International Journal of Computer Applications (0975 – 8887)
Volume 106 – No.5, November 2014

Neuro Fuzzy based Approach to Predict Component's Reusability

Shalini Goel
Assistant Professor
MIET, Meerut

Arun Sharma, Ph.D.
Professor and Head
KIET, Ghaziabad

ABSTRACT

The current scenario of open source development and outsourcing industry heavily depends upon the reusability of software components for achieving consistency in quality and cost optimization. Hence, the software developer needs excellent support in the assessment of the reusability levels of the software that they are trying to develop. Estimating Software Reusability has now become a topic of discussion. Reusability is measured in terms of cost, portability, understandability, interface complexity, coupling, interoperability, documentation, clarity etc. The combination of some factors were tested on different soft computing techniques like Fuzzy Logic and Neural Network and the corresponding average RMSE (Reusability Mean Square Error) was calculated to find out the reusability of those factors. In this paper it takes into account three different factors or variables of reusability of software components and and then propose a model for usability assessment using the Adaptive Neuro Fuzzy Inference System Approach (ANFIS).

General terms : Software Component, reusability

Keywords : Components based systems, FIS, Neuro Fuzzy, Reusability, RMSE, Software components, Sugeno

1. INTRODUCTION :

The current trend of the software industries and outsourcing industries heavily depends upon the reusability of software components to achieve increased quality software products with less time and less efforts. A lot of research has been done in the field of Software Reusability. Reusability is the quality of any software component to be used again with slight or no modification. Software reuse is the process of creating software systems from existing software assets rather than building them from scratch. [1**].** A reusable software component, or RSC, is a software entity intended for reuse. It may be the design, the code, or some other product of the software development [2]. A reusable software component can be considered to be a server that provides services for potential clients. A client, an application program, only needs to know the interface specifications of a potential server and does not need to know the details of how the server provides those services. A server can accept various kinds of requests from clients [3]. In component based systems development (CBSD), reusability of a software component is an important feature, which gives the determination to reuse the already existing developed component. It involves analyzing the reusability of software components by taking into consideration different factors every time. Reusability is the quality factor of software that qualifies it to be used again in another application, be it partially modified or completely modified. In other words, software reusability is a measure of the ease with which previously acquired concepts and objects can be used in new contexts. Component-based software engineering (CBSE) is a process that emphasizes the design and construction of computer-based systems using reusable software components. Three major ways to reuse software are (i) Use component in its original form (ii) Extend component functionality as needed for individual system (iii) Restrict component functionality as needed for independent system. [4] A lot of work has been done to estimate reusability of software components using fuzzy logic, artificial neural network and Neuro fuzzy also. In this paper the section 1 provides the introduction to the reusability of component based system, in section 2 related work to predict reusability is discussed, section 3 describes the concepts and variables used, section 4 elaborates the proposed work, section 5 shows experimental setup and section 6 explains the results and section 7 elaborates conclusion and future scope.

2. RELATED WORK TO PREDICT REUSABILITY :

The Component based Software Development, or CBSE process, however, must be characterized in a manner that not only identifies candidate components but also qualifies each component's interface, adapts components to remove architectural mismatches, assembles components into a selected architectural style, and updates components as requirements for the system change. [2].. An important factor in choosing the best component for reusability is to decide that which component is easily adaptable and it is based on the quality factors of the reusable software component. A lot of research has been done in the field of reusability. Anguswamy Reghu [5] presented two empirical studies – one for design with reuse process and other for design for reuse process. He gave some reuse design principles. In the method of design for reuse, total 107 subjects participated in the study. All the subjects were asked to implement s-stemming algorithm and to build a one use software component using the algorithm. Later they were asked to convert one use component to a reusable component using the platform of Java (Source line of code). Then reusable components were compared in terms of parameters like size (SLOC), effort (time), productivity (SLOC/hr). In the method of design with reuse some 25 components were selected randomly for this study. 34 subjects were allotted 5 components. A total of 170 (34*5) cases were analyzed in this study. The analysis was that, the larger the component the more difficult it was to reuse it.. Kumar et al. [6] formulates Neuro fuzzy approach to predict the reusability of software components. Neuro fuzzy yielded better accuracy as compared to standalone neural and fuzzy approach. Four different factors were taken on which Neuro fuzzy technique was applied to estimate the reusability of software components. Factors taken by them were customizability, interface complexity, understandability, and portability. They used the data of [11], its Reusability Mean Square Error (RMSE) was 0.1852 using artificial neural network approach but when ANFIS approach was used on the same data of [11], RMSE approach was 0.1695 which showed Neuro fuzzy approach provides better results. Ravichandaran et al. [7] emphasized on CBSE approach to select the appropriate reusable components and then combining them with well defined software architecture. COTS components or reusable components if properly selected reduce the maintainance and development cost, risk and completion time for product. It also delivers high quality software product.

They selected 14 reusable components and derive Fuzzy Weighted Relational Coefficient (FWRC) matrix between the components and CBS development to increase the accuracy of model. Components selected were reliability, stability, portability, consistency, completeness, interface and structural complexity, understandability, security, usability, accuracy, compatibility, performance, serviceability and customizability. ANFIS approach was proved to be the best to select reusable components. This paper also use some concepts of [10]. Anguswamy and Frakes [8] provide a list of some reuse design principles for CBSE. And those design principles were made independent from programming languages, domains and programming paradigms. 19 reuse design principles were proposed by them which were not specific to any particular project. Singh and Chana [9] focused on agile development methodologies to estimate the reusability of software components. An overview of extreme programming (XP) which is an agile software development process, was provided. They proposed a model to classify, store, search and retrieve the components using pattern matching algorithm. Some different storage and retrieval methods were also proposed to classify the components in software library where information retrieval methods depends on textual analysis of software assets and descriptive methods depends on textual description of software assets. Operational semantic methods depend on operational semantics of software assets. Topological methods identify library asset. Structural method retrieves programming pattern rather than executable code. So this approach helps making searching faster based on classification of components and also introduced reusability in agile software development. Singh and Toora [10] applied Neuro fuzzy technique on a case study which they take from a reputed journal. The case study is concerned with the reusability of software components. The reusable components or attributes were Coupling, Complexity, Volume regularity and Reuse frequency. They proved that Neuro fuzzy model yields less percentage average error as compared to standalone fuzzy logic and neural network. It also produces greater accuracy levels for software reusability as compared to fuzzy logic. Sharma et al. [11] takes 4 factors - Customizability, Interface Complexity, Portability and Understandability and estimate the reusability using Artificial Neural Network. Root Mean Square Error (RMSE) in this case was found to be 0.1852. Frakes and Kang [12] summarize reuse research, discussed unsolved problems and major contributions in research. Domain engineering was presented as a key idea to software reuse and the purpose was to improve the quality of products and services that a company provides. Problems of scalability, Safety, Reliability and Sustainability were addressed the main issues of reusable components. Selby [13] analyzed the factors that characterize successful software reuse in large scale system. His research approach was to evaluate software reuse by mining software repositories from NASA Software development environment that actively reuse software. This software achieves average reuse of 32% per project by following principles of reuse based software development. He examined the repositories for 25 software system ranging from 3000 to 112000 sources lines. He also analyzed 4 class of software modules based on the degree of modification required for reuse. Classes of software modules identified were - modules without revision, modules with slight revision (< 25), modules with major revision (>25) and newly developed modules. He applied non parametric statistical models to compare various development variables across 2954 software modules in the system. He also identified 2 categories of factors that characterize successful reuse in large scale system. Factors were module design factors, module implementation factors. He also evaluated the fault rates of various reused modifies and newly developed modules. He proved that earlier hypothesis of that there is no difference in the characteristics of software modules reused without revision, slight revision, major revision and newly developed. He proves that modules reused without revision had the fewest faults / source lines and lowest fault correction effort. And the modules reused with major revision had the highest fault correction effort and highest fault isolation effort, most changes / source lines and highest change correction effort. Morisio et al. [14] identified some key features for companywide reuse programs. Factors were derived from a survey of total 24 projects from 19 companies from 1994 to 1997. Data source for survey was a questionnaire with individual projects. Out of 24 projects, 9 projects were found to be failed by their managers themselves. The reason for failure were not considering human factors, reuse specific process and not modifying non reuse process and the most important shown was lack of top management's commitment. Houhamdi and Ghoul [15] approach was to show that RDF (Reuse Descriptive Formalism) can be used to represent and reuse some more abstract or complex entities like defects, process and projects also other than only as a tool to help increasing the reusability of software components at the function or subroutine level (code level). Attributes can be added in RDF with their values described and are used to refine the classification of selected instances and the attributes need not to be used in all instances but only those instances are modified for which new attribute is important. They also added an attribute named as ‘exception’. Poulin [16] established a taxonomy of approaches to reusability metrics based on two methods – empirical and qualitative. Empirical method depends on objective data whereas qualitative method depends on subjective value. He emphasized in Domain attributes to be considered in the reusability measuring metrics. Calderia and Basili [17] provides a statistical study stating that basic attributes of reusability depends on certain parameters like performance, testability and ease of modification etc but these attributes cannot be directly measured. They proposed 4 reusability measures with their range based on Mc Cabe and Halstead metrics. The measures were – Halstead program value measures cost and quality, Mc Cabe cyclomatic complexity must also balance cost and quality, Regularity measures readability, and Reuse frequency measures usefulness of modules. These 4 metrics were calculated for software in 9 example system. Prieto Diaz and Freeman [18] identified 5 program attributes and associated metrics to evaluate the reusability. Their process encouraged white box reuse and finding candidate reusable modules and then deciding the modules which the programmer can easily modify and adapt. They proposed 4 metrics which were module oriented and a fifth metric to modify the first four. The metrics were – program size, program structure, program documentation, programming language and reuse experience.

3. CONCEPTS AND VARIABLE USED

To develop the process to predict the software component’s reusability, the model needs to consider some dependent variables which can be given as an input to the developed method. The following dependent factors are considered in the proposed ANFIS approach.

3.1 Coupling

Coupling is a measure of interconnection among modules in a software structure. Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface. In software design, the lowest possible coupling is strived. Simple connectivity among modules results in software that is easier to understand and less prone to a "ripple effect", caused when errors occur at one location and propagate through a system. Here we refer Weighted Transitive Coupling (WT_{Coup}) metric [19] for components to estimate the coupling between various software components. This is readily achieved by summing all the couplings of all class pairs and dividing by the total number of such pairs. The weighted transitive coupling(WT_{Coup}) of a system is thus defined.

$$WT_{coup} = \frac{\sum_{i,j=1}^{m} Coup(i,j)}{m^2 - m}$$

where *m* is the number of classes in the system.

3.2 Complexity

Components should have well defined interfaces with less complexity as much as possible for high reusability. The high complexities of the interfaces will results to complex reuse and hence high efforts to change and understand the components. There are many traditional software complexity metrics, object oriented software complexity metrics, component complexity metrics. Here we are referring Sharma et al [20] interface complexity metric for the proposed work.

3.3 Portability

Portability is the ability of a component to be transferred from one environment to another with little modification, if required. For better reusability, component should be easily and quickly portable to specified new environments if and when necessary, with minimized porting efforts and schedules. 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.

4. PROPOSED ANFIS APPROACH

In the current research a new approach to predict component reusability. The Fuzzy Inference System (FIS) and Artificial Neural Network (ANN) have already been used by many researchers using different factors of reusability everytime. It is proven in many proposed techniques that neuro-fuzzy give the better results as compare to standalone FIS or ANN because it uses the power of rules decision of FIS and adaptive nature of ANN in a single system together.

Let C_m, P_o, C_p be the values of Complexity, Portability and Coupling respectively then

$R_{cn} = f_{cn}(C_m, P_o, C_p)$

where R_{cn} is the reusability of components and f_{cn} is implemented using ANFIS in MATLAB taking C_m, P_o, C_p as input dependent variables.

5. EXPERIMENTAL SETUP

In the proposed approach the neuro-fuzzy system is build, trained and tested in MATLAB:

- Inputs are given as crisp values of parameters
- Membership functions (low, medium, high) for each variable is determined.
- Membership functions (very low, low, medium, high and very high) for output function is determined.
- Based on the input parameters and membership functions fuzzy rules are generated.
- Based on Fuzzy rules, the input data i.e. training and testing data for ANFIS editor is evaluated.
- Using this training and testing data in ANFIS editor training and testing error is evaluated.
- Finally the training and testing error are compared and the results are viewed.

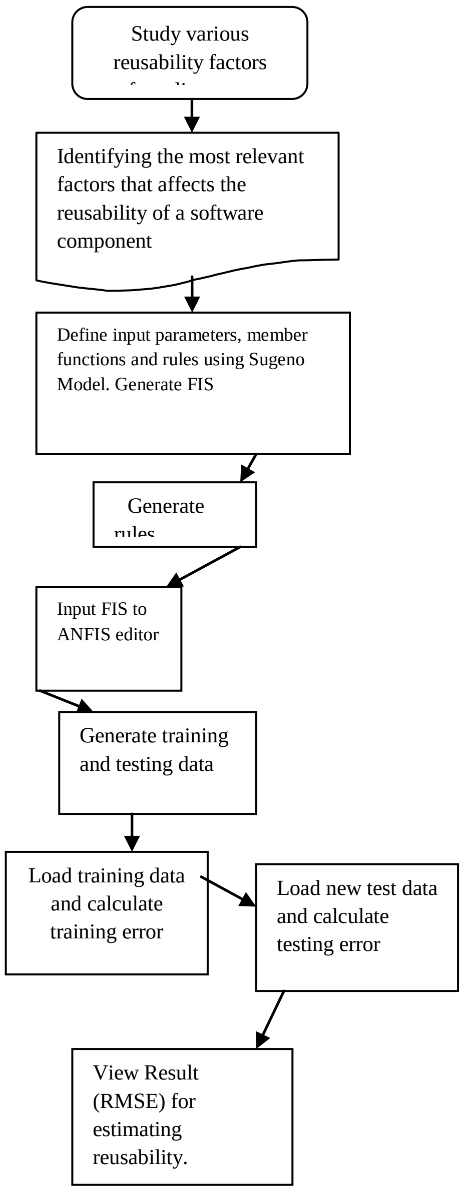
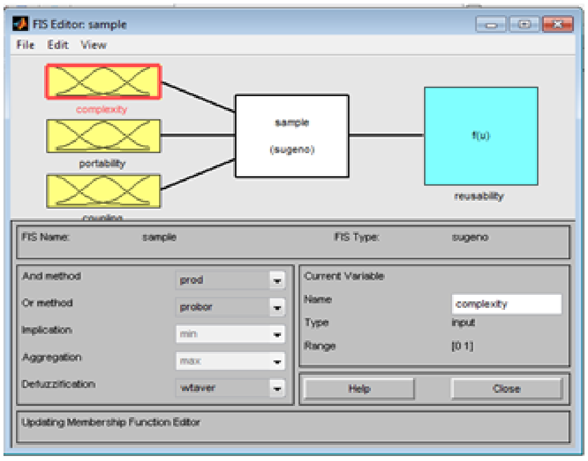

Study various reusability factors
Identifying the most relevant factors that affects the reusability of a software component
Define input parameters, member functions and rules using Sugeno Model. Generate FIS
Generate rules
Input FIS to ANFIS editor
Generate training and testing data
Load training data and calculate training error
Load new test data and calculate testing error
View Result (RMSE) for estimating reusability.

Fig 1: The Proposed approach

In the proposed model for reusability, the input variables which have been taken, each consisting of three different fuzzy linguistic sets (low, medium, high) using Triangular Membership functions (TMF) and the fuzzification process is applied on these sets by taking the normalized value for Cm. Po, Cp between [0 1] and for defuzzification, so as to produce a crisp value, five membership functions range is defined between [0 1] as (very low, low, medium high, very high) for reusability index. The given ANFIS system based on Expert knowledge contains 27 rules, 3 inputs and one single output for reusability. The entire data set of reusability level is 27 samples. They are referred to as training data and testing data. Upon training, the ANFIS shows the training error which reflects that how good the mapping function is working. To validate the model, further applied the testing data to see that how the ANFIS behaves for known data. ANFIS maps the function onto the testing data as per the training. Having created the data, sets the next step i.e. to train the network. This means a new FIS is created to fit the data into membership functions using the grid partitioning method as it generates a single-output Sugeno-type FIS. The ANFIS automatically selects the membership function and also generates the new FIS. 64% of the records were used as the training data and 36 % records for testing the data

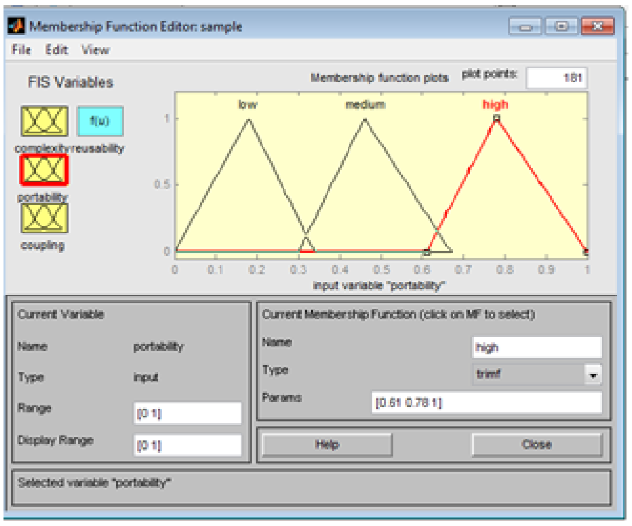
Table 1 : Experimental Setup Summary

Tool	**Fuzzy logic tool in MATLAB**
Fuzzy Inference System	Sugeno
Membership functions	Triangular (low, Medium, High)
Inputs	Complexity, Portability, Coupling
Rules	27
Editor	ANFIS
Training data	248 records
Testing data	90 records
Epochs	100
Output	Reusability



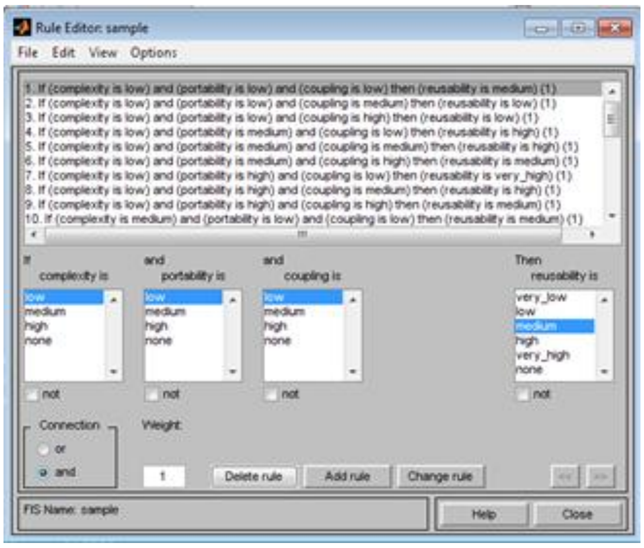
FIS Name: sample
FIS Type: sugeno
And method: prod
Or method: probor
Implication: min
Aggregation: max
Defuzzification: wtaver
Current Variable
Name: complexity
Type: input
Range: [0 1]
Updating Membership Function Editor

Fig 2: Sugeno type FIS for reusability



Current Variable
Name: portability
Type: input
Range: [0 1]
Display Range: [0 1]
Current Membership Function (click on MF to select)
Name: high
Type: trimf
Params: [0.61 0.78 1]
Selected variable "portability"

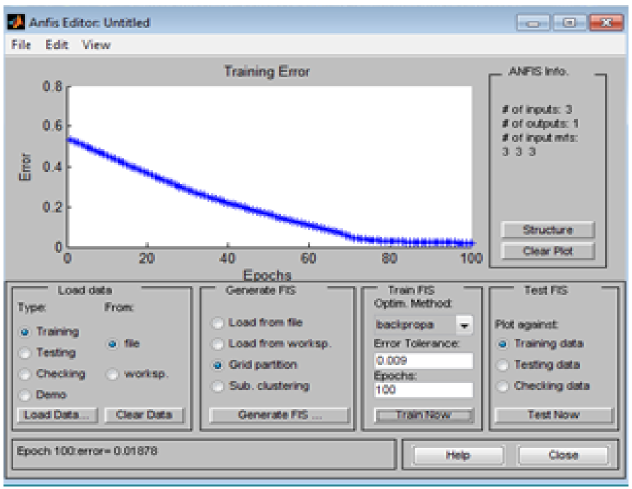
Fig 3: Membership function for Portability input variable



1. If (complexity is low) and (portability is low) and (coupling is low) then (reusability is medium) (1)
2. If (complexity is low) and (portability is low) and (coupling is medium) then (reusability is low) (1)
3. If (complexity is low) and (portability is low) and (coupling is high) then (reusability is low) (1)
4. If (complexity is low) and (portability is medium) and (coupling is low) then (reusability is high) (1)
5. If (complexity is low) and (portability is medium) and (coupling is medium) then (reusability is high) (1)
6. If (complexity is low) and (portability is medium) and (coupling is high) then (reusability is medium) (1)
7. If (complexity is low) and (portability is high) and (coupling is low) then (reusability is very_high) (1)
8. If (complexity is low) and (portability is high) and (coupling is medium) then (reusability is high) (1)
9. If (complexity is low) and (portability is high) and (coupling is high) then (reusability is medium) (1)
10. If (complexity is medium) and (portability is low) and (coupling is low) then (reusability is medium) (1)

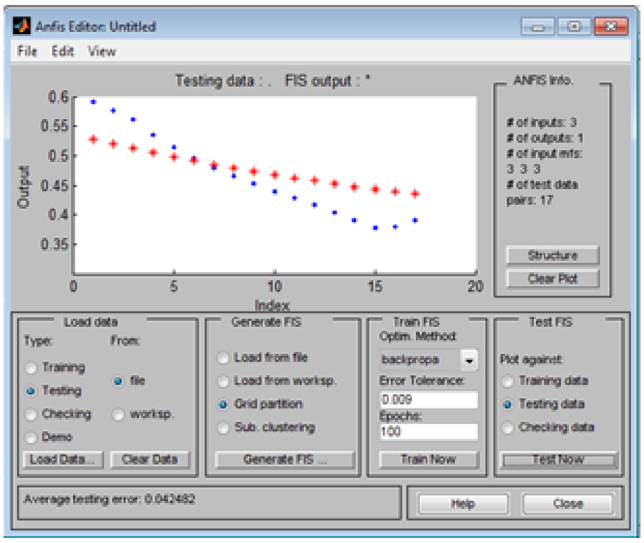
complexity is: low, medium, high, none
portability is: low, medium, high, none
coupling is: low, medium, high, none
reusability is: very_low, low, medium, high, very_high, none
Connection: or [✓ and]
Weight: 1
FIS Name: sample

Fig 4: Fuzzy rules in MATLAB



# of inputs: 3
# of outputs: 1
# of input mfs:
3 3 3
Type: [✓ Training] Testing Checking Demo
From: [✓ file] worksp.
Load from file / Load from worksp. / [✓ Grid partition] / Sub. clustering
Optim. Method: backpropa
Error Tolerance: 0.009
Epochs: 100
Plot against: [✓ Training data] Testing data Checking data
Epoch 100:error= 0.01878

Fig 5: Average Training error



ANFIS Info.
# of inputs: 3
# of outputs: 1
# of input mfs:
3 3 3
# of test data
pairs: 17

Load data — Type: Training, [✓ Testing], Checking, Demo; From: [✓ file], worksp.

Generate FIS — Load from file, Load from worksp., [✓ Grid partition], Sub. clustering

Train FIS — Optim. Method: backpropa; Error Tolerance: 0.009; Epochs: 100

Test FIS — Plot against: Training data, [✓ Testing data], Checking data

Average testing error: 0.042482

Fig 6: Average Testing error

6. RESULTS

The average training error is 0.01878 given the number of epochs is 100 using Backpropagation as optimization method. The average testing error after validation of the training data comes out to be 0.042482 given the error tolerance = 0.009 and the number of epochs to be 100 and using Backpropagation as optimization method.From the results, it is found that ANFIS model is appropriate for assessing the reusability of software components. The average testing error after validation of the training data comes out to be 0.42482 given the error tolerance = 0.009 and the number of epochs to be 100 and using back propagation as optimization method. This technique was able to reduce the error rate considerably.

7. CONCLUSION AND FUTURE SCOPE

As everyone knows that the reusability of any software component cannot be predicted without taking into consideration the different reusability factors of the software component and their interrelationship with one another. The need for reusability comes out from the fact that software systems often follow similar patterns; so as to avoid reinventing code and components to problems that have been encountered before. Reusability has an influence on all other aspects of software quality. Software reusability improves productivity and quality of software development. Reusability is also a5 very important quality factor for the selection criteria of software component. Identifying the critical factors and developing a rule base system for evaluating the reusability of the software components is need of today.

As the future scope is concerned this work finds its application in any domain.

- Validation can be improved by taking into consideration training data from real time projects.
- New factors can be added like understandability, cohesion, clarity, generality etc and the cumulative effect of those factors can be seen on the future predictions.
- Different techniques can be used other than ANFIS to predict the reusability like Support Vector Machines (SVM).
- Moreover, a much better Generalized approach is expected if the real time data is considered over the past many years, trained in the order of their occurrence.

8. REFERENCES

[1] Babu G.N.K. Suresh, Dr. Srivatsa .S.K., “Analysis and measure of software reusability”, Proceedings of International Journal of Reviews in Computing, 2009 IJRIC,

[2] Software Engineering, A Practitioner approach, Roger S Pressman, Mc Graw Hill

[3] Deng-Jyi Chen, Chorng-Shiuh Koong, Wu-Chi Chen, Shih-Kun Huang and N. W. P Van Diepen, “Integration of Reusable Software Components and Frameworks Into a Visual Software Construction Approach”, Journal of Information Science and Engineering 16, 863-884 (2000)

[4] Jalender B., Dr Govardhan A., Dr Premchand P, “Designing Code Level Reusable Software Components”, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012

[5] Anguswamy R,” A Study of Factors Affecting the Design and Use of Reusable Components”, International Doctoral Symposium on Empirical Software Engineering (IDoESE'12), July.31, 2013, Lund, Sweden

[6] Sharma A, kumar R, kumar V, “Applying Neuro-fuzzy Approach to build the Reusability Assessment Framework across Software Component Releases - An Empirical Evaluation”, IJCA Vol – 70, no-15, may 2013

[7] Ravichandran K.S., Suresh P. and Sekr, K.R. “ANFIS Approach for Optimal Selection of Reusable Components”, Maxwell Scientific Organization, Research Journal of Applied Sciences, Engineering and Technology 4(24): 5304-5312

[8] Anguswamy R, Frakes W B., “ Reuse Design Principles” International Journal of Software Engineering and Knowledge Engineering, IJSEKE, 2012

[9] Singh S , Chana I, “ Enabling Reusability in Agile Software Development”, International Journal of Computer Applications (0975 – 8887) Volume 50 – No.13, July 2012

[10] Singh H, Toora V K, “Neuro Fuzzy Logic Model for Component Based Software Engineering”, International Journal of Engineering, Issue July 2011, Vol. 1

[11] Sharma A, Kumar R, Grover P. S., “ Reusability Assessment for Software Components – a Neural Network Based Approach”, ACM SIGSOFT Software engineering notes, vol 34, issue 2, March pp. – 1- 6, 2009

[12] Frakes W. B. and Kang K. C., "Software reuse research: status and future," IEEE Transactions on Software Engineering, vol. 31, pp. 529-536, 2005.

[13] Selby R. W., "Enabling reuse-based software development of large-scale systems", IEEE Transactions on Software Engineering, vol. 31, pp. 495-510, 2005.

[14] M. Morisio, et al., "Success and Failure Factors in Software Reuse," IEEE Transactions on Software Engineering, vol. 28, pp. 340-357, 2002.

[15] Houhamdi Z, Ghoul S, “Classifying Software For Reusability”, www.webreview.dz/IMG/pdf/6-houhamdi.pdf, November 2001, pp. 41-47

[16] Jeffrey S. Poulin, “Measuring Software Reusability”, Proceedings of 3rd International conference on software reuse – Rio De Janerio, Brazil 1-4 November 1994

[17] Calderia G, Basili V R, “Identifying and qualifying reusable software components”, IEEE Software, vol. 24, no 2, Feb 1991, pp 61-76

[18] Diaz P, Freeman Ruben and Peter, “Classifying software for reusability”, Vol. 4, no – 1, January 1987, pp 6-16

[19] Gui Gui, Paul D. Scott,” Ranking reusability of software components using coupling metrics” Journal of Systems and Software 01/2007; DOI:10.1016/ j.jss.2006.09.048

[20] Sharma, A., Kumar, R., Grover, P. S., 2008. Empirical Evaluation of Complexity for Software Components, International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Vol. 18, Issue 5, pp: 519-530.

[21] Menzies Tim and Stefano J S. D, “More Success and Failure Factors in Software Reuse”, IEEE Transactions On Software Engineering, Vol. 29, No. 5, May 2003

[22] Saini J K, Sharma A, Dr. Sandhu P S., “Software Reusability Prediction using Density Based Clustering”,http://psrcentre.org/images/extraimages/380.pdf.