Research Journal of Applied Sciences, Engineering and Technology 4(24): 5304-5312, 2012
ISSN: 2040-7467

Submitted: March 14, 2012 Accepted: April 08, 2012 Published: December 15, 2012

ANFIS Approach for Optimal Selection of Reusable Components

[1]K.S. Ravichandran, [2]P. Suresh and [3]K.R. Sekr
[1]School of Computing, SASTRA University, Thanjavur-613 401, Tamil Nadu, India
[2]Department of Computer Science, Paavendhar College of Arts and Science, Salem
[3]School of Computing, SASTRA University, Thanjavur -613 401

Abstract: In a growing world, the development of modern software system requires large-scale manpower, high development cost, larger completion time and high risk of maintaining the software quality. Component-Based Software Development (CBSD) approach is based on the concept of developing modern software systems by selecting the appropriate reusable components or COTS (Commercial Off-The-Shelf) components and then assembling them with well-defined software architecture. The proper selection of COTS components will really reduce the manpower, development cost, product completion time, risk, maintenance cost and also it addresses the high quality software product. In this paper, we develop an automated process of component selection by using Adaptive Neuro-Fuzzy Inference Systems (ANFIS) based technique by using 14 reusable components' parameters as a first time in this field. Again, for increasing the accuracy of a model, Fuzzy-Weighted-Relational-Coefficient (FWRC) matrix is derived between the components and CBS development with the help of 14 component parameters, namely, Reliability, Stability, Portability, Consistency, Completeness, Interface & Structural Complexity, Understandability of Software Documents, Security, Usability, Accuracy, Compatibility, Performance, Serviceability and Customizable. In the recent literature studies reveals that almost all the researchers have been designed a general fuzzy-design rule for a component selection problem of all kinds of software architecture; but it leads to a poor selection of components and this paper suggests adoption of a specific fuzzy-design rule for every software architecture application for the selection of reusable components. Finally, it is concluded that the selection of reusable components through ANFIS performs better than the other models discussed so far.

Keywords: ANFIS, CBSD, fuzzy logic, reusable components

INTRODUCTION

Before the year 2000, the effectiveness of software development largely depended on structured and object oriented programming, abstract data type, modeling and special task based languages. But in the present era, the component based approach to software development has seen tremendous growth and has become very popular (Abreu and Goulão, 2001; Szyperski *et al*., 2004). Component-based design is an effective way of implementing "design-and-reuse" strategy. However, given a library of components with varying performance parameters and quality-of-service metrics, it is often difficult to select the right components that will satisfy the System Requirements.

Software Modularity (SM) was used very much in the CBSS development as well as conventional software systems development to improve flexibility and adaptability of software systems (Parsa and Bushehrian, 2004). Automatic reverse engineering process has been achieved by Khoshgoftaar *et al*. (2004) through resource allocation and ranking of software modules. Information theory was employed to perform SM (Chapin, 2002; Sarkar *et al*., 2007). The quality of SM was achieved through information theoretic metrics approach by Sarkar *et al*. (2007). SM was also classified through optimization algorithms like Genetic Algorithms (Khoshgoftaar *et al*., 2004; Parsa and Bushehrian, 2004), Evolutionary Algorithms (Mitchell and Mancoridis, 2006, 2008; Kwong *et al*., 2010), Fuzzy logic (Ioana and Doru, 2006; Kirti *et al*., 2010; Shrddha *et al*., 2010), Soft Computing (Harpreet and Vishal, 2011; Raj Kiran and Ravi, 2007) and clustering techniques (Land *et al*., 2008; Mitchell and Mancoridis, 2006).

In recent years, many researchers have been concentrating on the selection of COTS components and their studies are based on the service and qualitative parameters like customizability, interface complexity, portability, document quality, reliability etc., Reuse based on object oriented techniques was proposed by Sindre *et al*. (1993) who has taken four parameters for measuring COTS components, namely, portability, flexibility, understandability and confidence to estimate reusability. Rotaru *et al*. (2005) has measured components through parameters adaptability, complexity and compose-ability. Sharma *et al*. (2008) and Gill and Balkishan (2008) have

Corresponding Author: K.S. Ravichandran ,School of Computing, SASTRA University, Thanjavur-613 401, Tamil Nadu, India

proposed interface complexity as a measure of quality of the components. Shrddha *et al*. (2010) proposed to measure reusability of software components through customizability, interface complexity, portability and document quality parameters.

During the last two decades, the development of modularity based conventional systems followed the criteria; "minimizing the coupling and maximizing the cohesion of software modules" (Ian, 1993; Allen *et al*., 2001; Parsa and Bushehrian, 2004; Kwong *et al*., 2010). Before the year 2008, all the researchers considered homogenous components, whereas Kwong *et al*. (2010), for the first time proposed to find the maximum cohesion and minimum coupling by using heterogeneous software components.

In this study, we develop an automated process of component selection by using ANFIS approach using 14 parameters, namely, Reliability, Stability, Portability, Consistency, Completeness, Interface & Structural Complexity, Understandability of Software Documents, Security, Usability, Accuracy, Compatibility, Performance, Serviceability and Customizable. If we consider a large number of parameters to analyze and select low cost, most appropriate reusable components, it will result in a time consuming process. But at the same time we get a low cost, quality and most appropriate components with respect to the proposed software architecture.

Software reusability factors: Modern software system requires the above 14 attributes to select apt COTS components for any proposed architecture. Not all the attributes are essential for any proposed architecture but some of the attributes are essential for all kinds of architecture, namely, Interface Complexity, Document understandability, Portability, Compatibility, Customizable and Completeness. The following are the factors that affect the component selection efforts:

Reliability (Re): Reliability is the probability of a device fulfilling and performing its purpose adequately for the period of time intended under the operating conditions encountered. Reliability at the component level employs both qualitative information based on physical principles and quantitative reasoning based on data available on the component's documentation. Reliability estimates based on physical principles, use knowledge of the coding in the component and their suitability to various software development conditions. It is used to estimate the reliabilities of populations of specific components, quantitatively. Katerina and Trivedi (2001) describes that there is a necessity/need for modeling approaches that are capable of considering the architecture of the software and estimating the reliability with the consideration of components interaction, the utilization of the components and the reliabilities of the components and their interfaces with other components.

Stability (St): Stability states that the system requirements of the component are satisfied throughout its lifetime by its services. If any of the changes take place in the reusable component, the stability factor of that component is to be measured for having the same component as reusable one. Stability involves achieving consistent and higher process yields. The reusable component should be stable in any environment as the components are portable to the systems where work load is varying and many processes are depending on it.

Portability (Po): Portability can be described as the property that supports migration of component from one system to another with little or no modifications. Hence, it is easily portable to specified new environments with less cost and time. A portable component can be easily reinstalled from distribution files to another computer of same architecture. If the components are platform independent, then it is highly portable and hence its selection efforts are low. The portability of a component correlates with its usability. The ease of portability of a component can determine its level of reuse.

Consistency (Cy): A component is consistent if it shows same performance for any number of times under any load. This can be checked with the architectural description of software systems, which assumes that the system engineer will specify a metamodel and record the design against that metamodel. The metamodel will comprise of entities, relationships and constraints. The consistency constraints of reusable components are to be satisfied if the architecture being described is to be consistent.

Completeness (Cs): In addition to rules for checking consistency of architecture as it is developing, there will be many rules that specify completeness of the description. An example of such a rule for the metamodels used might be that every component should have at least one interface that it either requires or supplies. Otherwise, the completeness property is not satisfied. A component is complete when it satisfies all its constraints and produces the desired output. A component has to maintain its completeness property to enable portability property and its reuse.

Interface and structural complexity (In): The components are black box in nature where the source code is not available. Application may interact with the components, only through the well-defined interface. Interface acts as a primary source for understanding, using, implementation and maintenance of the

component. For better reusability, interface complexity should be as low as possible. Structural Based Complexity may be applied to identify the cross communication among components and interior communication among the component elements to do the particular system requirement. If a component has more complexity then it may not be reusable.

Understandability of software documents (Du): Software is a collection of programs, procedures and documentation that eases the hardware use. To understand software component, documentation quality of software is very crucial. The best-known metric to judge the quality of documentation is Gunning's Fog Index, which is a measure of the readability of a passage or text. The Fog Index is based on length of sentences and the number of difficult words where difficulty of a word is based on the number of syllables in the word. Lower value of Fog Index indicates better quality of the documentation while higher value is a reflection of poor documentation quality.

Customizable (Cu): Customization allows the application to be divided into a number of modules that can be developed and maintained separately. By implementing commonly used logic within custom components, a suite of reusable components can be built. One reason for customization is modification of an existing component to satisfy the requirements of an application. Another reason to customize a component is to add a new logic or behavior to it. An example of customization is to add label property to the button. The selection of reusable component also depends on this customization property.

Accuracy (Ac): Accuracy is the degree of closeness to a real system. It should not be changed even after much number of usages. Precision, the measurement how close the values are to each other and have a close relation with accurate values. Higher precision does not mean greater accuracy nor vice versa, but both are important in the calculation of the efficiency of the system. Various applications require different accuracy levels. Accuracy is basically related to the data quality and errors in the dataset. As the number of errors reduce the accuracy of component increases.

Performance (Pe): Performance can be described as latency and response time measurement, based on user load conditions. It further describes the system performance in terms of stability and response time under certain workload. It also used to investigate, verify, measure, or validate various systems' quality determining attributes. By the identification of performance bottlenecks, performance tuning can be done by localization of a small number of performance-critical components. Components can be internally optimized without affecting their specification to improve performance. Also, they can be moved between platforms, without affecting the functionality or usability of the application.

Security (Se): Security describes the integrity of a system and its users. Security plays a vital role in authorization and authentication of users. It is further used for the secured transportation of information. Access can be controlled through identification and authentication. To migrate the risk and protect component, security controls such as administrative controls which form a framework and consist of approved written policies, procedures, standards and guidelines, logical controls which use software and data to monitor and control access to information and computing systems, physical controls which monitor and control the environment of the work place and computing facilities are used.

Usability (Us): Usability is a quality attribute that assesses how easy user interfaces are to use. This also refers to methods for improving ease-of-use during the design process. It is the effectiveness and efficiency of a system to meet user's needs resulting in user satisfaction and productivity. Usability is a property of any reusable component and it serves its purpose only when the application that uses it, is designed flexible. The quality of reusable component can be measured through its learning ability, efficiency, memorability, errors and satisfaction.

Scalability (Sc): Scalability is a property to raise the capacity (mainly, the users) of deployed system over a period. It involves of addition of resources to the system without any changes in the deployment architecture i.e. it represents the ease with which a system or component can be modified, added, or removed, to accommodate changing load. It also represents the capability of application services and deployment of environments to cope with increased user requests, operations, transactions and data volume to provide same or better performance. Volume, software and hardware are three dimensions of scalability. It is easy to design a component that does not hold its state inside, whereas the scalability options are reduced if a component holds its state.

Serviceability (Sr): It is the ease with which a deployed system can be maintained, monitored, repaired and its components are upgraded. If the components are incorporated in a system its maintenance becomes difficult as the source codes are not available. So the Serviceability factor should ensure that the component is maintainable. The value of a component improves with the availability of serviceability and maintenance.

A Neuro-fuzzy technique is a combination of artificial neural networks and fuzzy inference systems. Fuzzy inference system is a methodology that uses fuzzy logic to optimize the mapping of the given input and output data sets. The decisions are made based on mapping. The FIS process with Mamdani model involves fuzzification, design of fuzzy-based systems using fuzzy if-then rules, fuzzy aggregator and defuzzification.

In this section, ANFIS architecture with Mamdani model is designed and it is used to determine the optimal component parameters required to implement the required software architectural design using reusable components. ANFIS is a hybrid-learning algorithm, which is used to identify and train the membership functions to produce single-output. In Mamdani-type inference system, the output membership functions are converted into fuzzy sets. To train the FIS membership function parameters, LS and BP gradient descent method combination is used.

DESIGN OF ANFIS BASED COMPONENT SELECTION METHODOLOGY

The designed ANFIS architecture can have 14 fuzzy inputs and one fuzzy output. The fuzzy inputs are Reliability, Stability, Portability, Consistency, Completeness, Interface & Structural Complexity, Understandability of Software Documents, Security, Usability, Accuracy, Compatibility, Performance and Customizable, Serviceability and the fuzzy output variable is Component suitability factor. All the fuzzy input variables are linguistically divided into three variables, namely Low (L), Medium (M) and High (H). The output variable called 'Suitability of Reusable Components Selection (SRCS)' is linguistically divided into five variables, namely Low (L), Low Medium (LM), Medium (M), High Medium (HM) and High (H); and its membership function is given in Fig. 1. The theoretical background, expert knowledge and Turing test can be used to evaluate the fuzzy membership functions. The membership diagram for the input variable 'Reliability' is given below:

All input membership functions are constructed with the help of combination of Gaussian Membership function and Bell-shaped membership function; and triangular membership functions alone is used to construct SRCS membership function and it is given in Fig. 2.

The advantages of the ANFIS procedure for optimal selection of the reusable components are:

- Of the distinct software developers, that can be adjusted during ANN training
- Anomalies, if any, can be self corrected
- The final decision has been arrived taken into view not merely the fuzzy design rule but also the estimation of hybrid learning algorithm The membership function for Fuzzy decision matrix is

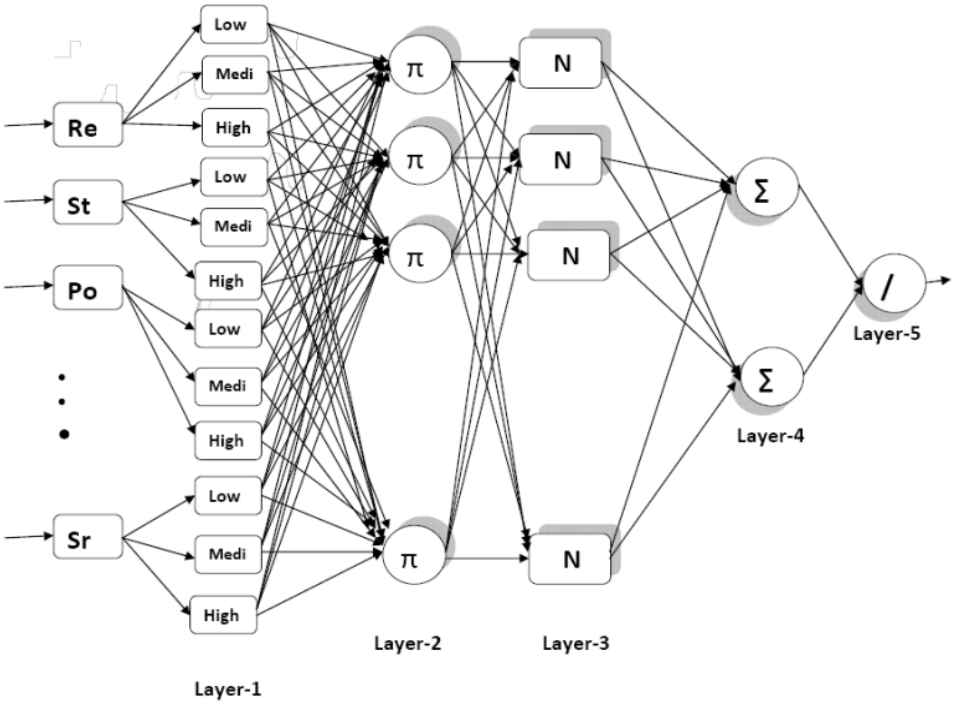

Fig. 1: ANFIS structure for a proposed reusable component selection problem

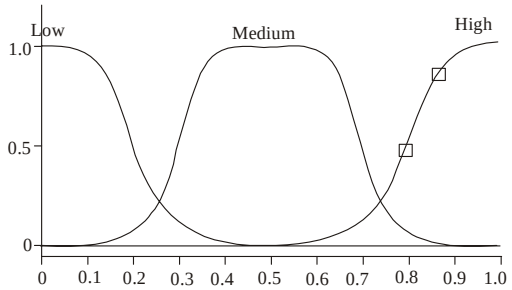

Fig. 2: Membership function for the reusable parameter 'reliability'

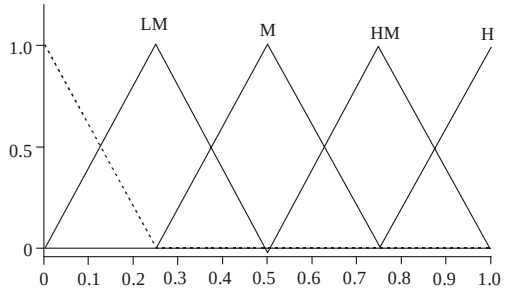

Fig. 3: Membership function for the output variable 'suitability of reusable component selection'

varying from software architecture to architecture and it is used to determine the 'Suitability of Reusable Component Selection'.

A typical ANFIS architecture is given in Fig. 3, where a square and square with rounded ends indicates an adaptive node and circles indicates a fixed node. This architecture has five layers and its layer-wise explanation is given below:

Layer 1: The first layer of the designed ANFIS architecture has 14 fuzzy input nodes, which are called attributes of components. The normalized input and output data are used to classify the essential requirement of attributes of a reusable component. Gaussian and Bell-shaped membership functions have been used to get the output of fuzzy-featured neurons and linear membership function is used to get the output of the non-fuzzy featured neurons. Symbolically, it is defined as:

$$O_{i,1} = \mu_{i,\,Low}(x), \text{ for } i = 1 \text{ to } 14$$

$$O_{i,2} = \mu_{i,\,Medium}(x), \text{ for } i = 1 \text{ to } 14$$

$$O_{i,3} = \mu_{i,\,High}(x), \text{ for } I = 1 \text{ to } 14$$

Layer-2: There are 3^{14} nodes used in this layer and all are fixed node, with the product of inputs as the output and it is labeled by π. Based on the fuzzy decision matrix some of the sample rules are given below:

- In the first example, let us select reusable components for designing an Internet Architecture Applications (IAA) oriented software development. Then the quality of service (QoS) for IAA software development mainly depends on the following parameters, namely, Performance, Security, Usability, Reliability, Consistency and completeness. Based on the software architectural design, initially, we have to formulate the corresponding fuzzy decision rule table based on their requirement and quality of service parameters. QoS parameters are often changeable even in the same application; the requirement specifications are different, so that the parameters selections are also different. Hence, common fuzzy decision table is not advisable to follow to determine reusable components and it always leads to less QoS. The following is the example of the fuzzy-design rule of the reusable component selection for web-based applications software development.
  *If (Re is in high) and (St is in low) and (Po is in medium) and (Cy is in high) and (Cs is in high) and (In is in low) and (Du is in high) and (Cu is in low) and (Ac is in low) and (Pe is in high) and (Se is in high) and (Us is in high) and (Sc is in low) and (Sr is in medium) then (output-component selection for IAA software development: SRCS is in high):
- In the second example, let us select reusable components for designing on OCR (Optical Character Recognition) based software development, then QoS for OCR project mainly depends on the parameters like, Performance, Consistency, Portability, Customizability, Scalability and Serviceability. In this software development, selection of the quality OCR component is the primary aim. In this case, performance of the component is the percentage of the character recognized correctly by the component; consistency of the component is that, all kinds of input will provide the same quality of the character recognition; Portability of the component is that, it can be applied to all kinds of languages, like English, French, Germany, Urdu etc., ; Customizability of the component is the provision of inserting other languages or updating the same component with new symbol of recognition; Scalability of a component is the measure of performance in terms of ease of accessibility, performance analysis and so on; Serviceability is the ease with which a deployed system can be maintained, monitored, repaired and its components can be upgraded. The following is the

example of the fuzzy-design rule of the reusable component selection for OCR based software development.

If (Re is in medium) and (St is in low) and (Po is in high) and (Cy is in high) and (Cs is in low) and (In is in low) and (Du is in high) and (Cu is in high) and (Ac is in low) and (Pe is in medium) and (Se is in low) and (Us is in low) and (Sc is in high) and (Sr is in high) then (output-component selection for OCR software development: SRCS is in high):

- In the third example, let us select reusable components for designing on BEA (Business Enterprise Applications) based software development, then QoS for full-fledged BEA development mainly depends on services provided by interactive product catalogue, security payment processing, business intelligence, RFID based billing systems, HR management, content management, online shopping, IT service management, manufacturing, resource planning, customer relationship management, forms automation and application integration. It mainly depends on the parameters Reliability, Portability, Consistency, Interface complexity, Customizability, Security, Usability and Serviceability. The following is the example of the fuzzy-design rule of the reusable component selection for BEA based software development.

  If (Re is in high) and (St is in low) and (Po is in high) and (Cy is in high) and (Cs is in low) and (In is in high) and (Du is in high) and (Cu is in high) and (Ac is in low) and (Pe is in medium) and (Se is in high) and (Us is in low) and (Sc is in medium) and (Sr is in high) then (output-component selection for BEA software development: SRCS is in high):

Hence, every software application development requires a new Fuzzy-Design Rule (FDR) for selecting reusable components through ANFIS approach. General fuzzy-design rules sometimes lead to select, unimportant reusable components. This is the advantage of the proposed methodology rather other methods:
Symbolically, it is defined as:

$$O_{2,k} = \left(\mu_{\mathrm{Re}}(x_1) \text{ and } \mu_{St}(x_2) \text{ and} \ldots \text{and } \mu_{Sr}(x_{14})\right)$$
$$= \mu_{SRCS}(z) \text{ for } k = 1 \text{ to } 3^{14}.$$

Each node output represents the firing strength of a rule.

The output of this layer is produced the weight value w_i, by using either max-min or max-product method. But here we have used max-product and it is explained in (Jang *et al*., 1997).

Layer-3: There are 3^{14} nodes assigned in this layer and every node in this layer is fixed node labeled N. Each node in this layer computes two constant values, namely a_i and z_i, where a_i = area of the compounded membership function obtained at node i and z_i = Centroid of the compounded membership function obtained at node i. Symbolically, the output of this layer is defined as:

$$O_{3i} = w_i a_i z_i \text{ for } i = 1, 2, 3, \ldots, 3^{14}$$

Layer-4: Two nodes are used in this layer which is fixed labeled $\sum$. The first node computes Σ wi ai zi and the second node computes $\Sigma\, w_i$.

Layer-5: The single node in the fifth layer is a fixed node labeled '/' that computes the overall output of the given ANFIS structure. Symbolically, the output of this layer is defined by:

$$O_{51} = \Sigma\, w_i a_i z_i / \Sigma\, w_i$$

It is observed that the overall output of ANFIS architecture with fixed premise parameters is:

$$z = \Sigma\, w_i a_i z_i / \Sigma\, w_i.$$

The optimal values of the consequent parameters can be found by using the Least-Square Method (LSM). When the premise parameters are not fixed, the search space becomes larger and the convergence of training becomes slower. This problem can be solved by use of hybrid learning algorithm combining the LSM and the back propagation algorithm. As the dimension of the search space in back propagation algorithm reduces, this algorithm converges faster. The layer1 premise parameters and the layer4 consequent parameters are tuned in training until FIS achieves the required response.

After exporting this system to the workspace, the network was trained using `anfis' function:

```
[fismat, error, stepsize, checkfismat, checkerror] =
anfis (traindata, initialfismat, trainoptions,
displayoptions, checkdata, optionalmethod).
```

RESULTS AND DISCUSSION

In this section, we have introduced FWRC (Fuzzy Weighted Relational Coefficient) concepts for quantifying fuzzy value between the reusable component parameters and the given software architecture; and explain how the FWRC value for the reusable components can be evaluated between the components' parameter and the given software architecture. Most of the components' parameters listed in section-2 are very difficult to quantify and some of the informations are straight away evaluated

through reusable components' documentation. In this section, we explain how FWRC value of the components parameter, namely 'Reliability' can be evaluated and in similar manner the other components parameters can also be evaluated.

- **Evaluation of FWRC for the components' parameter 'Reliability':** From the huge volume of reusable components available in the market, we have to select the most appropriate and low cost reusable components for the given software architecture. The evaluation procedure for finding FRC between the components' parameter 'Reliability' and the proposed software architecture is given below. Though reliability evaluation procedure is already discussed in Raj Kiran and Ravi (2007) through soft computing techniques. But, in the proposed work, first time the reliability is evaluated through FWRC method.

  In general, MTBF (Mean Time between Failures) Calculator supports 26 most known and accepted reliability prediction standards suggested in most of the Reliability Prediction Software. Software Components Failure Rate mostly depends on its past historical data:

  Let R_{ij} be the Fuzzy Weighted Relational Coefficient (FWRC) of 'Reliability' between the reusable components C_j and the given software architecture S_i. This can be evaluated from the following formula:

$$R_{ij} = \frac{\begin{array}{l} w_1.\mu_{\mathrm{Re}}(S_i,C_j) + w_2.\mu_{St}(S_i,C_j) + w_3.\mu_{Po}(S_i,C_j) + \\ w_4.\mu_{Cy}(S_i,C_j) + w_5.\mu_{In}(S_i,C_j) + w_6,\mu_{Du}(S_i,C_j) + \\ w_7.\mu_{Cu}(S_i,C_j) + w_8.\mu_{Sc}(S_i,C_j) + w_9.\mu_{Ac}(S_i,C_j) + \\ w_{10}.\mu_{Cs}(S_i,C_j) + w_{11}.\mu_{P_e}(S_i,C_j) + w_{12}.\mu_{Cy}(S_i,C_j) + \\ w_{13}.\mu_{Sr}(S_i,C_j) + w_{14}.\mu_{Us}(S_i,C_j) \end{array}}{\begin{array}{l} w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9 + w_{10} \\ + w_{11} + w_{12} + w_{13} + w_{14} \end{array}} \quad (1)$$

where,

$\mu_{Re}(S_i, C_j)$ = Fuzzy relational value between the software architecture S_i and the reusable component C_j.

$\mu_{Re}(.)$ = Fuzzy membership function for the parameter type 'Reliability'.

The value of $\mu_{Re}(S_i , C_j)$ is obtained by using the following procedure:

- The fuzzy value of all the 14 parameters with respect to the given software architecture S_i is given below:

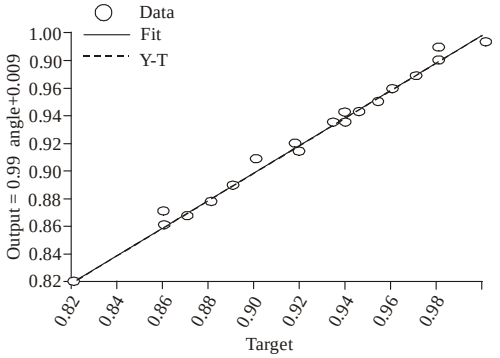

Fig. 4: Regression curve

$$\left\{\begin{array}{l} \frac{\mu_{Si\,(R_e)}}{R_e} + \frac{\mu_{Si\,(S_t)}}{S_t} + \frac{\mu_{Si\,(P_o)}}{P_o} + \frac{\mu_{Si\,(Cy)}}{C_y} + \frac{\mu_{Si\,(D_u)}}{D_u} \\ \frac{\mu_{Si\,(Cu)}}{C_u} + \frac{\mu_{Si\,(Cs)}}{Cs} + \frac{\mu_{Si\,(Ac)}}{A_c} \\ + \frac{\mu_{Si\,(Co)}}{C_o} + \frac{\mu_{Si\,(P_e)}}{Pe} + \frac{\mu_{Si\,(Us)}}{Us} + \frac{\mu_{Si\,(Sc)}}{Sc} + \frac{\mu_{Si\,(Sr)}}{Sr} \end{array}\right\}$$

- Similarly, the fuzzy value of all the 14 parameters with respect to the reusable component C_j is given below:

$$\left\{\begin{array}{l} \frac{\mu_{Cj\,(R_e)}}{R_e} + \frac{\mu_{Cj\,(S_t)}}{S_t} + \frac{\mu_{C_j\,(P_o)}}{P_o} + \frac{\mu_{Cj\,(C_y)}}{C_y} + \frac{\mu_{C_j\,(I_n)}}{I_n} + \frac{\mu_{C_j\,(D_u)}}{D_u} \\ + \frac{\mu_{C_j\,(C_u)}}{C_u} + \frac{\mu_{C_j\,(Cs)}}{Cs} + \frac{\mu_{C_j\,(A_C)}}{A_c} \\ + \frac{\mu_{C_j\,(C_o)}}{C_o} + \frac{\mu_{C_j\,(P_e)}}{Pe} + \frac{\mu_{C_j\,(Us)}}{Us} + \frac{\mu_{C_j\,(S_c)}}{Sc} + \frac{\mu_{C_j\,(Sr)}}{Sr} \end{array}\right\}$$

- To compute,

$$\mu_{R_e}(S_i,C_j) = \mu_{R_e}\left\{\frac{\min\left(\mu_{Si\,(R_e)},\mu_{Cj\,(R_e)}\right)}{\max\left(\mu_{Si\,(R_e)},\mu_{Cj\,(R_e)}\right)}\right\}$$

Equation (1) represents the Fuzzy Weighted Relational coefficient value of 'Reliability' between the reusable components Cj and the given software architecture Si and its value always lies between 0 and 1. The fuzzy membership function for the parameter is linguistically classified into three categories, namely, Low (L), Medium (M) and High (H). The membership diagram for the variable 'Reliability' is given in Fig. 4.

These fuzzy relational values are fed into the designed ANFIS classifier and this ANFIS classifier

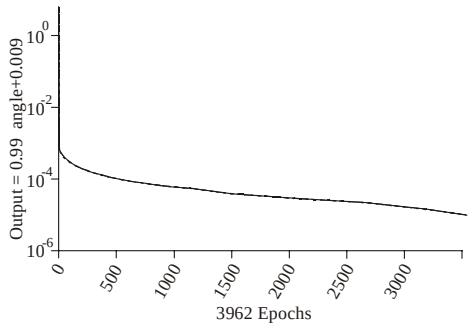

Fig. 5: Error curve

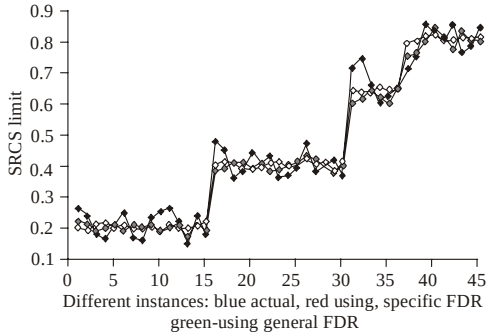

Fig. 6: Comparison between the actual, generalized FDR and a specific FDR

predicts the best 'Suitability of Reusable Components'. For training datasets of the reusable components based OCR software development projects, generalized FDR and a specific FDR based decision making with the permissible error limit 1e-05are applied. Again, the testing data sets are applied to the ANFIS classifier and it is found that the specific FDR based OCR software development is closely correlated with the actual value than a generalized FDR and it is given in Fig. 5.

Both Back propagation learning and hybrid learning rules are used to obtain the optimized ANFIS classifier and the optimization is achieved to 3962 epochs with permissible error le-05, regression coefficient 0.99547 and total processing time 2.43 min (Fig. 5 and 6).

CONCLUSION

While developing enterprise business application softwares, software developer undertakes multiple development tasks concurrently and it consumes lot of manpower, money and time. Reusable components address all those drawbacks and the selection of most appropriate, low cost and state-of-the-art quality reusable components is a difficult task. In this paper, Neuro-Fuzzy based approach is adopted to select optimal reusable components efficiently and it also tackles all the drawbacks. Further, the developed approach is validated with three data sets for three proposed software architecture, namely, IAA, OCR and BEA respectively with the help of generalized FDR and a specific FDR. The results which are determined, shows that the proposed approach is able to predict the reusability of these components with an acceptable accuracy. Further, we concluded that the derivation of a specific FDR with respect to each of the proposed software architecture is required and it provides high accuracy than it follows a generalized FDR.

REFERENCES

Abreu, F.B. and M. Goulão, 2001. Coupling and cohesion as modularization drivers: Are we being over-persuaded. Proceedings of the 5th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, Washington, DC, USA.

Allen, E.B., T.M. Khoshgoftaar and Y. Chen, 2001. Measuring coupling and cohesion of software modules: An information-theory approach. Proceedings of the 7th International Symposium on Software Metrics, pp: 125-134.

Chapin, N., 2002. Entropy-metric for systems with COTS software. Proceedings of the 8th IEEE Symposium on Software Metrics, pp: 173-181.

Gill, N.S. and Balkishan, 2008. Dependency and interaction oriented complexity metrics of component based systems. ACM SIGSOFT., 33(2): 1-5.

Harpreet, S. and K.T. Vishal, 2011. Neuro fuzzy logic model for component based software engineering. Int. J. Eng. Sci., 1: 303-314.

Ian, S., 1993. Software Engineering. Addison-Wesley Longman Publishing Co. Inc. Computing Program Modularizations using the k-Cut Method. In: Jermaine, C., 1999. Proceedings of the 6th Working Conference on reverse engineering, pp: 224-234.

Ioana, S. and T. Doru, 2006. Specification-based retrieval of software components through fuzzy inference. Acta Polytech. Hung., 3(3): 121-135.

Jang, J.S.R., C.T. Sun and E. Mizutani, 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice-Hall, Upper Saddle River, NJ.

Katerina, G.P. and K. Trivedi, 2001. Architecture-based approach to reliability assessment of software systems. Erformance Evaluat., 45: 179-204.

Khoshgoftaar, T.M., Y. Liu and N. Seliya, 2004. A multi-objective module-order model for software quality enhancement. IEEE T. Evolut. Comput., 8(6): 593-608.

Kirti, S., S. Arun and S. Ashish, 2010. Component selection efforts estimation-a fuzzy logic based approach. Int. J. Comput. Sci. Security (IJCSS), 3(3): 210-215.

Kwong, C.K., L.F. Mu, J.F. Tang and X.G Luo, 2010. Optimization of software components selection for component-based software system development. Comput. Ind. Eng., 58: 618-624.

Land, R., A. Alvaro and I. Crnkovic, 2008. Towards efficient software component evaluation: An examination of component selection and certification. Proceedings of the 34th Euromicro Conference of Software Engineering and Advanced Applications, pp: 274-281.

Mitchell, B.S. and S. Mancoridis, 2006. On the automatic modularization of software systems using the bunch tool. IEEE T. Software Eng., 32(3): 193-208.

Mitchell, B.S. and S. Mancoridis, 2008. On the evaluation of the bunch search-based software modularization algorithm: Soft computing-A fusion of foundations. Methodologies Appl., 12(1): 77-93.

Parsa, S. and O. Bushehrian, 2004. A framework to investigate and evaluate genetic clustering algorithms for automatic modularization of software systems. Lect. Notes Comput. Sc., 3037: 699-702, DOI: 10.1007/978-3-540-24687-9_106.

Raj Kiran, N. and V. Ravi, 2007. Software reliability prediction by soft computing technique. J. Syst. Software, 81(4): 132-140.

Rotaru, O.P., M. Dobre and M. Petrescu, 2005. Reusability metrics for software components, Proceedings of the 3rd ACS / IEEE International Conference of Computer Systems and Applications (AICCSA-05), Cairo, Egypt, pp: 24-29.

Sarkar, S., G.M. Rama and A.C. Kak, 2007. API-based and information-theoretic metrics for measuring the quality of software modularization. IEEE T. Software Eng., 33(1): 14-32.

Sharma, A., R. Kumar and P.S. Grover, 2008. Empirical evaluation of components for software components. Int. J. Softw. Eng. Know., 18(5): 519-330.

Shrddha, S., N.W. Nerurkar and A. Sharma, 2010. A soft computing based approach to estimate reusability of software components. ACM Sigsoft, 35(4): 1-5.

Sindre, G., R. Convadi and E.A. Karlsson, 1993. The REBOOT approach to software reuse. J. Syst. Software, 30(3): 201-215.

Szyperski, C., D. Gruntz and S. Murer, 2004. Component Software: Beyond Object Oriented Programming. Addison-Wesley Professional, London, Boston, MA.