International Journal of Information Technology and Knowledge Management
January-June 2012, Volume 5, No. 1, pp. 205-209

MEASURING SOFTWARE REUSABILITY USING SVM BASED CLASSIFIER APPROACH

Ajay Kumar

ABSTRACT: Here we presented classification of the reusability of software components using Support Vector Machine (SVM). The identification of Reusable Software modules in Procedure Oriented Software System. Metrics has been used for the structural analysis of the different procedures. Software metrics for Procedure oriented paradigm has been used in this paper Cyclometric Complexity Using Mc Cabe's Measure, Halstead Software Science Indicator, Regularity Metric, Reuse frequency metric, Coupling Metric. The values of these Metrics will become the input dataset for the different neural network systems. Neural Network Based Approach is used to establish the relationship between different attributes of the reusability and serve as the automatic tool for the evaluation of the reusability of the procedures by calculating the relationship based on its training. Algorithms of neural network are experimented and the results are recorded in terms of Accuracy, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

Hence in this paper the proposed model can be used to improve the productivity and quality of software development.

Keywords: Software reusability, neural networks, MAE, RMSE, accuracy. support vector machine.

1. INTRODUCTION

Software reuse is the improvement efforts of the productivity of the software because reuse can result in higher quality software at a lower cost and delivered within a shorter time [1]. Reused software is more accurate than new software because already it has been tried and tested in working system.

Reusable software components have been promoted in recent years. The software development community is gradually drifting toward the promise of widespread software reuse, in which any new software system can be derived virtually from the existing systems. There are two approaches for reuse of code: develop the code from scratch or identify and extract the reusable code from already developed code. With the existence of the software there is less uncertainty in the cost of reusing which is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components as sub-systems are reused. Reusing software can speed up system production because both development and validation time should be reduced. Thus the reuse of software in systems development is a strategy that increases productivity and quality. Code reuse is the idea that a partial or complete computer program written at one time can be, should be, or is being used in another program written at a later time. The reuse of programming code is a common technique which attempts to save time and energy by reducing redundant work.

[1] Research Scholar, Department of Computer Science, NIMS University, Jaipur, India, E-mail: ic.garg188@gmail.com.

Major challenge is to develop a robust framework for software reuse. The proposed framework has two layers. The first layer is formed by best practices related to software reuse. Non-technical factors, such as education, training, incentives, and organizational management are considered. This layer constitutes a fundamental step before of the introduction of the framework in the organizations. The second layer is formed by important technical aspects related to software reuse, such as processes (reuse, reengineering, adaptation, component certification), environments, and tools (repository systems and its associated tools). This framework constitutes a solid basis for organizations that are moving towards an effective reuse program. Its elements not only help the organization in adopting reuse, but also guide it in the migration process, reducing its risks and failure possibilities

Reuse is an act of synthesizing a solution to a problem based on predefined solutions to sub problems. The reuse activity is divided into six major steps performed at each phase in preparation for the next phase. These steps are:

1. Developing a reuse plan or strategy after studying the problem and available solutions to the problem,
2. Identifying a solution structure for the problem following the reuse plan or strateg.
3. Reconfiguring the solution structure to improve the possibility of using predefined components available at the next phase.
4. Acquiring, instantiating, and modifying predefined components.
5. Integrating the components into the products for this phase, and. evaluating the products.

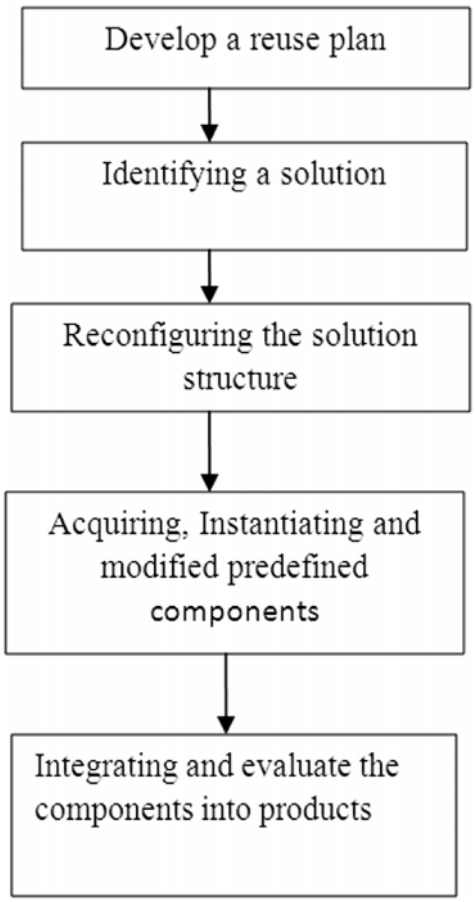

Figure 1.1: Six Major Steps of Formulate Neural Network.

2. RELATED WORK

The measure of a components reusability comes from its success. Other measures come from static code metrics. There are basically two approaches to evaluate software reusability: qualitative and empirical.

Author [5] identified a number of characteristics of those components, from existing systems, that are being reused at NASA laboratory and reported that the developers were successful in achieving a 32 percent reusability index. Selby's recent experimental study has identified two categories of factors that characterize successful reuse-based software development of large-scale systems: module design factors and module implementation factors [6]. 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 in size (source lines) and with many assignment statements (i.e. low Cyclometric Complexity). The modules reused without revision had the fewest faults, fewest faults per source line, and lowest fault correction effort.

Chen and Lee developed about 130 reusable C++ components and used these components in a controlled experiment to relate the level of reuse in a program to software productivity and quality [9]. The software metrics collected included the Halstead size, program volume, program level, estimated difficulty and effort. They found that lower the value of the software complexity metrics, the higher the programmer productivity.

Dunn and Knight also experimented and reported the usefulness of reusable code scavenging [11]. Chen, Nishimoto and Ramamoorty discussed the idea of subsystem extraction by using code information stored in a relational database [12]. They also described a tool called the C Information Abstraction System to support this process. Esteva and Reynolds [13] proposed the use of Inductive Learning techniques based on software metrics used to identify reusable modules. Their system was able to recognize reusable components.

Caldiera and Basili [14] proposed a tool called, Care that was used to identify reusable components according to a set of "reusability attributes" based on software metrics. The paper proposed four candidate measures of reusability based largely on McCabe and Halstead metrics. These attributes include measurement of utilization of the component in the problem domain, the cost of reuse and its quality.

Mayobre [15] described how these techniques can be extended and used to help in identifying data communication components of Hewlett-Packard.

Arnold [16] [17] mentioned 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.

The ESPRIR-2 project called REBOOT (Reuse Based on Object-Oriented Techniques) developed a taxonomy of reusability attributes. They listed Portability, Flexibility, Understandability and Confidence as four reusability factors. The analyst combines the individual metric values into an overall value of reusability.

3. METHODOLOGY OF WORK

Reusability evaluation System for function Based software Components can be framed using following steps:

1. Selection of Metric Suit for Procedure Oriented Paradigm: A framework of metrics is proposed for structural analysis of procedure or function-oriented software. The code of software is parsed to calculate the metric values. The following suits of metrics are able to explore different structural dimensions of procedure oriented components.

The proposed metrics for Function Oriented Paradigm are as follows:

(a) Cyclometric Complexity Using Mc Cabe's Measure [23] [24].

(b) Halstead Software Science Indicator [23] [25].

(c) Regularity Metric [23][25].

(d) Reuse-Frequency Metric [23] [25].

(e) Coupling Metric [23].

2. Calculate the metric values of the sampled software components.

3. Use SVM Based Classifier for the Reusability Prediction: SVM is a useful technique for data classification. Even though it's considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

   Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1.

   To construct an optimal hyperplane, SVM employees an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, SVM models can be classified into four distinct groups: (a) Classification SVM Type 1 (also known as C-SVM classification) (b) Classification SVM Type 2 (also known as nu-SVM classification) (c) Regression SVM Type 1 (also known as epsilon-SVM regression) (d) Regression SVM Type 2 (also known as nu-SVM regression).

3.1. Design & Evaluate Neural Network System

The following Neural Network algorithms are experimented:

1. Batch Gradient Descent.
2. Batch Gradient Descent with momentum.
3. Variable Learning Rate.
4. Variable Learning Rate training with momentum.
5. Resilient Backpropagation.

The following are the steps for each Neural Network based system:

(a) Perform the training of the different neural networks with the training dataset.

(b) The trained Neural Network is evaluated against the testing data on the different comparison criteria as described in the next step.

3.2. Comparison Criteria

The comparisons are made on the basis of value of MAE, RMSE and Accuracy values of the neural network model. The details of the MAE and RMSE are given below:

Mean absolute error (MAE): 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.

Root mean-squared error (RMSE): 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. The conclusions are made on the basis of the results calculated in the previous section.

4. IMPLEMENTATION AND RESULTS

In this paper, the implementation of the algorithm is done in Matlab environment and SVM toolbox for Matlab is used. The dataset of procedure oriented software is collected and Batch Gradient Descent, Batch Gradient Descent with momentum, Variable Learning Rate, Variable Learning Rate training with momentum, Resilient Backpropagation, based neural networks are experimented to obtain the results in terms of Accuracy, MAE and RMSE values. The mean or average values of algorithm under study depict that the accuracy, MAE AND RMSE values of the resilient back propagation (RB) algorithm experimented respectively. In this technique first the network is created and training of the network is performed with the training data set.

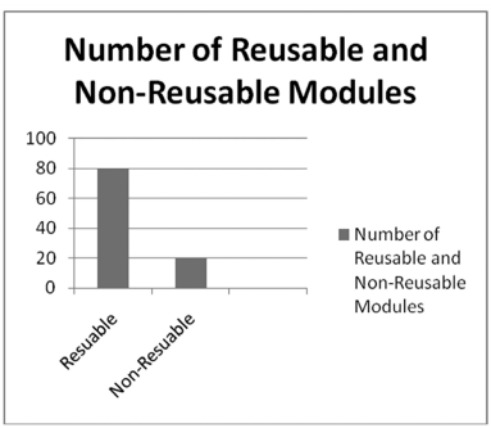

Figure 1.2: Bar-Chart of Count of Examples of the Reusability Output Attribute in the Dataset.

The function oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of two numeric labels i.e. Reusable and Non-Reusable. The Graphical representation of the count of the number of examples of certain reusability label is shown in the Figure 1.1

Then SVM clustering based algorithm is implemented in Matlab 7.4. environment and svm clustering algorithm toolbox for Matlab is used. The input attribute-wise statistical details of the count of the examples of the labels are shown in Table 1, Table 2, Table 3, Table 4, Table 5. The input attributes are expressed in the three linguistic labels i.e. 1, 2, and 3. The label 1 corresponds to the Low value, label 2 corresponds to the Medium value and label 3 corresponds to the high value.

Table 1
Statistics of the Input Attribute Coupling in the Dataset

Selected attribute

Name: Coupling Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	1	38
2	2	48
3	3	23

Table 2
Statistics of the Input Attribute Volume in the Dataset

Selected attribute

Name: Volume Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	1	16
2	2	78
3	3	15

Table 3
Statistics of the Input Attribute COMPLEXITY in the Dataset

Selected attribute

Name: Complexity Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	1	20
2	2	77
3	3	12

Table 4
Statistics of the Input Attribute Regularity in the Dataset

Selected attribute

Name: Regularity Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	1	13
2	2	67
3	3	29

Table 5
Statistics of the Input Attribute Reuse-Frequency in the Dataset

Selected attribute

Name: Reuse_Frequency Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	1	55
2	2	9
3	3	45

The given data is with five Input Attributes i.e. Coupling, Volume, Complexity, Regularity, Reuse_ Frequency and one Output attributes named as Reusability Level of the Software Component. Then SVM clustering based algorithm is implemented in Matlab 7.4.

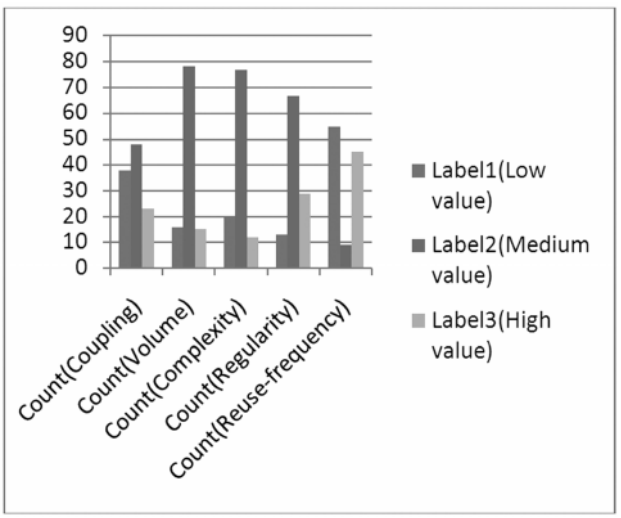

Figure 1.3: Bar Chart of Statistics of the Input Attribute in Data Set.

The algorithm learns about its environment through set of input-output training samples. We will take different input values to conduct the output. On running algorithm in Matlab it will show different output as based on SVM classifier for software prediction.

5. CONCLUSION

In this paper, Support Vector Algorithm (SVM) based algorithms are experimented to develop the reusability evaluation system for procedure oriented software systems like McCabe's Cyclometric Complexity Measure for Complexity measurement, Regularity Metric, Halstead Software Science Indicator for Volume indication, Reuse Frequency metric and Coupling Metric are used for structural analysis of a software module. First of all, randomly selection of training and test sets are made. Thereafter, Training of support vector machine classifier is performed with the training dataset created. The training data is provided to SVM in form of Matrix, where each row

corresponds to an observation or replicate, and each column corresponds to a feature or variable. Groups are also provided to the SVM as Column vector, character array, or cell array of strings for classifying data in Training into two groups. It has the same number of elements as there are rows in Training. Each element specifies the group to which the corresponding row in Training belongs.

The trained SVM is now used to classify the test dataset and the performance of the classification is recorded in terms of the Correct Rate and Error Rate.

REFERENCES

[1] Anderson, J.A (2003)., "An Introduction To Neural Networks", Prentice Hall of India.

[2] Arnold, R.S. (1990), "Heuristics for Salvaging Reusable Parts From Adav Code, SPC Technical Report", ADA_REUSE_ HEURISTICS-90011-N, March 1990.

[3] Arnold, R.S. (1990)., "Salvaging Reusable Parts From Ada Code: A Progress Report, SPC Technical Report", SALVAGE_ADA_PARTS_PR-90048-N, September 1990.

[4] Basili, V. R. and Rombach, H. D. (1988)., The TAME Project: Towards Improvement Oriented Software Environments", IEEE Trans. Software Eng., 14, No. 6, June 1988, pp. 758-771.

[5] Basili, V.R. (1989)., "Software Development: A Paradigm for the Future", Proceedings COMPAC'89, Los Alamitos, California, IEEE CS Press, 1989, pp. 471-485.

[6] Boetticher, G. and Eichmann, D. (1993)., A Neural Network Paradigm for Characterizing Reusable Software", Proceedings of the Australian Conference on Software Metrics, Australia, July, 1993, pp. 234-237.

[7] Boetticher, G., Srinivas, K. and Eichmann, D. (1990)., "A Neural Net-Based Approach to the Software Metrics", Proceedings of the 5th International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, 14-18 June 1990, pp. 271-274.

[8] Caldiera, G. and Basili, V. R. (1991)., "Identifying and Qualifying Reusable Software Components", IEEE Computer, February 1991.

[9] Chen, Y. F. Nishimoto, M. Y. and Ramamoorty, C. V. "The C Information Abstraction System", IEEE Trans. on Software Engineering, 16, No. 3, March 1990.

[10] Dunn, M. F. and Knight, J. C. (1993)., "Software Reuse in Industrial Setting: A Case Study", Proc. of the 13th International Conference on Software Engineering, Baltimore, MA, 1993. pp. 56-62.

[11] Esteva, J. C. and Reynolds, R. G. (1991)., "Identifying Reusable Components using Induction", International Journal of Software Engineering and Knowledge Engineering, 1, No. 3, 1991, pp. 271-292.

[12] Frakes, W.B. and Kyo Kang (2005)., "Software Reuse Research: Status and Future", IEEE Trans. Software Engineering, 31, issue 7, July 2005, pp. 529-536.

[13] Herenji, H.R and Khedkar,P(1992),"Learning and Tuning Fuzzy Logic Controllers through Reinforcements", IEEE Trans. on Neural Networks, 3, 1992, pp. 724-740.

[14] Jang, J-S. R. and Sun, C.T. (1995)., "Neuro-Fuzzy Modeling and Control", Proceeding of IEEE, March 1995, pp. 123-135.

[15] Jerome Feldman (1996)., "Neural Networks - A Systematic Introduction", Berlin, New-York, 1996.

[16] "Software Reusability and Efficiency: A Scientific And Technological Study", Undertaken by Parallab, Bergen Center for Computational Science, University of Bergen (Norway) for the Enacts Network, Sectoral Report, Final Version-April 2004, http://www.enacts.org.

[17] Sonia Manhas, Rajeev Vashisht, Parvinder S. Sandhu and Nirvair Neeru, "Reusability Evaluation Model for Procedure Based Software Systems", International Journal of Computer and Electrical Engineering, 2, No.6, December, 2010.

[18] R W Selby, "Enabling Reuse-Based Software Development of Large-Scale Systems", IEEE Transactions on Software Engineering (2005), 31, Issue 6, pp. 495-510.

[19] EbruArdil, Parvinder S. Sandhu, "A Soft Computing Approach For Modeling of Severity of Faults In Software Systems ", International Journal of Physical Sciences, 5(2), February, 2010, ISSN 1992 - 1950, pp. 74-85. ISI Indexed.

[20] Parvinder S. Sandhu, Pavel Blecharz and Hardeep Singh, "A Taguchi Approach to Investigate Impact of Factors for Reusability of Software Components", Transactions on Engineering, Computing and Technology, 19, Jan. 2007, ISSN 1305-5313, pp. 135-140.

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

[22] Parvinder Singh and Hardeep Singh (2005), "Critical Suggestive Evaluation of CKMETRIC", Proc. of 9th Pacific Asia Conferenceon Information Technology (PACIS-2005), Bangkok, Thailand, July 7–10, 2005, pp. 234-241.

[23] http://www.statsoft.com/textbook/support-vector-machines/

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

[25] Richard, W.S. (2005)., "Enabling Reuse-Based Software Development of Large-Scale Systems", IEEE Trans. on Software Engineering, 31, No. 6, June 2005, pp. 495-510.

[26] R.S. Pressman, "Software Engineering: A Practitioner's Approach", McGraw-Hill Publications, 5th edition,2005.