

A Reusability Evaluation Model for OO-Based Software Components

Parvinder S. Sandhu, and Hardeep Singh

Abstract—The requirement to improve software productivity has promoted the research on software metric technology. There are metrics for identifying the quality of reusable components but the function that makes use of these metrics to find reusability of software components is still not clear. These metrics if identified in the design phase or even in the coding phase can help us to reduce the rework by improving quality of reuse of the component and hence improve the productivity due to probabilistic increase in the reuse level. CK metric suit is most widely used metrics for the object-oriented (OO) software; we critically analyzed the CK metrics, tried to remove the inconsistencies and devised the framework of metrics to obtain the structural analysis of OO-based software components. Neural network can learn new relationships with new input data and can be used to refine fuzzy rules to create fuzzy adaptive system. Hence, Neuro-fuzzy inference engine can be used to evaluate the reusability of OO-based component using its structural attributes as inputs. In this paper, an algorithm has been proposed in which the inputs can be given to Neuro-fuzzy system in form of tuned WMC, DIT, NOC, CBO, LCOM values of the OO software component and output can be obtained in terms of reusability. The developed reusability model has produced high precision results as expected by the human experts.

Keywords—CK-Metric, ID3, Neuro-fuzzy, Reusability.

I. INTRODUCTION

THE aim of Object Oriented (OO) Metrics is to predict the quality of the object oriented software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. The requirement nowadays is to explore the relation of the reusability attributes with the metrics and to find how these metrics collectively determine the reusability of the software component. To achieve both the quality and productivity objectives, it is always recommended to go for the software reuse that not only saves the time taken to develop the product from scratch but also delivers the almost error free code, as the code is already tested many times during its earlier reuse.

The manuscript was submitted for review on June 2, 2006.

Parvinder S. Sandhu is Assistant Professor with Computer Science & Engineering Department, Guru Nanak Dev Engineering College, Ludhiana(Punjab)-141006 India (phone: +91-98555-32004; Fax: +91161-2490339; Email: parvinder.sandhu@gmail.com, parvsandhu@yahoo.co.in)

Hardeep Singh is Professor and Head with Computer Science & Engineering Department, Guru Nanak Dev University, Amritsar (Punjab) India.

A great deal of research over the past several years has been devoted to the development of methodologies to create reusable software components and component libraries, where there is an additional cost involved to create a reusable component from scratch. That additional cost could be avoided by identifying and extracting reusable components from the already developed large inventory of existing systems. But the issue of how to identify good reusable components from existing systems has remained relatively unexplored. Our approach, for identification and evaluation of reusable software, is based on software models and metrics. As the exact relationship between the attributes of the reusability is difficult to establish, a Neural Network approach could serve as an economical, automatic tool to generate reusability ranking of software [1] [2] by formulating the relationship based on its training. When one designs with Neural Networks alone, the network is a black box that needs to be defined; this is a highly compute-intensive process. One must develop a good sense, after extensive experimentation and practice, of the complexity of the network and the learning algorithm to be used. Neural nets and fuzzy systems, although very different, have close relationship: they work with impression in a space that is not defined by crisp, deterministic boundaries [3]. Neural network can be used to define fuzzy rules for the fuzzy inference system. A neural network is good at discovering relationships and pattern in the data, so neural network can be used to preprocess data in the fuzzy system. Furthermore, neural network that can learn new relationships with new input data can be used to refine fuzzy rules to create fuzzy adaptive system. With the objective of taking advantage of the features of the both [4], we used Neuro-Fuzzy approach to economically determine reusability of OO-based software components in existing systems as well as the reusable components that are in the design phase. Inputs to Neuro-fuzzy system, are provided in form of tuned WMC, DIT, NOC, CBO, LCOM values of the OO-based software component and output is be obtained in terms of reusability.

II. BACKGROUND

It was Selby [5], who tried to identify a number of characteristics of those components, from existing systems, that are been reused at NASA laboratory and reported that the developers there has achieved a 32 percent reusability index. Dunn and Knight in 1991[6] also experimented and reported the usefulness of reusable code scavenging. Chen, Nishimoto and Ramamoorthy briefly discuss the idea of subsystem extraction by using code information stored in a relational

database [7]. They also describe a tool called the C Information Abstraction System to support this process. Esteva and Reynolds [8] describe the use of Inductive Learning techniques based on software metrics used to identify reusable modules. Their system was able to learn to recognize reusable components. Caldiera and Basili [9] describe a tool, called Care, that helps identify reusable components according to a set of “reusability attributes” based on software metrics. These attributes include measurement of how useful the component is in the problem domain, how much it would cost to reuse it, and its quality. The idea behind Care is that it will do the initial identification of the components that have strong reusability characteristics, and then a domain expert will do a further examination of these components to determine their appropriateness to the domain, and package them to reuse. Mayobre [10] describes how these techniques can be extended and used to help in identifying data communication components at Hewlett-Packard.

Arnold [11][12] mentions a number of heuristics that can be used for locating reusable components in the Ada source code. The heuristics count the number of references to a particular procedure, identifying the loosely coupled modules and identifying modules that carry high cohesion.

Selby’s recent experimental study [13] has identified two categories of factors that characterize successful reuse-based software development of large-scale systems: module design factors and module implementation factors. The module design factors that characterize module reuse without revision were: few calls to other system modules (i.e. low coupling), many calls to utility functions (i.e. high cohesion), few input-output parameters, few reads and writes, and many comments. The module implementation factors that characterize module reuse without revision were small size in source lines and have many assignment statements (i.e. low Cyclometric complexity). The modules reused without revision had the fewest faults per source line, and lowest fault correction effort. The modules reused with major revision had the highest fault correction effort and highest fault isolation effort as well as the most changes per source line and highest change correction effort.

III. OO-BASED REUSABILITY MODEL

Reusability evaluation Model for OO-Based Software Components can be framed using following steps:

- Selection and refinement of metrics targeting the quality of OO-based software system and perform parsing of the software system to generate the Meta information related to that Software.
- Neuro-Fuzzy system, which is already initialized and trained using the training data, will get the meta information from the earlier stage and determines the reusability value of the software components. Considering the reusability value, the component can be extracted and put into the Reusable Software Reservoir for future reuse.

A. OO-Based Metric Framework

As CK metric suit is able to target all the essential attributes of OO-based software, as mentioned by Selby [13] in his latest findings, so we analyzed, refined and used following metrics of CK metric suit to explore different structural dimensions of a class.

1) *Weighted methods per class (WMC)*: According to this metric if a Class C, has n methods and $c_1, c_2 \dots c_n$ be the complexity of the methods, then $WMC(C) = c_1 + c_2 + \dots + c_n$. McCabe’s complexity metric is chosen for calculating the complexity values of the methods of a class. The value is normalized so that nominal complexity for a method takes on a value of 1.0. If all method complexities are considered to be unity, then $WMC = n$ i.e. the number of methods existing in that class [14][15].

We have used “tuned WMC” (TWMC) measure as input to the neuro-fuzzy inference engine by restricting the WMC value in between 0 and 1 with help of sigmoidal function as shown in (1).

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (1)$$

Where $a=10$ and $c=0.5$.

2) *Depth of inheritance tree (DIT)*: According to this metric, Depth of inheritance of a class is “the maximum length from the node to the root of the tree”. More is the depth of the inheritance tree greater the reusability of the class corresponding to the root of that tree as the class properties are shared by more derived classes under that class. So there too much depth dilutes the abstraction. So there is a need to set the minimum & maximum DIT value for a class as an contribution towards the reusability [14][15].

The definition of DIT is ambiguous when multiple inheritance and multiple roots are present as the alternative length of the path is not being considered in case of multiple inheritance. If we add all the ancestor classes coming in common path to the ancestor classes coming in alternative paths then that will be the true representation of the theoretical basis of the DIT metric [16].

We have used “lack of tuned degree of inheritance” (LTDIT) measure as input to the Neuro-fuzzy inference system, in order to restrict the input value between 0 and 1.

3) *Number of Children (NOC)*: According to this metric Number of children (NOC) of a class is the number of immediate sub-classes subordinated to a class in the class hierarchy. So greater is the value of NOC greater will be the reusability of the parent class. Hence there should be some minimum value of NOC for a parent class for its reusability [14][15].

Theoretical basis of NOC metric relates to the notion of scope of properties. It is a measure of how many sub-classes are going to inherit the methods of the parent class [14]. The

definition of NOC metric gives the distorted view of the system as it counts only the immediate sub-classes instead of all the descendants of the class. So according to [16] the NOC value of a class, say class i , should reflect all the subclasses that share the properties of that class as shown in (2).

$$NOC(i) = N + \sum_i^{AllSubclasses} NOC(i) \quad (2)$$

Where N is the total number of immediate subclasses of class i . In order to restrict the input value between 0 and 1, we have used “lack of tuned Number of Children” (LTNOC) measure as input to the Neuro-fuzzy inference system.

4) *Coupling Between Object Classes (CBO)*: According to this metric “Coupling Between Object Classes” (CBO) for a class is a count of the number of other classes to which it is coupled. Theoretical basis of CBO relates to the notion that an object is coupled to another object if one of them acts on the other, i.e. methods of one object use methods or instance variables of another. Here we are restricting the unidirectional use of methods or instance variables of another object by the object of the class whose reusability is to be measured. As Coupling between Object classes increases, reusability decreases and it becomes harder to modify and test the software system. So there is the need to set some maximum value of coupling level for its reusability. If the value of CBO for a class is beyond that maximum value then the class is said to be non-reusable[14][15].

In order to restrict the input value between 0 and 1, we have used “lack of CBO” (LCBO) measure as input to the Neuro-fuzzy inference system.

5) *Lack of Cohesion in Methods (LCOM)*: Consider a Class C_1 with n methods M_1, M_2, \dots, M_n . Let $\{I_j\}$ be set of instance variables used by method M_i . There are n such sets $\{I_1\}, \{I_2\}, \dots, \{I_n\}$. Let $P = \{ (I_i, I_j) \mid I_i \cap I_j = \emptyset \}$ and $Q = \{ (I_i, I_j) \mid I_i \cap I_j \neq \emptyset \}$. If all n sets $\{I_1\}, \{I_2\}, \dots, \{I_n\}$ are \emptyset then let $P = \emptyset$ [4]. Lack of Cohesion in Methods (LCOM) of a class can be defined as

$$LCOM = |P| - |Q|, \text{ if } |P| > |Q|$$

$$LCOM = 0 \text{ otherwise}$$

The high value of LCOM indicates that the methods in the class are not really related to each other and vice versa means less reusability otherwise low value of LCOM depicts high internal strength of the class which results into high reusability. So there should be some maximum value of LCOM after which class becomes non-reusable [14][15].

We have used “tuned LCOM” (TLCOM) measure as input to the neuro-fuzzy inference engine by restricting the LCOM value in between 0 and 1 with help of sigmoidal function as shown in (3).

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (3)$$

Where $a=4$ and $c=1.5$.

B. Neuro-Fuzzy System's Architecture

The fuzzy logic approach is beneficial for measuring the reusability of a software component as the conventional model based approaches are difficult to be implemented. Unfortunately, with the increase in the complexity of the problem being modeled and unavailability of the precise relationship among various constituents for measuring the reusability, has led to rely on another approach which is mostly known as neuro-fuzzy or fuzzy-neuro approach. It has the benefits of both neural networks and fuzzy logic. The neuro-fuzzy hybrid system combines the advantages of fuzzy logic system, which deal with explicit knowledge that can be explained and understood, and neural networks, which deal with implicit knowledge, which can be acquired by learning.

A fuzzy system can be considered to be a parameterized nonlinear map, called f , which can be expressed as in (4).

$$f(x) = \frac{\sum_{l=1}^m y^l \left(\prod_{i=1}^n \mu_{A_i^l}(x_i) \right)}{\sum_{l=1}^m \left(\prod_{i=1}^n \mu_{A_i^l}(x_i) \right)} \quad (4)$$

Where y^l is a place of output singleton if Mamdani reasoning is applied or a constant if Sugeno reasoning is applied. The membership function $\mu_{A_i^l}(x_i)$ corresponds to the input $x=[x_1, x_2, x_3, \dots, x_m]$ of the rule l . The “and” connective in the premise is carried out by a product and defuzzification by the center-of-gravity method [4]. Consider a Sugeno type of fuzzy system having the rule base

Rule1: If x is A_1 and y is B_1 , then $f_1 = p_1x + q_1y + r_1$

Rule2: If x is A_2 and y is B_2 , then $f_2 = p_2x + q_2y + r_2$

Let the membership functions of fuzzy sets $A_i, B_i, i=1,2$, be, μ_{A_i}, μ_{B_i} .

1) Evaluating the rule premises results values as shown in (5).

$$w_i = \mu_{A_i}(x) * \mu_{B_i}(y) \quad (5)$$

where $i = 1,2$ for the rule rules stated above.

2) Evaluating the implication and the rule consequences gives values as shown in (6)-(9).

$$f(x, y) = \frac{w_1(x, y)f_1(x, y) + w_2(x, y)f_2(x, y)}{w_1(x, y) + w_2(x, y)} \quad (6)$$

Or

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} \quad (7)$$

Let

$$\bar{w}_i = \frac{w_i}{w_1 + w_2} \quad (8)$$

then f can be written as

$$f = \bar{w}_1 f_1 + \bar{w}_2 f_2 \tag{9}$$

These all computations can be presented in a diagram form [4] as shown in the fig. 1(a) and 1(b).

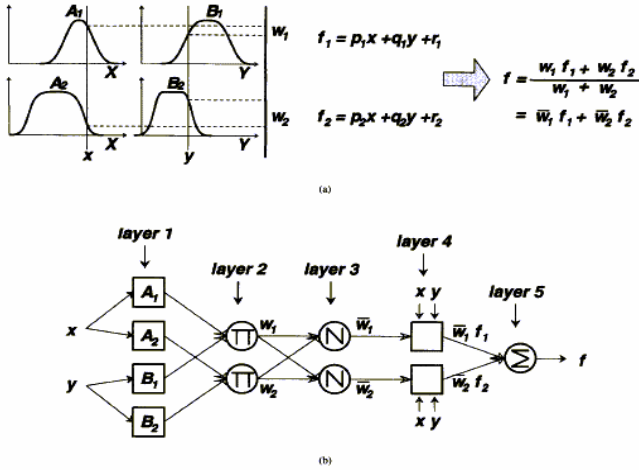


Fig. 1(a) A two-Input First-Order Sugeno Fuzzy Model with two rules (b). Equivalent Neuro-Fuzzy System

In the neuro-fuzzy inference system using a given input/output data set, we have constructed a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using a hybrid method consisting of back-propagation form of the steepest descent method for the parameters associated with the input membership functions, and least squares estimation for the parameters associated with the output membership functions. As a result, the training error decreases, at least locally, throughout the learning process. Therefore, the more the initial membership functions resemble the optimal ones, the easier it will be for the model parameter training to converge.

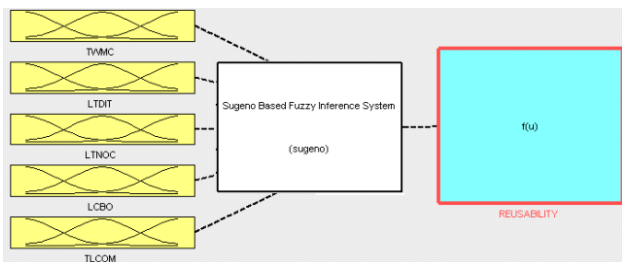


Fig. 2 Fuzzy Inference System with 5 inputs & 1 output

Before training, the initial rule section is done using ID3 decision tree generation algorithm. This allows fuzzy systems to learn from the data they are modeling in less number of iterations and with less training error. We are using the Sugeno type Fuzzy Inference System with five metric value Inputs and one output as shown in Fig. 2.

IV. IMPLEMENTATION

In the fuzzy Inference system Linguistic variables are then assigned to the input parameters based on their values. The assignment of the linguistic variables depends on the range of the input measurement.

Values to the linguistic variables of TWMC are assigned in terms of complexity of the software component. TWMC is assigned three linguistic variables LOW, MEDIUM and HIGH in the range of 0 to 1.

Values to the linguistic variables of LTDIT (Lack of tuned degree of Inheritance) are assigned in terms of level of inheritance for the software component under consideration. LTDIT is assigned three linguistic variables LOW, HIGH and MEDIUM in the range of 0 to 1.

Values to the linguistic variables of LTNOC (Lack of tuned Number of Children) are assigned in terms of number of children of the software class in the inheritance tree. Quality attribute LTNOC is assigned three linguistic variables LOW, MEDIUM and HIGH in the range of 0 to 1.

Values to the linguistic variables of LCBO (Lack of Coupling between Objects) are assigned in terms of coupling between objects of the software. LCBO is assigned three linguistic variables LOW, MEDIUM and HIGH in the range of 0 to 1.

Values to the linguistic variables of TLCOM (tuned Lack of Cohesion) are assigned in terms of Cohesion in methods of the class of object-oriented software TLCOM is assigned three linguistic variables LOW, MEDIUM and HIGH in the range of 0 to 1.

Values to the linguistic variables of Reusability are assigned in terms of “how reusable the software component is?” The output membership functions are only linear or constant for Sugeno-type fuzzy inference. Reusability is assigned six linguistic variables PERFECT, HIGH, MEDIUM, LOW, VERY-LOW and NIL as constants in the range of 0-1.

A network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map is shown in the fig. 3. The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure. The Error Tolerance is used to create a training stopping criterion, which is related to the error size. The training will stop after the training data error remains within this tolerance. This is set to 0 as we don't know how training error is going to behave.

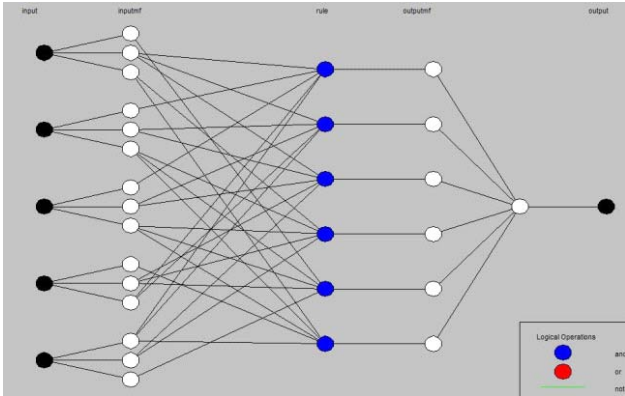


Fig. 3 Neural Network incorporating the Fuzzy inference system

The Training of the neuro-fuzzy system is performed using Training Data, as mentioned in appendix-I, for 1000 iterations and the training error reduces after each iteration as shown by the fig. 4 and stabilizes at the error value of 0.041969, so at this point the network is said to be converged.

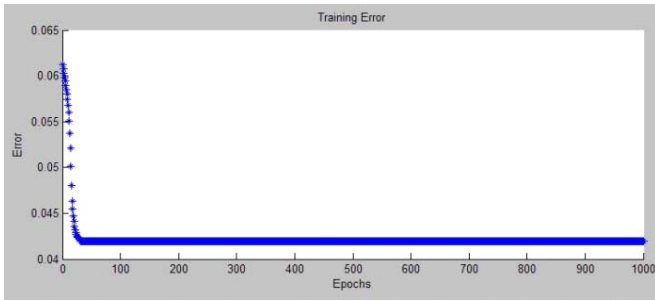


Fig. 4 Plot of Training error V/s Epochs.

V. CONCLUSION

During the testing phase, when the developed system is tested against the testing data, as mentioned in appendix-II, Average Testing error 0.039364 is obtained. The plot between the actual output and the expected output is shown in fig. 5. As the actual output produced by the “Reusability Evaluation Model” is close to the expected output, so the system can be recommended for Automatic identification potential reusable object oriented software components from the legacy systems and evaluating the quality of developed or developing reusable components for better productivity and quality.

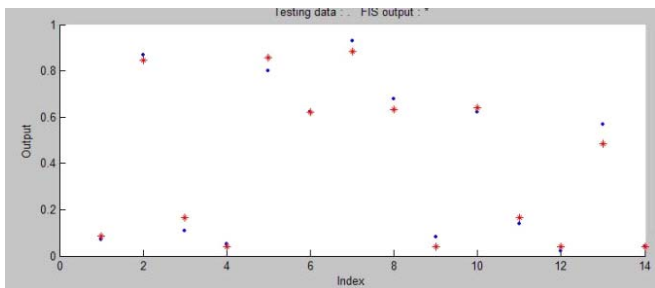


Fig. 5 Plot between the actual output and expected output

APPENDIX-I

Training data for Neuro-Fuzzy System is shown in Table I.

TABLE I
TRAINING DATA FOR NEURO- FUZZY SYSTEM

| TWMC | LTDIT | LTNOC | LCBO | TLCOM | REUSABILITY |
|------|-------|-------|------|-------|-------------|
| 0.27 | 0.12 | 0.15 | 0.81 | 0.24 | 0.93 |
| 0.36 | 0.29 | 0.31 | 0.81 | 0.14 | 0.72 |
| 0.45 | 0.53 | 0.38 | 0.51 | 0.63 | 0.59 |
| 0.96 | 0.82 | 0.79 | 0.53 | 0.48 | 0.37 |
| 0.18 | 0.83 | 0.78 | 0.35 | 0.93 | 0.15 |
| 0.83 | 0.91 | 0.79 | 0.73 | 0.21 | 0.07 |
| 0.35 | 0.23 | 0.18 | 0.78 | 0.13 | 0.87 |
| 0.58 | 0.36 | 0.24 | 0.84 | 0.17 | 0.68 |
| 0.63 | 0.46 | 0.29 | 0.37 | 0.58 | 0.54 |
| 0.85 | 0.79 | 0.85 | 0.62 | 0.39 | 0.34 |
| 0.13 | 0.74 | 0.87 | 0.41 | 0.84 | 0.11 |
| 0.88 | 0.79 | 0.92 | 0.22 | 0.14 | 0.05 |
| 0.42 | 0.17 | 0.23 | 0.84 | 0.21 | 0.82 |
| 0.67 | 0.43 | 0.36 | 0.79 | 0.20 | 0.65 |
| 0.32 | 0.35 | 0.43 | 0.52 | 0.49 | 0.47 |
| 0.77 | 0.83 | 0.93 | 0.72 | 0.68 | 0.28 |
| 0.23 | 0.94 | 0.79 | 0.38 | 0.91 | 0.17 |
| 0.77 | 0.83 | 0.79 | 0.18 | 0.23 | 0.06 |
| 0.48 | 0.20 | 0.19 | 0.75 | 0.18 | 0.80 |
| 0.43 | 0.56 | 0.44 | 0.68 | 0.23 | 0.62 |
| 0.57 | 0.42 | 0.28 | 0.61 | 0.33 | 0.38 |
| 0.88 | 0.79 | 0.82 | 0.39 | 0.45 | 0.32 |
| 0.19 | 0.81 | 0.93 | 0.67 | 0.77 | 0.19 |
| 0.92 | 0.94 | 0.86 | 0.12 | 0.11 | 0.03 |
| 0.57 | 0.14 | 0.20 | 0.81 | 0.12 | 0.85 |
| 0.38 | 0.39 | 0.28 | 0.81 | 0.19 | 0.70 |
| 0.49 | 0.63 | 0.68 | 0.47 | 0.58 | 0.45 |
| 0.79 | 0.85 | 0.78 | 0.48 | 0.53 | 0.36 |
| 0.0 | 0.78 | 0.89 | 0.49 | 0.84 | 0.13 |
| 0.86 | 0.79 | 0.93 | 0.20 | 0.19 | 0.01 |
| 0.68 | 0.23 | 0.13 | 0.93 | 0.24 | 0.93 |
| 0.69 | 0.45 | 0.67 | 0.64 | 0.20 | 0.68 |
| 0.68 | 0.51 | 0.36 | 0.32 | 0.43 | 0.48 |
| 0.91 | 0.93 | 0.80 | 0.62 | 0.37 | 0.25 |
| 0.12 | 0.90 | 0.81 | 0.73 | 0.80 | 0.24 |
| 0.94 | 0.88 | 0.84 | 0.13 | 0.20 | 0.08 |
| 0.72 | 0.19 | 0.22 | 0.89 | 0.16 | 0.84 |
| 0.72 | 0.64 | 0.59 | 0.93 | 0.16 | 0.62 |
| 0.71 | 0.39 | 0.52 | 0.57 | 0.68 | 0.40 |
| 0.83 | 0.81 | 0.92 | 0.28 | 0.41 | 0.36 |
| 0.0 | 0.83 | 0.91 | 0.39 | 0.93 | 0.14 |
| 0.78 | 0.80 | 0.77 | 0.19 | 0.15 | 0.02 |
| 0.74 | 0.24 | 0.23 | 0.92 | 0.21 | 0.81 |
| 0.74 | 0.58 | 0.42 | 0.86 | 0.23 | 0.60 |
| 0.43 | 0.42 | 0.39 | 0.46 | 0.71 | 0.57 |
| 0.78 | 0.90 | 0.83 | 0.54 | 0.63 | 0.28 |
| 0.21 | 0.95 | 0.78 | 0.45 | 0.82 | 0.18 |
| 0.81 | 0.76 | 0.89 | 0.21 | 0.17 | 0.04 |

APPENDIX-II

Testing data for Neuro-Fuzzy System is shown in TABLE II.

TABLE II
TESTING DATA FOR NEURO- FUZZY SYSTEM

| TWMC | LTDIT | LTNOC | LCBO | TLCOM | REUSAB- -ILITY |
|------|-------|-------|------|-------|-------------------|
| 0.83 | 0.97 | 0.79 | 0.63 | 0.21 | 0.07 |
| 0.32 | 0.29 | 0.18 | 0.78 | 0.13 | 0.87 |
| 0.13 | 0.80 | 0.87 | 0.43 | 0.84 | 0.11 |
| 0.85 | 0.79 | 0.92 | 0.22 | 0.14 | 0.05 |
| 0.45 | 0.20 | 0.19 | 0.75 | 0.12 | 0.80 |
| 0.40 | 0.56 | 0.44 | 0.68 | 0.23 | 0.62 |
| 0.62 | 0.23 | 0.13 | 0.93 | 0.24 | 0.93 |
| 0.66 | 0.45 | 0.67 | 0.64 | 0.20 | 0.68 |
| 0.91 | 0.88 | 0.88 | 0.13 | 0.20 | 0.08 |
| 0.69 | 0.64 | 0.59 | 0.93 | 0.19 | 0.62 |
| 0.0 | 0.83 | 0.89 | 0.39 | 0.93 | 0.14 |
| 0.71 | 0.80 | 0.77 | 0.19 | 0.14 | 0.02 |
| 0.41 | 0.42 | 0.40 | 0.46 | 0.71 | 0.57 |
| 0.87 | 0.76 | 0.89 | 0.21 | 0.23 | 0.04 |

- [16] Parvinder Singh and Hardeep Singh, "Critical Suggestive Evaluation of CK METRIC", Proc. of 9th Pacific Asia Conference on Information Technology (PACIS-2005), Bangkok, Thailand, July 7 – 10, 2005. (Paper is available online at <http://www.pacis-net.org/>)

ACKNOWLEDGMENT

The authors like to express their gratitude towards Dr. S. B. Singh (Principal, G.N.D.E.C., Ludhiana) and Dr. H. K. Grewal, HOD (CSE & IT), G.N.D.E.C, Ludhiana for provision of laboratory facilities.

REFERENCES

- [1] G. Boetticher and D. Eichmann, "A Neural Network Paradigm for Characterizing Reusable Software", Proc. of the Australian Conference on Software Metrics, 18-19 November 1993.
- [2] G. Boetticher, K. Srinivas, and D. Eichmann, "A Neural Net-based Approach to Software Metrics", Proc. of the 5th International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, 14-18 June 1993, pp 271-274.
- [3] S. V. Kartalopoulos, Understanding Neural Networks and Fuzzy Logic-Basic Concepts and Applications, IEEE Press, 1996, pp. 153-160.
- [4] J-S. R. Jang and C.T. Sun, "Neuro-fuzzy Modeling and Control," Proc. of IEEE, March 1995.
- [5] R. W. Selby, Empirically Analyzing Software Reuse in a Production Environment, Software Reuse: Emerging Technology, W. Tracz, ed, IEEE Computer Society Press, 1988.
- [6] M. F. Dunn and J. C. Knight, "Software reuse in Industrial setting : A Case Study," Proc. of the 13th International Conference on Software Engineering, Baltimore, MA, 1993.
- [7] Y. F. Chen, M. Y. Nishimoto and C. V. Ramamoorthy, "The C Information Abstraction System," IEEE Trans. on Software Engineering, Vol. 16, No. 3, March 1990.
- [8] J. C. Esteva and R. G. Reynolds, "Identifying Reusable Components using Induction," International Journal of Software Engineering and Knowledge Engineering, Vol. 1, No. 3 (1991) 271-292.
- [9] G. Caldiera and V. R. Basili, "Identifying and Qualifying Reusable Software Components," IEEE Computer, February 1991.
- [10] G. Mayobre, "Using Code Reusability Analysis to Identify Reusable Components from Software Related to an Application Domain," Proc. of the Fourth Workshop on Software Reuse, Reston, VA, November, 1991.
- [11] R.S. Arnold, Heuristics for Salvaging Reusable Parts From Ada Code, SPC Technical Report, ADA_REUSE_HEURISTICS-90011-N, March 1990.
- [12] R.S. Arnold, Salvaging Reusable Parts From Ada Code: A Progress Report, SPC Technical Report, SALVAGE_ADA_PARTS_PR-90048-N, September 1990.
- [13] Richard W. Selby, "Enabling Reuse-Based Software Development of Large-Scale Systems", IEEE Trans. on Software Engineering, Vol. 31, No. 6, June 2005 pp. 495-510.
- [14] S.R. Chidamber and C.F. Kemerer, "A Metric Suite for Object Oriented Design", IEEE Trans. on Software Engineering, Vol. 20, pp. 476-493, 1994.
- [15] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design," Proc. Conf. Object Oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, pp. 197-211, 1991.