World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:3, No:8, 2009

Modeling of Reusability of Object Oriented Software System

Parvinder S. Sandhu, Harpreet Kaur, and Amanpreet Singh

***Abstract*—**Automatic reusability appraisal is helpful in evaluating the quality of developed or developing reusable software components and in identification of reusable components from existing legacy systems; that can save cost of developing the software from scratch. But the issue of how to identify reusable components from existing systems has remained relatively unexplored. In this research work, structural attributes of software components are explored using software metrics and quality of the software is inferred by different Neural Network based approaches, taking the metric values as input. The calculated reusability value enables to identify a good quality code automatically. It is found that the reusability value determined is close to the manual analysis used to be performed by the programmers or repository managers. So, the developed system can be used to enhance the productivity and quality of software development.

***Keywords*—**Neural Network, Software Reusability, Software Metric, Accuracy, MAE, RMSE.

I. Introduction

THE demand for new software applications is currently increasing at the exponential rate, as is the cost to develop them. The numbers of qualified and experienced professionals required for this extra work are not increasing commensurably [1]. Software professionals have recognized reuse as a powerful means of potentially overcoming the above said software crisis [2,3] and it promises significant improvements in software productivity and quality [4,5]. There are two approaches for reuse of code: develop the reusable code from scratch or identify and extract the reusable code from already developed code. The organizations that has experience in developing software, but not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir [4]. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed and existing software systems or legacy systems [6]. But the issue of how to identify reusable components from existing systems has remained relatively unexplored. In both the cases, whether we are developing software from scratch or reusing code from already developed projects, there is a need of evaluating the quality of the potentially reusable piece of software.

The aim of Metrics is to predict the quality of the software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. The requirement today is to relate 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.

Tracz observed that for programmers to reuse software they must first find it useful [7]. Experimental results confirm that prediction of reusability is possible but it involves more than the set of metrics that are being used [8]. According to Poulin [9], in some sense, researchers have fully explored most traditional methods of measuring reusability: complexity, module size, interface characteristics, etc., but the ability to reuse software also depends on domain characteristics. It means we should concentrate on evaluating the software in terms of its relevancy to a particular domain.

The contribution of metrics to the overall objective of the software quality is understood and recognized [10, 11, 12]. But how these metrics collectively determine reusability of a software component is still at its naïve stage although a number of attempts are mode in [15-22]. A neural network approach could serve as an economical and automatic tool, to generate reusability ranking of software components [13]. In this paper, neural network approach is extensively explored to automatic evaluate the reusability of Object-oriented software components in existing systems as well as the developed reusable components. Inputs to Neural Network system are provided in form of structural attributes of software component in form of metric values and output is the reusability value category. In this paper a different types of Neural Networks are experimented.

II. Proposed Methodology

Reusability evaluation System for Object oriented Software Components can be framed using following steps:

A. Selection & Refinement of Metrics

Selection and refinement of metrics targeting the quality

Harpreet Kaur is working as Lecturer in Department of Computer Science & Engineering, Rayat Bahra College of Engineering & Bio-Technology for Women, Sahauran, Mohali (India).

Parvinder S. Sandhu and Amanpreet Singh are associated with Rayat Bahra Institute of Engineering &Bio-Technology, Sahauran, Mohali (India).

of object oriented software system and perform parsing of the software system to generate the Meta information related to that Software. The metrics of [17] [22] are used and details of the metrics are as under:

i) Weighted Methods per Class (WMC): According to this metric if a Class C, has n methods and c_1, c_2 …, c_n be the complexity of the methods, then WMC(C) = $c_1 + c_2 + \ldots + c_n$. Mc Cabe'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 [23] [24].

"Tuned WMC" (TWMC) measure is used as input to the NF inference engine by restricting the WMC value in between 0 and 1 with help of sigmoid function shown below:

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

Where a=10 and c=0.5 .

ii) 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. Greater depth dilutes the abstraction and there is a need to set the minimum and maximum DIT value for a class as a contribution towards the reusability [23] [24].

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 all the ancestor classes coming in common path are added to the ancestor classes of alternative paths then that will be the true representation of the theoretical basis of the DIT metric.

"Lack of Tuned Degree of Inheritance" (LTDIT) measure is used as input to the NF inference system, in order to restrict the input value between 0 and 1.

iii) 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. Thus, 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 [23] [24].

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. 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. The NOC value of a class (say class '*i*') should reflect all the subclasses that share the properties of that class as shown in the following equation:

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

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 NF inference system.

iv) 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. 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 a need to set some maximum value of coupling level for its reusability and if the value of CBO for a class is beyond that maximum value then the class is said to be non-reusable [23] [24].

In order to restrict the input value between 0 and 1, we have used "Lack of CBO" (LCBO) measure as input to the NF inference system.

v) Lack of Cohesion in Methods (LCOM): Consider a Class C_1 with *n* methods M_1, M_2 ..., M_n. Let $\{I_j\}$ = set of instance variables used by method M_i .There are *n* such sets $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Let P = $\{(I_i , I_j) \mid I_i \cap I_j = \varnothing\}$ and Q = $\{(I_i , I_j) \mid I_i \cap I_j \neq \varnothing\}$. If all n sets $\{I_1\}$, $\{I_2\}$, $\{I_n\}$ are $\varnothing$ then P = $\varnothing$ [25]. Lack of Cohesion in Methods (LCOM) of a class can be defined as:

LCOM = |P| - |Q|, if |P| > |Q|
LCOM = 0 otherwise

The high value of LCOM indicates that the methods in the class are not really related to each other and vice versa. It means that 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 that class becomes non-reusable [23] [24].

"Tuned LCOM" (TLCOM) measure is used as input to the NF inference engine by restricting the LCOM value between 0 and 1 with help of sigmoid function as shown in the following equation:

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

Where a=4 and c=1.5.

B. Collection of Metric Values

Collect the object oriented components and generate the different metric values of the exemplars.

C. Modeling of the Reusability Data

The different Neural Network approaches are used for the Modeling of the reusability data as generated from the previous step. For each approach following steps are used:

- Perform the Training of the Neural Network

- After training tests the Neural Network on the basis of *Accuracy*, *MAE* and *RMSE*.

The mean absolute error is chosen as the standard error. The technique having lower value of mean absolute error is chosen as the best modeling or prediction technique.

- *Mean absolute error*

Mean absolute error, *MAE* is the average of the difference between predicted and actual value in all test cases; it is the average prediction error. The formula for calculating *MAE* is given in equation shown below [26]:

$$\frac{|a_1 - c_1| + |a_2 - c_2| + \ldots + |a_n - c_n|}{n} \quad (4)$$

Assuming that the actual output is *a*, expected output is *c*.

- *Root mean-squared error*

RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error as shown in equation given below [26]:

$$\sqrt{\frac{(a_1 - c_1)^2 + (a_2 - c_2)^2 + \ldots + (a_n - c_n)^2}{n}} \quad (5)$$

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared-error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values.

III. Results & Discussion

The proposed Neural based methodology is implemented in MATLAB 7.4 environment is one such facility that lends a high performance language for technical computing. A variety of neural network techniques are experimented in the study to predict reusability of the objected oriented software components. The list is as follows:

- Batch Gradient Descent without momentum
- Batch Gradient Descent with momentum
- Variable Learning Rate without momentum
- Variable Learning Rate training with momentum
- Resilient Backpropagation
- Fletcher-Reeves version of the conjugate gradient algorithm
- Polak-Ribiére Update version of the conjugate gradient algorithm
- Powell-Beale Restarts version of the conjugate gradient algorithm
- Scaled Conjugate Gradient
- Quasi-Newton BFGS Algorithm
- Quasi-Newton One Step Secant Algorithm
- Levenberg-Marquardt Algorithm
- Generalized Regression Neural Networks
- Self Organizing Network

The generalized structure of the Neural Network is shown in Fig. 1.

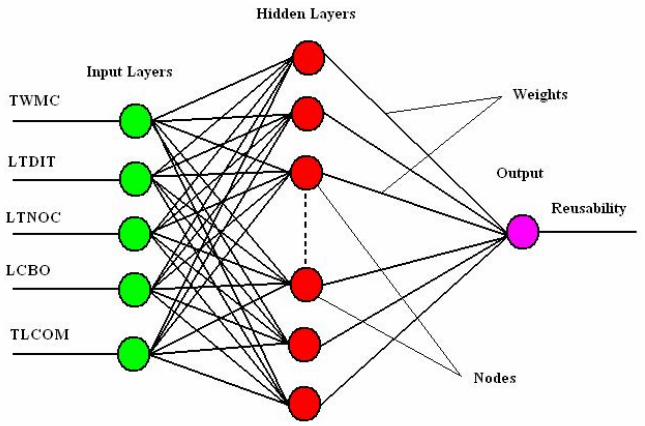

Fig. 1 Generalized structure of the Neural Network with Inputs and Outputs depicted

In each of the above mentioned technique first the network is created and training of the network is performed with the training dataset. There after the network is tested and the results are shown in Table I.

TABLE I
Performance Statistics of Various Neural Network Algorithms

Algorithm	Projects		
	CM1		
	Accuracy	*MAE*	*RMSE*
BGD	94.2529	0.2799	0.3282
BGDWM	97.7011	0.1561	0.1966
VLR	96.5517	0.1277	0.1798
VLRM	98.8506	0.1033	0.134
RB	100	0.0779	0.1001
FRCG	100	0.074	0.1003
PRUCG	100	0.0649	0.0899
PBRCG	100	0.0592	0.0806
SCG	100	0.0638	0.0818
QNBFGS	100	0.0589	0.0767
QNOSS	100	0.0786	0.1037
LM	100	0.0443	0.0605
GRNN	100	0.2391	0.27
SON	3.4483	2.6324	3.0021

As evidenced from Table I, Levenberg–Marquardt (LM) algorithm is the best among other different neural network algorithms under study with least MAE and RMSE values i.e. 0.0443 and 0.0605 respectively. The algorithm shows the highest *Accuracy* with 100% value. Training Performance of Levenberg–Marquardt (LM) algorithm for reusability evaluation is shown in Fig. 2 below:

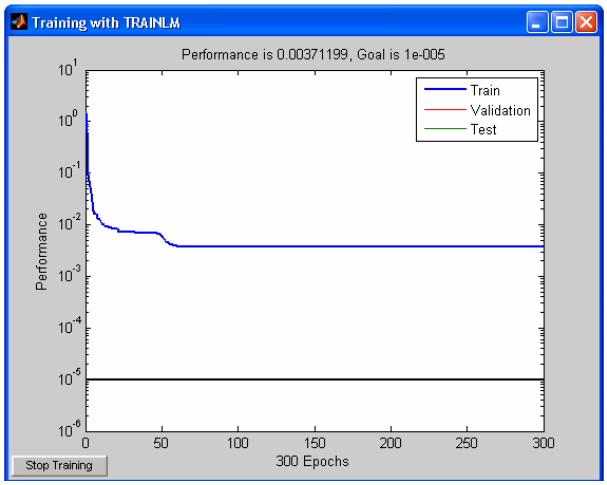

Fig. 2 Training Performance of Levenberg–Marquardt (LM) Algorithm for Reusability Evaluation

IV. Conclusion and Future Scope

The metric based approach is used successfully in designing the framework for evaluation of reusability of object oriented systems and Levenberg – Marguardt algorithm is proved to be best as compared to the other algorithms considered in this work. The performance of this neural network is better than the results of the Neuro-Fuzzy Technique as used in the literature. The Quasi-Newton BFGS Algorithm come out to be the second best algorithm for modeling of reusability data and the Self Organizing Network are showing the worst results among fourteen neural network algorithms under study. Hence, it is concluded that for non linear and complex engineering applications involving control, inference and analysis by and large Neural Network is an efficient technique.

The proposed approach is applied on the C++ based software modules/components and it can further be extended to the Artificial Intelligence (AI) based software components e.g. Prolog Language based software components.

References

[1] E. Smith, A. Al-Yasiri, and M. Merabti, A Multi-Tiered Classification Scheme For Component Retrieval, Proc. Euromicro Conference, 24(Vol. 2) (1998) 882 – 889.

[2] V.R. Basili, Software Development: A Paradigm for the Future, Proc. COMPAC '89, (Los Alamitos, Calif.: IEEE CS Press, 1989) 471-485.

[3] B.W. Boehm and R. Ross, Theory-W Software Project Management: Principles and Examples, IEEE Trans. Software Eng., 15(7), 1989, p. 902.

[4] W. Lim, Effects of Reuse on Quality, Productivity, and Economics, IEEE Software, 11(5, Oct. 1994), 23-30.

[5] H. Mili, F. Mili and A. Mili, Reusing Software: Issues And Research Directions, IEEE Trans. Software Eng., 21(6, June 1995) 528 - 562.

[6] G. Caldiera and V. R. Basili, Identifying and Qualifying Reusable Software Components, IEEE Computer, (1991) 61-70.

[7] W. Tracz, A Conceptual Model for Mega programming, SIGSOFT Software Engineering Notes, 16(3, July 1991) 36-45.

[8] Stephen R. Schach and X. Yang, Metrics for targeting candidates for reuse: an experimental approach, ACM, (SAC 1995) 379-383.

[9] J. S. Poulin, Measuring Software Reuse–Principles, Practices and Economic Models (Addison-Wesley, 1997).

[10] W. Humphrey, Managing the Software Process, SEI Series in Software Engineering (Addison-Wesley, 1989).

[11] L. Sommerville, Software Engineering, 4th edn. (Addison-Wesley, 1992).

[12] R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th edn. (McGraw-Hill, 2005).

[13] G. Boetticher and D. Eichmann, A Neural Network Paradigm for Characterizing Reusable Software, Proc. of the 1st Australian Conference on Software Metrics (18-19 November 1993).

[14] S. V. Kartalopoulos, Understanding Neural Networks and Fuzzy Logic-Basic Concepts and Applications (IEEE Press, 1996)153-160.

[15] Parvinder Singh Sandhu and Hardeep Singh, "Software Reusability Model for Procedure Based Domain-Specific Software Components", International Journal of Software Engineering & Knowledge Engineering (IJSEKE), Vol. 18, No. 7, 2008, pp. 1–19.

[16] Parvinder Singh Sandhu and Hardeep Singh, "Automatic Quality Appraisal of Domain-Specific Reusable Software Components", Journal of Electronics & Computer Science, vol. 8, no. 1, June 2006, pp. 1-8.

[17] Parvinder Singh Sandhu and Hardeep Singh, "A Reusability Evaluation Model for OO-Based Software Components", International Journal of Computer Science, vol. 1, no. 4, 2006, pp. 259-264.

[18] Parvinder Singh Sandhu and Hardeep Singh , "Automatic Reusability Appraisal of Software Components using Neuro-Fuzzy Approach, International Journal Of Information Technology, vol. 3, no. 3, 2006, pp. 209-214.

[19] Parvinder S. Sandhu and Hardeep Singh, "A Fuzzy Based Approach for the Prediction of Quality of Reusable Software Components", IEEE 14th International Conference on Advanced Computing & Communications (ADCOM 2006), NIT Suratkal, Dec. 20 – 23, 2006, pp. 761-764.

[20] Parvinder S. Sandhu and Hardeep Singh, "A Neuro-Fuzzy Based Software Reusability Evaluation System with Optimized Rule Selection", IEEE 2nd International Conference on Emerging Technologies (IEEE ICET 2006), Peshawar, Pakistan, Nov. 13-14, 2006, pp. 664-669.

[21] Parvinder Singh and Hardeep Singh, "A Neuro-fuzzy Based Approach for the Prediction of Quality of Reusable Software Components", 4th International Conference on Software Methodologies, Tools and Techniques (SoMeT 2005), Tokyo, Japan, Sept. 28-30, 2005, pp. 156-169. (http://www.booksonline.iospress.nl/)

[22] Parvinder S. Sandhu, P. P.Singh, H. Singh,, "Reusability Evaluation with Machine Learning Techniques", WSEAS TRANSACTIONS on COMPUTERS, issue 9, Volume 6, September 2007, pp. 1065-1076

[23] Chidamber, S.R. and Kemerer, C.F., "A Metric Suite for Object Oriented Design", IEEE Trans. Software Eng., vol. 20, 1994, pp. 476-493.

[24] Chidamber, S.R. and Kemerer, C.F., "Towards a Metrics Suite for Object Oriented Design", Proceedings Conference Object Oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, 1991, pp. 197-211.

[25] Boehm, B.W. and Ross, R., "Theory-W Software Project Management: Principles and Examples", IEEE Trans. Software Eng., vol. 15, no. 7, 1989, pp. 902.

[26] Ebru Ardil, Erdem Ucar, Parvinder S. Sandhu, "Software Maintenance Severity Prediction with Soft Computing Approach", International Conference on Computer, Electrical, and Systems Science, and Engineering, Feb. 25-27, 2009, Penang, Malaysia; vol. 50, ISSN: 2070-3724, pp. 139-144.