Applied Computing and Informatics (2014) **10**, 38–51

King Saud University
Applied Computing and Informatics
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

An adaptive neuro fuzzy model for estimating the reliability of component-based software systems

Kirti Tyagi [a,*], **Arun Sharma** [b]

[a] *Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, India*
[b] *Department of Computer Science and Engineering, Krishna Institute of Engineering and Technology, Ghaziabad, India*

Received 15 January 2014; revised 16 March 2014; accepted 24 April 2014
Available online 6 May 2014

KEYWORDS

Neuro fuzzy;
Component-based software systems (CBSS);
Fuzzy;
Reliability;
Reliability model

Abstract Although many algorithms and techniques have been developed for estimating the reliability of component-based software systems (CBSSs), much more research is needed. Accurate estimation of the reliability of a CBSS is difficult because it depends on two factors: component reliability and glue code reliability. Moreover, reliability is a real-world phenomenon with many associated real-time problems. Soft computing techniques can help to solve problems whose solutions are uncertain or unpredictable. A number of soft computing approaches for estimating CBSS reliability have been proposed. These techniques learn from the past and capture existing patterns in data. The two basic elements of soft computing are neural networks and fuzzy logic. In this paper, we propose a model for estimating CBSS reliability, known as an adaptive neuro fuzzy inference system (ANFIS), that is based on these two basic elements of soft computing, and we compare its performance with that of a plain FIS (fuzzy inference system) for different data sets.

© 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.

* Corresponding author. Tel.: +91 9990802717.
E-mail address: kirti.twins@gmail.com (K. Tyagi).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

2210-8327 © 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.aci.2014.04.002

1. Introduction

Software generally has two user requirements: reliability and availability. Reliability is required when the product's nonperformance will have the greatest impact, while availability is required when downtime will have the greatest impact. Although it is difficult to formally define reliability, we cannot simply define it as a binary notion by saying that if a program is correct its reliability is 1, and if it is incorrect its reliability is 0. Instead, reliability can usually be measured probabilistically as

$$R_{\mathrm{SYS}} = (1 - \text{probability of failure}).$$

Specifically, software reliability is defined as a software system's probability of failure-free operation for a specified period of time in a specified environment. As the complexity of software applications continues to grow, greater emphasis has been placed on reuse. Therefore, component-based software system (CBSS) applications have come into existence.

Component-based software engineering (CBSE) is a specialized form of software reuse concerned with building software from existing components, including commercial off-the-shelf (COTS) components, by assembling them together in an interoperable manner. Achieving a highly reliable software application is a difficult task, even when high-quality, pretested, and trusted software components are combined. Several techniques have therefore emerged to analyze the reliability of component-based applications. These fall into two groups:

- System-level reliability estimation: reliability is estimated for the application as a whole.
- Component-based reliability estimation: application reliability is estimated based on the reliability of the individual components and their interconnection mechanisms.

Traditional approaches to software reliability analysis treat the software as a whole and use test data during the software test phase to model only the software's interactions with the outside world. These are known as black-box models. However, black-box models ignore the structure of software constructed from components as well as the reliability of individual components and are therefore not appropriate for modeling CBSS applications. Recently, soft computing techniques have emerged. Since reliability is a real-world issue, many run-time parameters are associated with reliability. This makes soft computing techniques ideal for estimating CBSS reliability, as these techniques deal mainly with uncertainty. The two basic soft computing techniques are neural networks and fuzzy logic. In this paper we combine these two techniques to estimate CBSS reliability.

This paper presents an adaptive neuro fuzzy inference system (ANFIS) model for estimating CBSS reliability. In a fuzzy inference system (FIS), the if-then rules

are formulated on the basis of expert advice. However, this is a relatively time-consuming process. The advantage of an ANFIS over the FIS model is that it combines fuzzy logic with the learning capabilities of neural networks (NNs) to solve this problem. The aim of our proposed model is to integrate the best features of fuzzy logic and neural networks in a CBSS reliability estimation model. Our results demonstrate how this is an improvement over an FIS.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 provides the framework for our proposed model, and Section 4 presents the model. Section 5 presents our experiments to evaluate the proposed model. Section 6 discusses the experimental results, and Section 7 presents our conclusions.

2. Related work

Researchers have proposed a number of models for estimating CBSS reliability. We can categorize the approaches into three types:

- Architecture-based reliability models.
- Mathematical models for estimating CBSS reliability.
- Soft computing techniques for estimating CBSS reliability.

Architecture-based reliability models such as state-based and path-based models are most suitable for CBSSs (Popostojanova and Trivedi, 2001; Cai et al., 2003; Gokhle, 2007). CBSS reliability depends not only on the architecture but also on the operational profile for the input. Heiko and Koziolek (Koziolek and Becker, 2005) describe the role of a component as a transformer from one operational profile to another. In state-based models (Cheung, 1980; Wang et al., 2006; Gokhle et al., 1998; Littlewood, 1979), control flow between components is taken into consideration. It is assumed that components fail independently. These models consider transfer among components to be Markov behavior, which means that the current behavior of a component is independent of its previous behavior. These models can be represented in two ways, namely, as composite models or as hierarchical models. Path-based models (Shooman, 1976; Krishnamurthy and Mathur, 1997) consider possible execution paths for estimating the reliability of an application. A sequence of components along different paths is obtained by either algorithmic or experimental testing.

Yacoub et al. (2004) propose an approach to reliability analysis called *scenario-based reliability analysis*. This approach introduces component dependency graphs (CDGs) which can be extended for complex distributed systems. Using the same algorithm, sensitivity can be analyzed as a function of component reliabilities and link reliabilities. This approach is based on scenarios which can be captured with sequence diagrams, which means that the approach can be automated. A limitation of this approach is that it does not consider failure dependencies among

the components. Zhang et al. (2008) also proposed a model based on a CDG. In this approach, a system's operational profile is given. It is assumed that control flow transits from component i to component j, and the reliability of component j is calculated as $T_{ij} = R_{ij} \times W_{ij}$, where

T_{ij} = the transition probability from component i to component j,
R_{ij} = reliability vector for each subdomain of component j, and
W_{ij} = weight vector for each subdomain of component j in transition from component i to component j.

This approach can characterize component reliability for an application when there are changes in the system's operational profile.

Dong et al. (2008) proposed a new model for estimating CBSS reliability in which various complex component relationships are analyzed. The Markov model is used to solve these complicated relationships, which have a large impact on a system's reliability. The results were used to develop a new tool to calculate software application reliability. Huang et al. (2008) proposed a technique based on algebra which provides a framework for describing the syntax and predicting the reliability of a CBSS, implemented on Seth et al. (2010) proposed a minimum spanning-tree-based approach to estimating CBSS reliability.

Goswami and Acharya (2009) proposed an approach to CBSS reliability analysis which takes the component usage ratio, which is the time allotted for a component's execution out of the application's overall execution time, into consideration. Mathematical formulas are used to calculate the usage ratio. Due to the flexibility of the component usage ratio, this approach can be used in real-time applications.

Fiondella et al. (2013) proposed an approach based on correlated component failures (COCOF). In this paper an efficient approach to access the reliability of a software application, considering the component reliabilities, correlation and application architecture is proposed. This proposed approach is based on an algorithm that transforms a Multivariate Burnoulli distribution (MVB) into a joint distribution of the component outcomes.

Palviainen et al. (2011) proposed a technique for reliability estimation, prediction and measuring of CBS. The proposed approach explains that reliability is a key driver for safety critical systems such as health care systems and traffic controllers. This paper gives software reliability evaluation during design and implementation phases. The contribution of this approach lies in the integrating the component level reliability. There is a need for a systematic approach that facilitates software developers to both construct reliable component based software systems and ensure that the software architecture, selected components and constructed software system meet the reliability goals. The approach integrates heuristic reliability estimation, model base reliability prediction and model based reliability measuring activities at component level and reliability prediction

activities at system level to support the incremental and iterative development of reliable CBSS. This proposed approach is not applicable for distributed systems.

Hu et al. (2013) proposed a software reliability estimation method using modified adaptive testing (MAT) for reliability of CBS. In this paper, a set of extended metrics based on the Nelson's software reliability model account for information gained from a user's point of view regarding the severity of the observed failures. In the approach, the use of test history information allows the resulting test process to be adaptive in the selection of tests under the limited test budget. This approach can enhance the software reliability estimation testing by guiding test case selection process by providing more descriptive and accurate results.

Dimov and Sasikumar (2010) proposed a fuzzy reliability model for CBSSs, based on fuzzy logic and possibility theory. A mathematical fuzzy model based on necessity and possibility is proposed to predict the reliability of a CBSS. Like many other models, this model does not require component failure data because it is based on uncertainty. However, a mechanism is necessary to model the propagation of failure between components and failure behavior.

Si et al. (2011) proposed a framework for estimating reliability through a component composition mechanism. The approach proposes five basic component composition mechanisms and techniques for their reliability estimation. After calculating the reliability for each composition, a procedure estimates the overall application reliability based on the component composition mechanisms and component utilization frequencies. It is possible to recognize additional composition mechanisms.

Many soft computing approaches are based on support vector machines (SVMs), genetic algorithms (GAs), and fuzzy logic. Lo (2010) proposed a software reliability estimation model based on an SVM and a GA. This model specifies that recent failure data alone are sufficient for estimating reliability. Reliability estimation parameters for the SVM are determined by the GA. This model is less dependent on failure data than are other models.

Hsu and Huang (2011) proposed a model which is an adaptive approach for testing path reliability estimation for complex CBSSs. Three methods were proposed for path reliability estimation: these use sequence, branch, and loop structures. The proposed path reliability can be used to estimate the reliability of the overall application. Tyagi and Sharma (2012) proposed an approach based on fuzzy logic for estimating CBSS reliability. In this approach, four critical factors were identified for estimating the reliability of a CBSS, and these were used to design an FIS for the estimation.

3. Framework for our proposed model

The present paper proposes a neuro fuzzy reliability model. The difference between this model and other models that have been proposed is that this model is fully based on CBSS characteristics. Fuzzy logic has been widely studied in

different fields of engineering with varying degree of success. An FIS often consists of if-then rules. These rules are fired in parallel: when if- conditions are met, the consequents of those rules fire to the degree to which the antecedents of the rules are met. One of the main challenges in creating an FIS is determining the fuzzy sets and fuzzy rules to be used. This requires acquiring deep domain knowledge from human experts, and the fine tuning of the fuzzy sets and rules can be very time consuming. The solution to this problem is to combine an FIS with the learning capabilities of NNs, resulting in an ANFIS. The proposed model is an improvement over the fuzzy model proposed by Tyagi and Sharma (2012).

3.1. Neural networks and fuzzy logic

NNs and fuzzy logic are complementary concepts. NNs have learning capabilities: they can learn from data and feedback. They are black-box models. Fuzzy logic models are rule-based models, where the rules usually have an if-then form. Fuzzy models do not have learning capabilities, so for learning, they must adopt techniques from other methods. It is therefore natural to marry the two technologies. Fuzzy logic is based on the combination of four concepts shown in Fig. 1.

3.2. ANFIS

ANFISs were first proposed by Jang (1992) in 1992. These are adaptive networks whose function is similar to FISs. The aim of an ANFIS is to integrate the best features of fuzzy systems and NNs. The advantages of an ANFIS over an FIS are as follows:

- An ANFIS can simulate and analyze the mapping relation between input and output data through a learning algorithm to optimize the parameters of a given FIS.
- An FIS is fully dependent on its membership functions. An ANFIS involves networks which combine nodes and directional links, and some learning rules are associated with these networks. The networks learn relationships between inputs and outputs.
- An ANFIS can be trained without any need for the expert knowledge usually required for the standard fuzzy logic design.

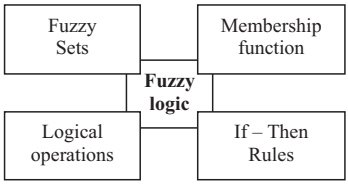

Figure 1 Working of fuzzy system.

4. Proposed model

In this paper, we propose an automated model for CBSS reliability estimation that uses the ANFIS approach. An ANFIS is a class of adaptive networks which are functionally equivalent to FISs. This is an extension of the fuzzy model proposed by Tyagi and Sharma (2012). We take into consideration all four factors proposed in that model to compare the results of both models.

The four factors proposed in Tyagi and Sharma (2012) are as follows. There are two main factors for estimating component reliability, the reusability of the component and its operational profile.

4.1. Reusability of the component

Reusability is one of the most basic concepts of component-based development (CBD). As the name suggests, reusability refers to how frequently a component is used in different applications. We selected reusability as a factor for estimating component reliability because it is believed that components that have been used in many applications are highly reliable. Hence the reliability of a component is directly proportional to its reusability: component reliability α reusability

We selected components from the site www.ejbeans.com; reusability measures for these components are given on the site. Sharma et al. (2009) proposed another method of calculating reusability using artificial neural networks.)

Operational profile for the component: The operational profile (OP), a quantitative characterization of how the software will be used, is essential in any software reliability engineering application. An OP is a complete set of operations with their probabilities of occurrence. The reliability may be different for different OPs. The OP for any component describes the inputs supplied to the component.

Other factors may contribute to component reliability, but these are the two factors that have the greatest effect on a component's reliability. Because the components are platform independent, the measures of these two factors will be the same for all applications.

Two main factors are used to estimate interface reliability:

Component dependency: In a CBSS, various components are connected to one another to form a larger application. The components are dependent on each other to perform their functions, that is, the output of one component may serve as an input for another component. This dependency plays an important role for estimating the reliability of the overall application. If components are highly dependent on one another, the reliability of the system will be low.

There are several mechanisms for characterizing dependencies among components. For example, an adjacency matrix representation of dependencies indicates which dependencies exist among components. However, this representation does not explain the interactions between components. These interactions play an important role in relation to the dependencies of components and therefore their

reliability, and so we need a different way to represent the interactions. Sharma et al. (2009) proposed a dependency representation which is based on linked lists and implemented using Hash Map in Java. This representation can store dependencies along with other information like provided and required information, which can be used to analyze several interaction- and dependency-related issues.

Application complexity: The complexity of any CBSS application can be defined in terms of the number of components in that application and their interconnectivity. An application that has more components is said to be more complex and hence less reliable. Therefore, the complexity of an application is inversely proportional to its reliability:

$$\text{Application complexity} \propto \frac{1}{\text{reliability}}$$

Application complexity can be measured in terms of the number of components in the application. Where N is the number of components in a CBSS, we define the application complexity AC as

$$\text{AC} \propto N.$$

Our proposed model is an additive model. It does not explicitly take the architecture of the CBSS into consideration, but the measure of application complexity will be platform dependent because it comes from the glue code (integration code).

The FIS studied in this work has four input factors as described above, namely, reusability, application complexity, component dependency, and operational profile. For a first-order Sugeno fuzzy model, a rule set with fuzzy if-then rules is as follows:

Assumptions:

(1) R = reusability
(2) CD = component dependency
(3) AC = application complexity
(4) OP = operational profile

Rule 1:

If R is A_i and AC is B_i and CD is C_i and OP is D_i, then $f = p_iR + q_i\text{AC} + r_i\text{CD} + n_i\text{OP} + m_i$

where A_i, B_i, C_i, and D_i are fuzzy sets and p_i, q_i, r_i, n_i, and m_i are parameters that are determined during the training process.

There are a total of 81 rules in which *low* (L), *medium* (M), and *high* (H) are linguistic variables.

Nodes in the same layer have similar functions. ANFIS has the following five layers:

Layer 1: Every node in this layer is an adaptive node with a node function.

$O_{1,i}$ is the membership grade for the fuzzy set. Parameters for this layer are known as *premise* parameters.

$$O_{1,i} = \mu\_A\_i(R) \quad \text{for} \quad i = 1, 2, 3$$

where μA_{i} is the membership function for R.

Layer 2: Every node in this layer is a fixed node labeled $\prod$ whose output is the product of all incoming signals.

$$O_{2,i} = w_i = \mu\_A\_i(R)\mu\_B\_i(\mathrm{AC})\mu\_C\_i(\mathrm{CD})\mu\_D\_i(\mathrm{OP}) \quad \text{for} \quad i = 1, 2, 3$$

Each node output represents the firing strength of the rule.

Layer 3: Every node in this layer is a fixed node labeled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rule firing strengths. The output of this layer is called the *normalized firing strengths*.

Layer 4: Every node in this layer is an adaptive node with a node function.

$$O_{4,i} = w_i f_i = w_i(p_i R + q_i \mathrm{AC} + r_i \mathrm{CD} + n_i \mathrm{OP} + m_i)$$

The parameters for this node are referred to as *consequent* parameters.

Layer 5: The single node in this layer is a fixed node labeled $\sum$, which computes the overall output as the summation of all incoming signals (see Fig. 2).

$$\text{Overall output} = O_{5,1} = \sum w_i f_i \Big/ \sum w_i$$

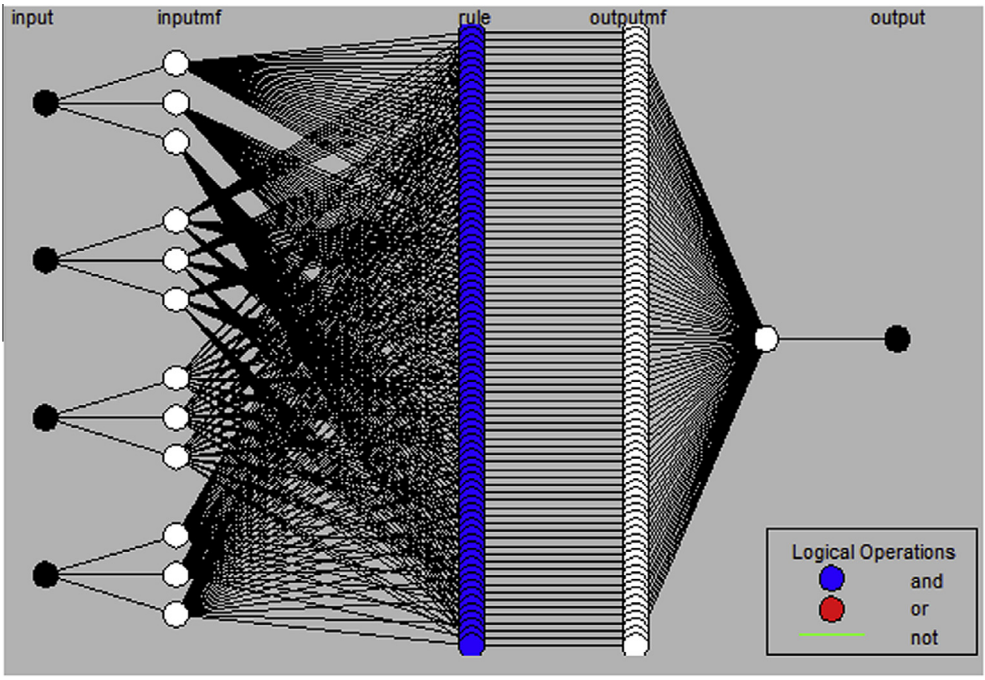

Figure 2 Proposed architecuture of the system.

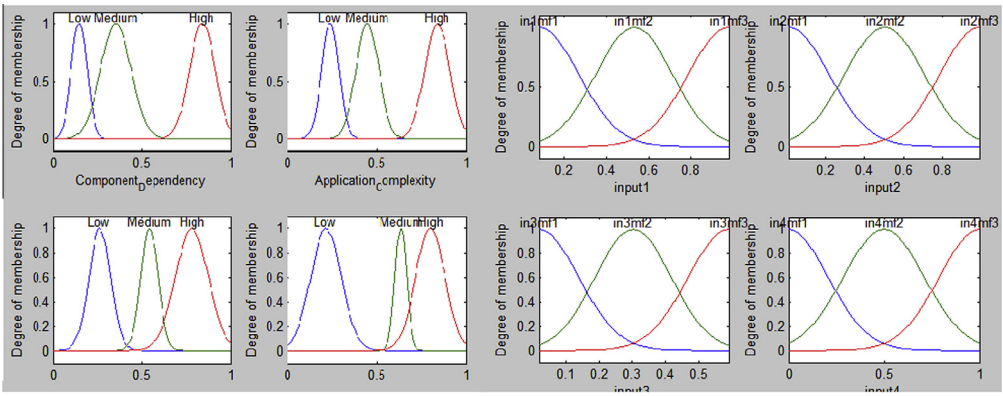

Figure 3 Membership function before training and after training.

One of the major limitations of this ANFIS is that it uses only Sugeno-style fuzzy inference systems. The difference between Sugeno and Mamdani inference engines lies in the defuzzification method. A Mamdani-type FIS uses the defuzzification technique for fuzzy outputs, and a Sugeno-type FIS uses a weighted average to compute crisp outputs. A Mamdani FIS also has output membership functions, while a Sugeno FIS doesn't have output membership functions.

5. Experimentation and evaluation of the model

We designed our ANFIS for a Sugeno FIS. To train the ANFIS, we collected data from 47 classroom-based projects. Some of these projects are complex CBSS sand

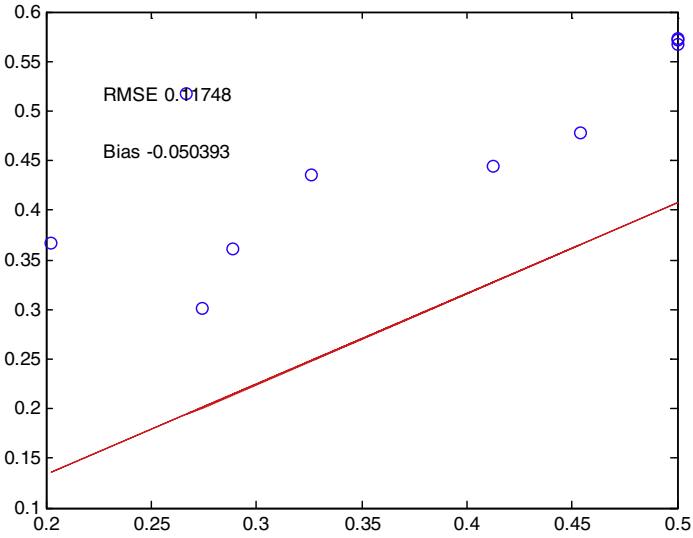

Figure 4 Root mean square error of FIS.

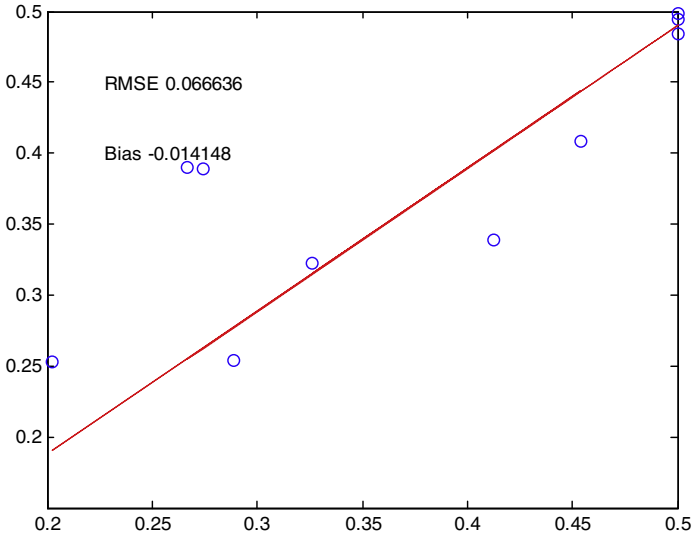

Figure 5 Root mean square error of ANFIS.

Table 1 Performance analysis of FIS and ANFIS.

Inputs				Output reliability		
CD	AC	*R*	OP	Original	FIS	ANFIS
.0202	.0202	.0202	.0202	.5000	.5667	.4837
.2424	.2424	.2424	.2424	.3263	.4351	.3228
.2323	.2323	.2323	.2323	.4127	.4438	.3390
.3131	.3131	.3131	.3131	.2025	.3672	.2526
.3434	.3434	.3434	.3434	.2891	.3604	.2541
.6970	.6970	.6970	.6970	.4543	.4772	.4079
.6667	.6667	.6667	.6667	.2667	.5178	.3896
.6465	.6465	.6465	.6465	.2740	.3013	.3890
.9798	.9798	.9798	.9798	.5000	.5734	.4942
.9293	.9293	.9293	.9293	.5000	.5710	.4984

some are simple CBSSs. The ANFIS was trained and tested using the data obtained from these projects. Some of the data were used for training and testing, and some were used for the results. A total of 81 rules were formed on the basis of the available data. The rules divide the input factors into three variables: *low*, *medium*, and *high*.

To evaluate our proposed model, the training data set was loaded, and the ANFIS was trained using that data set. To validate the model, we loaded the testing data set and then plotted the testing error. The membership function of the FIS before and after training is shown in Fig. 3.

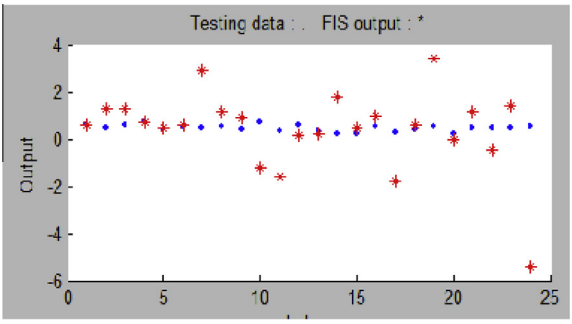

Figure 6 Testing error mapping sugeno FIS to ANFIS.

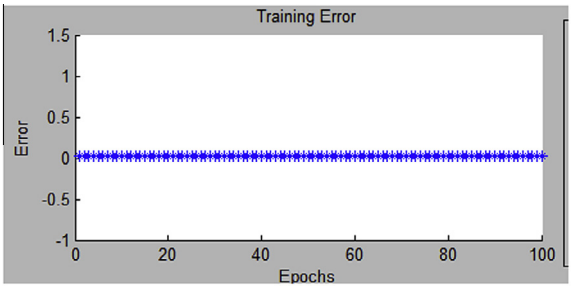

Figure 7 Training error mapping sugeno FIS to ANFIS.

6. Results and discussion

After creating the ANFIS model, we compared the output reliability values for different input sets with the original values. We calculated the root mean square error (RMSE) for the output obtained by the FIS and the output obtained by the ANFIS with the original output. The degree of membership had significant changes (see Figs. 4 and 5).

The difference between the two RMSEs can be observed. Using only the FIS, there was an 11.74% error in the results, but the ANFIS reduces the error to 6.66%. Hence, the ANFIS performs better than the FIS. Using the ANFIS, we first trained the FIS, and the rules were formed on the basis of training data to produce the output of the trained model. The only limitation of this model is that its execution is complex for large data sets. Our results show that the ANFIS model gives a more accurate measure of reliability than the FIS model (Table 1).

The testing error and training error obtained from the MATLAB FIS TOOLBOX is shown in Figs. 6 and 7.

7. Conclusion and future work

For any software system, there are two user requirements: reliability and availability. Reliability is required when the product's performance will have the greatest

impact. This paper proposes a neuro fuzzy model for estimating CBSS reliability. Many approaches to CBSS reliability have been proposed, some based on mathematical formulas and others based on soft computing techniques. Our proposed method is based on an ANFIS, which is a soft computing technique. This is a hybrid method that requires less computational time than traditional approaches and the previously proposed FIS approach. The CBSSs used to test the models are classroom-based projects. Our results show that the ANFIS improves the reliability evaluation of the FIS technique.

A limitation of our proposed model is that while the four factors for which the model was implemented are the most critical factors, there may be other relevant factors that should be used. One such factor is fault density, but currently we only have data available for the four factors, so the addition of this factor is left for future work.

References

Cai, K., Bai, C., Zhong, X., 2003. Introduction to reliability models of component based software system. J. Xi'an Jiaotong Univ. 37 (6), 560–564.

Cheung, R.C., 1980. A user oriented software reliability model. IEEE Trans. Softw. Eng. 6 (2), 118–125.

Dimov, Aleksandar, Sasikumar, Punnekkat, 2010. Fuzzy reliability model for component-based software systems, 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 39–46.

Dong, W., Huang, N., Ming, Y., 2008. Reliability analysis of component-based software based on relationships of components, IEEE Conference on Web Services, pp. 814–815.

Fiondella, Lance, Rajasekaran, Sanguthever, Gokhale, Swapana, 2013. Efficient software reliability analysis with correlated component failures. IEEE Trans. Reliab. 62 (1), 244–255.

Gokhle, S.S., 2007. Architecture based software reliability analysis: overview and limitations. IEEE Trans. Dependable Secure Comput. 4 (1), 32–40.

Gokhle, S.S., Dong, W.E., Trivedi, K.S., Horgan, J.R., 1998. An analytical approach to architecture based software reliability prediction, Proc. of the Third International Computer Performance and Dependability Symposium, Durham, NC, pp. 13–22.

Goswami V., Acharya, Y.B., 2009. Method for reliability estimation of COTS components based software systems, International Symposium on Software Reliability Engineering.

Hsu, C., Huang, C., 2011. An adaptive reliability analysis using path testing for complex component based software systems. IEEE Trans. Reliab. 60 (1), 158–170.

Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong, Aditya P. Mathur, 2013. Enhancing software reliability estimates using modified adaptive testing, Information and Software Technology Journal Elsevier, pp. 288–300.

Huang, N., Wang, D., Jia, X., 2008. FAST ABSTRACT: an algebra-based reliability prediction approach for composite web services, 19th International Symposium on Software Reliability Engineering, pp. 285–286.

Jang, J.R., 1992. ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Systems, Man, and Cybernetics.

Koziolek H., Becker, S., 2005. Transforming operational profiles of software components for quality of service predictions, Proc. 10th WCOP'05, pp. 1–8.

Krishnamurthy, S., Mathur, A.P., 1997. On the estimation of reliability of a software system using reliabilities of its components, Proceeding of the Eighth International Symposium on Software Reliability Engineering, Albuquerque, NM, pp. 146–155.

Littlewood, B., 1979. Software reliability model for modular program structure. IEEE Trans. Reliab. 28 (3), 241–246.

Lo, J., 2010. Early software reliability prediction based on support vector machines with genetic algorithms, Fifth IEEE Conference on Industrial Electronics and Applications, pp. 2221–2226.

Palviainen, Markov, Evesti, Antti, Ovaska, Eila, 2011. The reliability estimation, prediction and measuring of component – based software. J. Syst. Softw., 1054–1070.

Popostojanova, K.G., Trivedi, K.S., 2001. Architecture based approach to reliability assessment of software systems. Perform. Eval. J. 45 (2), 179–204.

Seth, K., Sharma, A., Seth, A., 2010. Minimum spanning tree-based approach for reliability estimation of COTS-based software applications. IUP J. Comput. Sci. 4 (4), 13–21.

Sharma, A., Grover, P.S., Kumar, R., 2009. Dependency analysis for component based software systems. ACMSIGSOFT Softw. Eng. Notes 34 (4), 1–6.

Shooman, M., 1976. Structural models for software reliability prediction, Proceeding of the Second International Conference on Software Engineering, San Francisco, CA, pp. 268–280.

Si Yuanjie, Xiaohu Yang, Xinyu Wang, Chao Huang, Aleksander J. Kavs, 2011. An architecture-based reliability estimation framework through component composition mechanisms, 2nd International Conference on Computer, Engineering and Technology, pp. 165–170.

Tyagi, K., Sharma, A., 2012. A rule-based approach for estimating the reliability of component-based systems. Adv. Eng. Softw. 54, 24–29.

Wang, W., Pan, D., Chen, M.H., 2006. Architecture-based software reliability modeling. J. Syst. Softw. 79 (1), 132–146.

Yacoub, S., Cukic, B., Ammar, H., 2004. Scenario based reliability analysis approach for component based systems. IEEE Trans. Reliab. 53 (4), 465–480.

Zhang, F., Zhou, X., Chen, J., Dong, Y., 2008. A novel model for component-based software reliability analysis, 11th IEEE High Assurance Systems Engineering Symposium, pp. 303–309.