quality will be indicated by minimum in fuzzy (Mittal & Bhatia, 2007). Figure 4,5,6,7 envision the Membership function for the input variables "Document Quality", "Reusability", "Coupling" and "Extensibility" respectively. Figure 8 envision the Membership function for the output Variable "Maintainability".

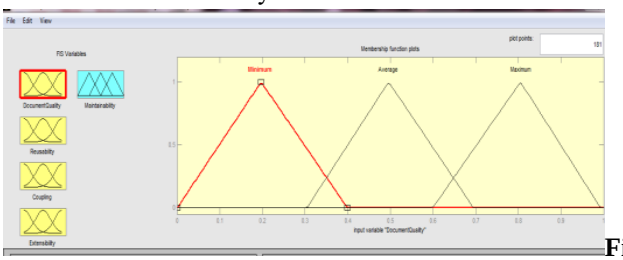


Figure 4. Envision the Membership function for the first input variable (Document Quality).

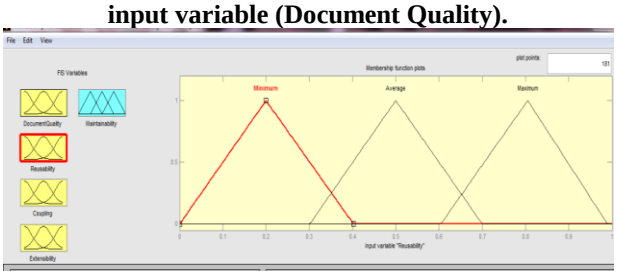


Figure 5. Envision the Membership function for the second input variable (Reusability).

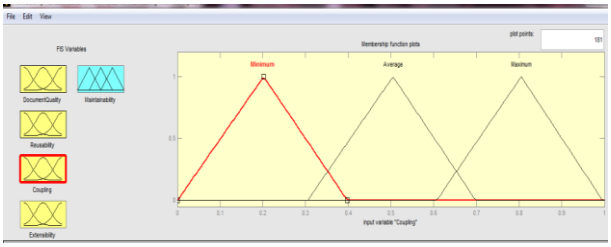


Figure 6. Envision the Membership function for the third input variable (Coupling).

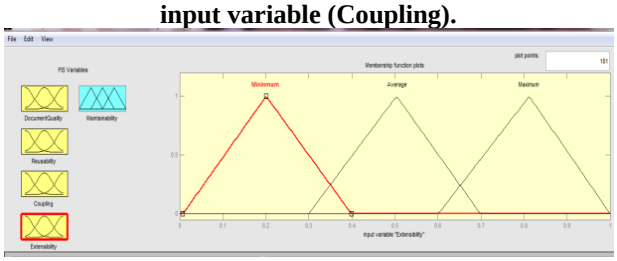


Figure 7. Envision the Membership function for the fourth input variable (Extensibility).

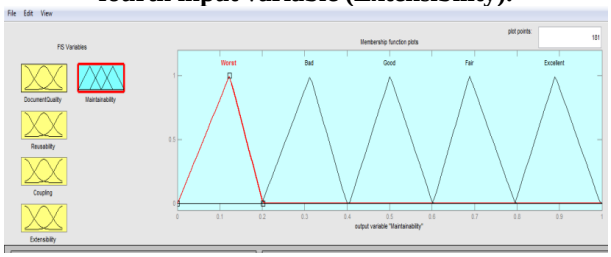


Figure 8. Envision the Membership function for the output variable (Maintainability).

C. Fuzzy Inference system and Rules

Major functionality of fuzzy Inference system is to perform the decisions. It utilizes the rules and converts the fuzzified inputs into fuzzified output (Malviya & Maurya, 2012). Fuzzy Logic toolbox of MATLAB provides two categories of fuzzy inference systems i.e. Mamdani, Sugeno. The major dissimilarity among both the systems is the way, how they defuzzify and the consequent of fuzzy rules (Olatunji et al., 2010). Mamdani fuzzy inference method is frequently used system and this model also utilizes the same. Figure 9 conceptualize the Fuzzy Inference system (FIS) of the suggested model.

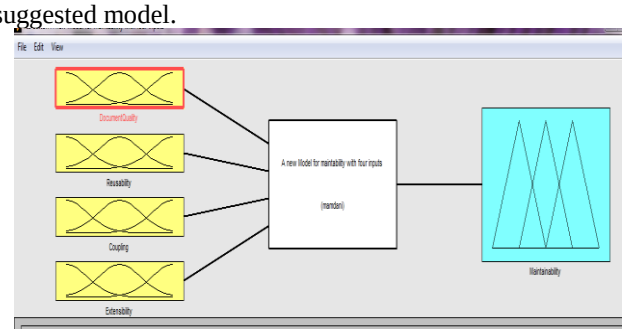


Figure 9. Fuzzy Inference system (FIS) of proposed model

As a first step fuzzification module, will fuzzify all the four inputs, next step is to construct the rule base with all possible combination of inputs (34=81). The functioning of Fuzzy Inference System (FIS) leans on the specified rules. Total number of rules formed can be calculated by the formula given in equation 1:

Total no. of Rules= (Values of the membership function) No. of Inputs (1)

In the proposed DRCE maintainability model, numbers of inputs are four (Document Quality, Reusability, Coupling, Extensibility) and value of membership functions are three (minimum, maximum, average) so the total number of rules according to the formula will be 34=81.The three inputs of model i.e. Document Quality, Reusability, Extensibility are directly proportional to the maintainability, whereas Coupling is inversely proportional to the maintainability. It means if the coupling is high, maintainability is low. Each rule corresponds to any value of the four inputs and its related output. Certain rules for the proposed model are described as follows:

- Input Values: Document quality – Maximum AND Reusability – Maximum AND Extensibility – Maximum AND Coupling – Minimum. Output Value: Maintainability - Excellent.
- Input Values: Document quality – Minimum AND Reusability – Minimum AND Extensibility – Maximum AND Coupling – Maximum. Output Value: Maintainability - Bad.
- Input Values: Document quality – Maximum AND Reusability – Average AND Extensibility -Average AND Coupling – Minimum. Output Value: Maintainability is good.

In a similar fashion, all the 81 rules are created and inserted into the rule editor .This set of 81 rules forms a complete rule base of the proposed model. Based on the rigorous set of facts provided by professional, the rule is fired up and output value of maintainability is observed. Figure 10 visualize rule editor for the DRCE proposed model.

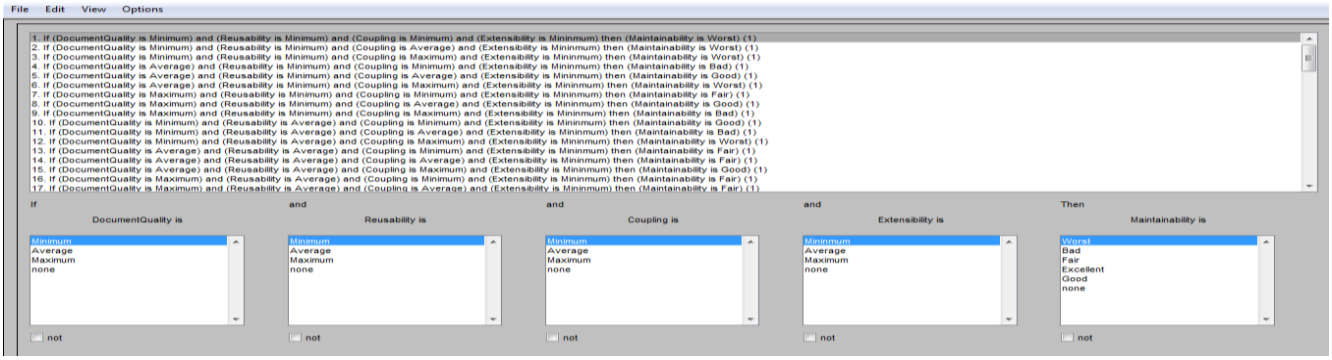


Figure 10. Rule Editor for the proposed DRCE maintainability model

D. Defuzzification and Output

Aggregation is the process of concluding final output, based on the analyses of all the applicable rules in the Fuzzy Inference System (FIS). This process takes place in the defuzzification module itself. It is the second last stage before defuzzification. Outcome of the aggregation step is one fuzzy set for each output whereas output of defuzzification is the single output value from the set.

Defuzzifier converts this single fuzzy value to the crisp value. There are five built in defuzzification methods to perform this task, such as centroid, bisector, middle of maximum, largest of maximum and smallest of maximum. The proposed model uses centroid calculation, which returns the centre of area under the curve.

E. Result Compilation:

To visualize the results, click on the view and then select "rule". Rule viewer will emerge; this demonstrates defuzzification of the input. In the proposed method, all input and output variables are defined as continuous variables, it means they can attain any value in between 0 and 1. A value close to 1 shows higher value of that attribute and a value close to 0 indicates lower value of the attribute.

Table 2 illustrates the outcome provided by the proposed model for a particular input: Document Quality 0.5, Reusability 0.75, coupling 0.5, Extensibility 0.7. The maintainability output appears out to be 0.706. Rule viewer for this input is given by the figure 11. Figure 12 shows the fine point view of the rule viewer for the same input. The output value provided by the proposed model can be verified using center of gravity formula as centroid defuzzification method is used here. Equation 2 below shows that it is calculated to be .743 which is approximately same as the value provided by the proposed DRCE maintainability model. Comparing the proposed model's output with the base model's output shows an improvement in the range of 6-10%.

Surface viewer shown in Figure 13 depicts the three dimensional view of input Document Quality on X-axis, Reusability on Y-axis and output Maintainability on Z-axis.

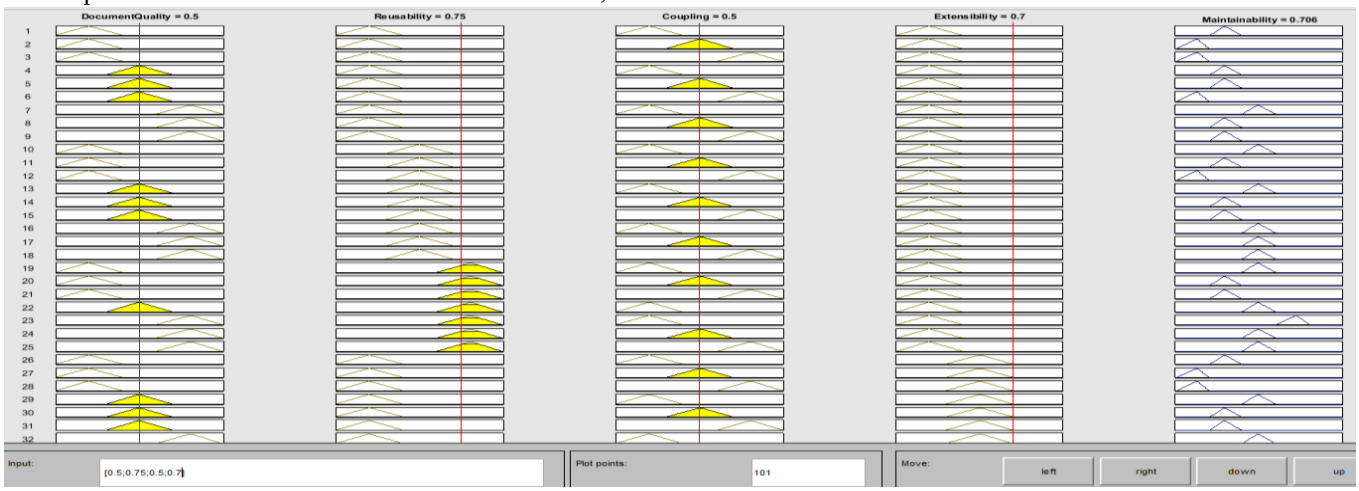
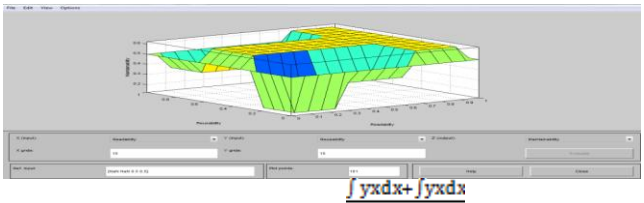


Figure 11. Rule viewer showing Maintainability output 0.706 for input (0.5, 0.75, 0.5, and 0.7)



Centre of gravity= $\frac{\int yxdx+\int yxdx}{\int ydx+\int ydx}$ = .743 (2)

Figure 13 Three dimensional view of input Document Quality on X-axis, Reusability on Y-axis and output Maintainability on Z-axis.

Table 2 *Result for a particular input set provided by the proposed DRCE maintainability model*

Input Parameters				Output Parameter
Document Quality	Reusability	Coupling	Extensibility	Maintainability
0.5	0.75	0.5	0.7	0.706

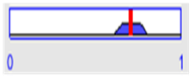


Figure 12 Fine point view of rule viewer, showing Output for the Input [.5 .75 .5 .7].

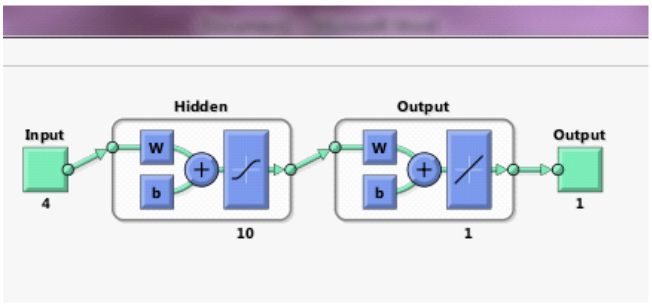
IV. USING NEURAL NETWORK (ANN) FOR THE PROPOSED METHODOLOGY

An artificial neural network is a computational model with layered configuration similar to the complex arrangement of neurons in the mind. A neural network can be trained from the data, so as to recognize patterns, to apply classifiers on the data, and to predict upcoming events. The proposed model introduces the concept of utilization of artificial neural network (MATLAB) in determining the maintainability of CBS. Network fitting app of neural network is utilized, which is a two layer feed forward neural network that maps numeric input data set into numeric output data set. Using this type of model, one can easily select the data, create a network and train it with the available data. Network can be easily evaluated using mean square error and regression.

As we have already prepared data set for fuzzy, this data set will be given as input to the artificial neural network. This model will have four inputs such as Document Quality, Reusability, Coupling and Extensibility and one output (maintainability). Training function 'Trainlm' is utilized for training the inputs which is back propagation algorithm. Linear transfer function 'Tansig' is used for the Artificial Neural Network model. 10 neurons are utilized here and model is trained with 56 data training sample, 12 testing data samples and 12 validation data samples. Artificial neural network randomly generates these data sets so as to provide good results. Figure 14 shows neural network model having four inputs and one output. It also depicts 10 neurons at the hidden layer of the model. Figure 15 shows a graph plotted by the trained model for training data samples, testing data samples and validation data samples.

A. Results of the experiment

Figure 16 depicts the experimental results of the proposed model, which shows plot for trained data set, validation data sets and test data sets. In neural network data sets are identified as circles and circles near the plotted line are said to well train.



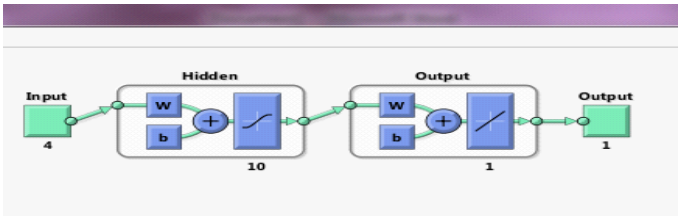


Figure 14 Neural network model with four inputs, one output and 10 layers

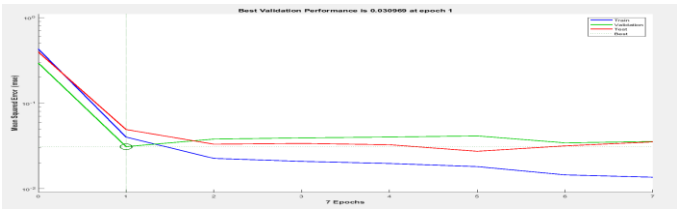


Figure 15 Graph of training model for training data samples, testing data samples and validation data samples

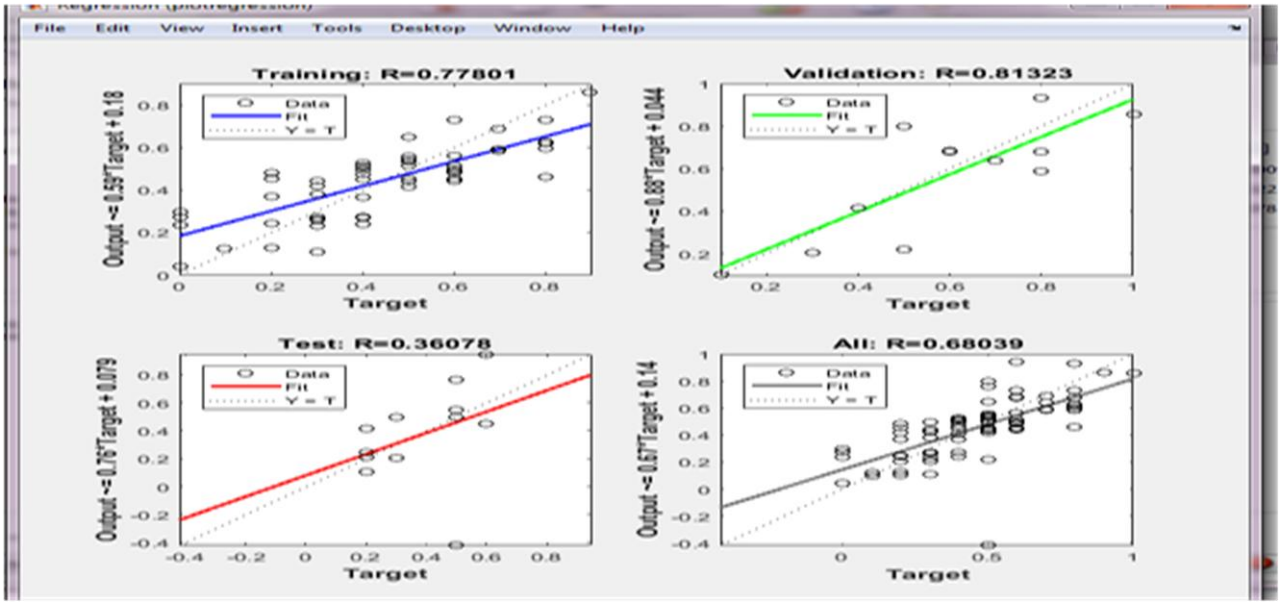


Figure 16 Results of the proposed model for trained data set, validation and test data sets.

Figure 16 depicts the experimental results of the proposed model, which shows plot for trained data set, validation data sets and test data sets. In neural network data sets are identified as circles and circles near the plotted line are said to well train. In last graph are the circles remain close to the line. It means the model is well trained and experimental results shows that this model can simply be used to calculate the maintainability of component based system.

Further to verify the outcome from the model, we have employed the Mean Absolute Error (MAE). The formula for the same is given by equation 3.

$$MAE=1/n \sum_{i=1}^{n} |xi - x| \quad (3)$$

Here 'xi' is the model output, 'x' is the actual maintainability and 'n' is the amount of observations taken into consideration. It is also necessary to authenticate and check the precision of validation data of the experiment, which was done using Relative Error (RE). Equation 4 illustrates the formula for RE. Table 3 shows the results of neural network model to be .028 and .045.

$$REaccuracy=\frac{\text{Absolute error}}{\text{True value}}*100\% \quad (4)$$

Table 3: Results of neural network model

Validation Metrics	Results
MAE	.028
$RE_{accuracy}$	.045

As the MAE and RE values are within acceptable range, it means network is well trained for predicting maintainability. The proposed model can be used suggested for the calculation of maintainability of component based system.

V. PROPOSED NEURO –FUZZY MODEL

Neuro-fuzzy is a combination of artificial neural networks and fuzzy. Neuro-fuzzy hybridization came out in a crossbreed intellectual model. It adjoins both the human like analysis of fuzzy logic systems and the learning configuration of neural networks.

Fuzzy logic is important for measuring the maintainability of software components as the conventional methods are complicated to implement. Unluckily, as the complexity of the problem increases, various factors to determine the maintainability, had to lean on supplementary models such as neuro –fuzzy model. The proposed model uses the input data set same as developed for fuzzy system. Figure 18 shows the mapping of training data in the neuro –fuzzy model.

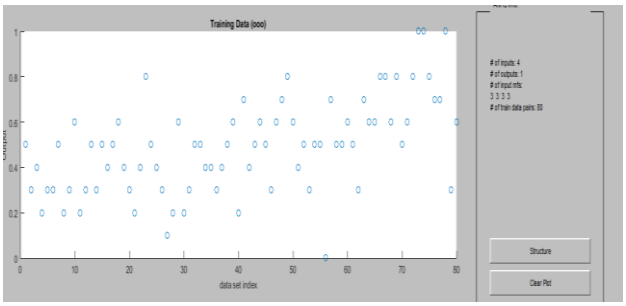


Figure 18 Training data in the neuro -fuzzy model.

Neuro –fuzzy supports an error tolerant function, which means training of the model will be done till the model achieve the value given in this function. We have selected the epoch's value to be 100 and error tolerance value is .025. Training of the model will automatically stops at epoch 100 or earlier when tolerance value reaches .025. Figure 19 shows that model has the capability to improve and reduce error on each epoch. Figure 20 shows Anfis after loading of training data, testing data and validation data. Figure 21 visualize the auto generated anfis rules.

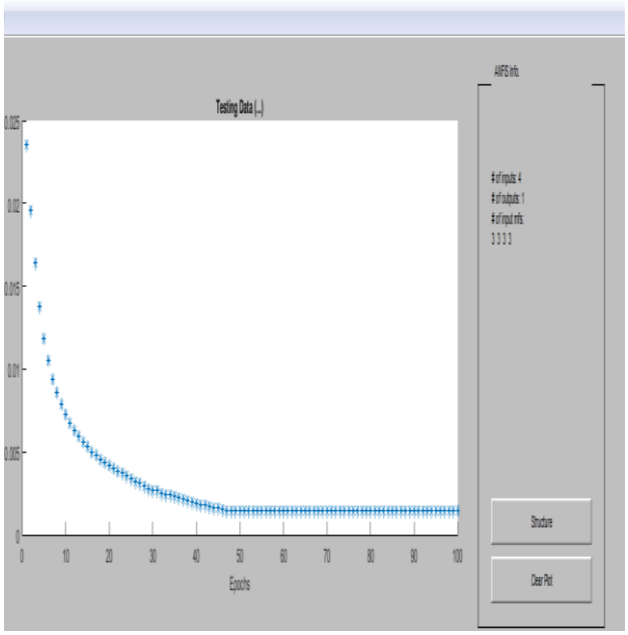


Figure 19 Reducing errors at each epoch.

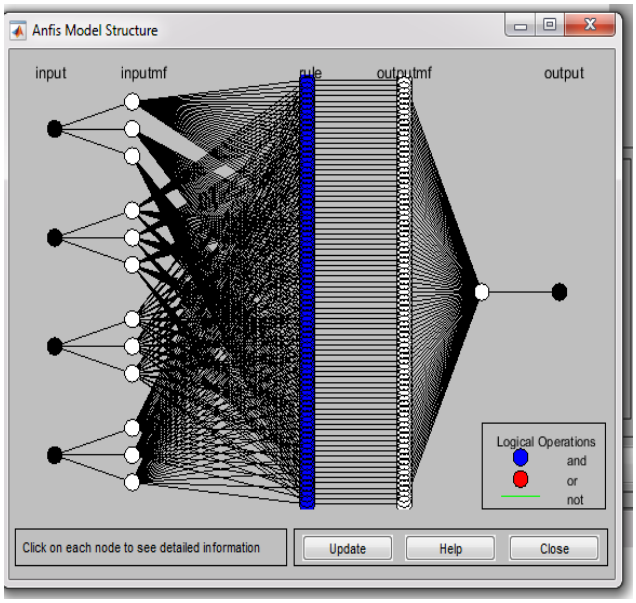


Figure 20 Anfis after loading data.

A. Experiment Results

Table 4 shows some of the outcomes from fuzzy model and neuro fuzzy model. Figure 22 shows a graph which has been plotted for the comparison of these models.

Table 4: Results from fuzzy model and neuro fuzzy model

Document Quality	Reusability	Coupling	Extensibility	Fuzzy output	Neuro fuzzy output
.5	.5	.5	.5	.494	.544
.3	.78	.25	.8	.706	.799
.4	.6	.5	.7	0.497	0.675
.4	.8	.5	.7	.706	.85
.4	.8	.7	.7	.496	.562
.4	.8	.3	.7	.708	.85
.3	.8	.3	.7	.706	.632
.3	.3	.8	.7	.299	.261
.3	.3	.3	.7	.496	.637
.3	.3	.6	.7	.299	.266
.3	.3	.4	.4	.299	.269
.4	.4	.4	.4	.136	.292
.7	.4	.3	.9	.706	.496
.7	.8	.3	.6	0.706	0.377

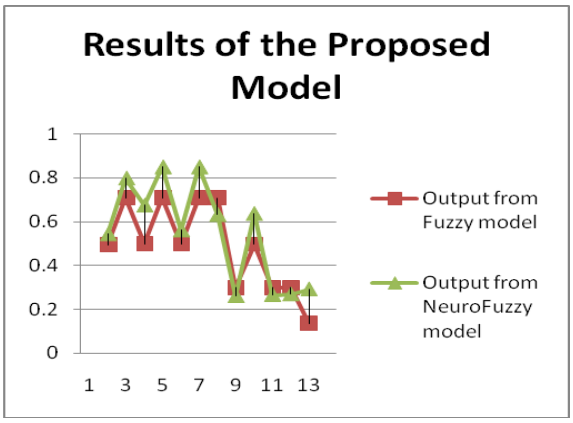


Figure 22 Comparison of the outcomes of the models.

REFERENCES

1. Abran, A., Al-Qutaish, R. E., & Cuadrado-Gallego, J. J. (2006). Analysis of the ISO 9126 on Software Product Quality Evaluation from the Metrology and ISO 15939 Perspectives. World Scientific & Engineering Academy and Society (WSEAS) Transactions on Computers, 5(11), 2778-2786.
2. Aggarwal, K. K., Singh, Y., Chandra, P., & Puri, M. (2005) Measurement of Software Maintainability Using a Fuzzy Model. Journal of Computer Sciences 1 (4), 538-542.
3. Almugrin, A.,Albattah W. , Melton, A. (2016) Using Indirect Coupling Metrics tPredict Package Maintainability and Testability The Journal of Systems & Software , doi: 10.1016/j.jss.2016.02.024.
4. Alsolai, H., Roper, M., Nassar, D. (2018) Software Maintainability in Object-Oriented Systems Using Ensemble Techniques IEEE International Conference on Software Maintenance and Evolution Predicting, DOI 10.1109/ICSME.2018.00088, 716-721.
5. Anda, B. (2007). Assessing Software System Maintainability using Structural Measures and Expert Assessments. IEEE International Conference on Software Maintenance, 204–213
6. Bernz, G. (1984). Assessing Software Maintainability. Communication of the ACM, 27, 14-23.
7. Boehm, B. & In, H. (1996). Identifying Quality-Requirement Conflicts. IEEE Software, 13(2), 25-35.
8. Chen, C., Alfayez, R., Srisopha, S., Boehm, B., & Shi, L. (2017).Why is it Important to Measure Maintainability, and what are the Best Ways to Do it? IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 377-378.
9. Coleman, D., & Ash, D. (1994). Using Metrics to Evaluate Software System Maintainability. IEEE Transactions on Computing Practices, 44-49
10. Dromey, R. G. (1995). A model for software product quality. IEEE Transactions on Software Engineering, 21(2), 146-162.
11. Elshoff, J. L. (1978). An investigation into the effects of the counting method used on software science measurements. ACM SIGPLAN Notices, 13(2), 30-45.
12. Grady, B. R. (1992). Practical Software Metrics for Project Management and Process Improvement. NJ: Prentice Hall.
13. Jain, D., Jain, A., & Pandey, A.

K. (2018). Quantification of Dynamic Metrics for Software Maintainability Prediction. International Journal of Recent Research Aspects, 5(1), 164-168.
14. Jyothi, R. & Agrawal, V.K. (2012). Study of perfective Maintainability for Component based software systems using Aspect-Oriented-Programming Techniques. International conference on intelligent Computational System, 62-66.
15. Koscianski, A., Candido, B. & Costa, J. (1999). Combining Analytical Hierarchical Analysis with ISO/IEC 9126 for a Complete Quality Evaluation Framework. IEEE International Software Engineering Standards Symposium and Forum (ISESS'99), 218-226.
16. Kumar, B. (2012). A Survey of Key Factors Affecting Software Maintainability. International Conference on Computing Sciences, 261-266.
17. Kumar, R. & Dhanda, N. (2015). Maintainability Measurement Model for Object- Oriented Design. International Journal of Advanced Research in Computer and Communication Engineering, 4(5), 68-71.
18. Lenarduzzi, V., Sillitti, A. & Taibi, D. (2017). Analyzing Forty Years of Software Maintenance Models. IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 146-148.
19. Malviya, A.K., & Maurya, L.S. (2012). Some observation on maintainability metrics and models for web based software system. Journal of Global Research in Computer Science, 3(5), 22-29.
20. Mathworks: Fuzzy Inference Process. Retrieved July, 2018, from https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html
21. McCall, J. & Walters, G. (1997). Factors in Software Quality. The National Technical Information Service (NTIS), Springfield, VA, USA, 1-168.
22. Mittal, H., & Bhatia, P. (2007). Optimization Criterion for Effort Estimation using Fuzzy Technique. CLEI Electronic Journal, 10(1), 1-11.
23. Muthanna, S., Kontogiannis, K., Ponnambalaml, K., & Stacey, B. (2000). A maintainability model for industrial software systems using design level metrics. Proceedings of Seventh Working Conference on Reverse Engineering, 248–256.
24. Olatunji, S. O., Rasheed, Z., Sattar, K. A., Mana, A. M., Alshayeb, M., & Sebakhy, E. A. (2010). Extreme Learning Machine as Maintainability Prediction Model for Object- Oriented Software Systems. Journal of Computing, 2(8), 49-56.
25. Peercy, D. E. (1981). A Software Maintainability Evaluation Methodology. IEEE Transactions on Software Engineering, 7(4), 343-351
26. Punia, M., & Kaur, A. (2014). Software Maintainability Prediction Using Soft Computing Techniques. International Journal of Innovative Science, Engineering & Technology, 1(9), 431-442.
27. Rizvi, S. W. A., & Khan, R. A. (2010). Maintainability Estimation Model for Object- Oriented Software in Design Phase (MEMOOD). Journal of Computing, 2(4), ISSN 2151-9617, Retrieved from https://sites.google.com/site/journalofcomputing/ , 26-32.
28. Saini, R., Dubey, S. K., & Rana, A. (2011). Aanalytical study of maintainability models for quality evaluation. Indian Journal of Computer Science and Engineering (IJCSE), ISSN: 0976-5166. 2(3), 449-454.
29. Sharma, V., & Baliyan, P. (2011). Maintainability Analysis of Component Based Systems. International Journal of Software Engineering and Its Applications. 5(3), 107-118.
30. Siddhi, P., & Rajpoot, V. K. (2012). A cost Estimation of Maintenance phase for component based Software. IOSR Journal of Computer science, 1(3), 1-8.
31. Sneed, H. & Mercy, A. (1985) Automated Software Quality Assurance. IEEE Transaction Software Engineering 11(9), 909-916.
32. Swanson, E. B. (1999). IS Maintainability: Should It Reduce the Maintenance Effort? Proceedings of the ACM SIGCPR conference on Computer personnel research. New Orleans LA, USA, 164-173.

AUTHORS PROFILE


Photo

Kiran Narang is a research scholar in Computer Science & Engineering Department of SRM University, India. She has published many research papers in conferences and journals. She is working in the field of Software Engineering. Her interest includes software engineering and cloud computing.

.

Puneet Goswami Received Ph.D. in Computer Science & Engineering from Guru Jambheshwar University of Science & Technology (GJUS &T), Hisar ,India in 2013. He is working as professor in Computer Science & Engineering Department at SRM University, Delhi NCR, India. His research interest includes Big Data, Cloud and Software Engineering.