I.J. Modern Education and Computer Science, **2014, 8, 45-58**
Published Online August 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.08.06

MLP based Reusability Assessment Automation Model for Java based Software Systems

Surbhi Maggo
Jaypee Institute of Information Technology, India
surbhi.maggo@gmail.com

Chetna Gupta
Jaypee Institute of Information Technology, India
Chetnagupta04@gmail.com

***Abstract*—**Reuse refers to a common principle of using existing resources repeatedly, that is pervasively applicable everywhere. In software engineering reuse refers to the development of software systems using already available artifacts or assets partially or completely, with or without modifications. Software reuse not only promises significant improvements in productivity and quality but also provides for the development of more reliable, cost effective, dependable and less buggy (considering that prior use and testing have removed errors) software with reduced time and effort. In this paper we present an efficient and reliable automation model for reusability evaluation of procedure based object oriented software for predicting the reusability levels of the components as low, medium or high. The presented model follows a reusability metric framework that targets the requisite reusability attributes including maintainability (using the Maintainability Index) for functional analysis of the components. Further Multilayer perceptron (using back propagation) based neural network is applied for the establishment of significant relationships among these attributes for reusability prediction. The proposed approach provides support for reusability evaluation at functional level rather than at structural level. The automation support for this approach is provided in the form of a tool named JRA^{2}M^2 (Java based Reusability Assessment Automation Model using Multilayer Perceptron (MLP)), implemented in Java. The performance of JRA^{2}M^2 is recorded using parameters like accuracy, classification error, precision and recall. The results generated using JRA^{2}M^2 indicate that the proposed automation tool can be effectively used as a reliable and efficient solution for automated evaluation of reusability.

***Index Terms*—**Back propagation; Maintainability Index; Metrics; Multi Layer Perceptron; Neural Network; Procedure Oriented; Reusability.

I. INTRODUCTION

Human dependability on computer software has increased multifold in the past few decades, making software very omnipresent. This requires development of more and more complex, sophisticated, reliable and secure software systems on a very large scale. Such a growth in software requirements is exponential in nature and is difficult to achieve even with the advancements in technology, enhanced hardware performances, increase in storage and memory capacity, improved computing architectures and rising number of computer professionals [1]. This has placed immense pressure on software engineers for fulfilling software requirements timely with high quality and reliable software thus keeping software productivity in pace with the demands placed on the software industry. Several years of research and study have shown that using the traditional practices for software development has led to widening the gap between the demands placed on the software industry and their fulfillment, and software reuse is the only realistic, practical and technically feasible solution that could help reduce this gap while improving software quality.

Conceptually software reuse refers to the development of new software systems using the already available software assets and resources in combination with each other or some newly developed components, rather than from scratch. As proposed by Mcllory in [2], the formal idea of software reuse proposed the development of an industry of reusable software components and the industrialization of the production of application software from off-the-shelf components. Reuse possesses great potential for significant improvements in software productivity, quality and reliability [3]. Further software reuse also leads to the accelerated development (reducing both development and maintenance time and effort) of more reliable, cost and time effective, dependable, maintainable, less buggy and error free (considering that prior use and testing has led to removal of bugs) software products. Other potential benefits of reusability include effective use of specialist, reduced process risk, standard compliance etc. [4]. Software reusability thus can be defined as the measure of the ease of using the formerly acquired concepts and objects in new contexts [5]. This ease of reuse depends on certain attributes and factors that influence the reusability of a software component. Loose coupling, modularity, high cohesion, separation of concerns, ease of understanding, proper documentation, information hiding and low complexity are a few attributes that increase the likelihood of a component of being reusable [6].

Software reuse has long been misconstrued as source code reuse, whereas the term software reuse refers to the reuse of any artifact or asset or product associated with software development, be it algorithm, requirement, plan, design, documentation, estimation templates, test plans or test cases, user manuals or even human interfaces [7]. Software Reuse has been an active and open area of research over the past 35 years where research challenges associated with systematic reuse still exist. Reuse can be perceived as applicable in software development industry in two different forms, one being "development *for* reuse" and the other being "development *with* reuse". Developing *for* reuse refers to the design and development of software in such a way that it can be reused in the future whereas developing *with* reuse refers to currently benefiting from reuse by using the already existing software resources for development of new ones. Clearly development *with* reuse offers greater potential benefits with lesser efforts as compared to the former one, still there are a number of risks and challenges associated with it that include identification of components that are worth reuse, understanding and adapting these components and maintaining them. Researchers have presented a number of empirical approaches for measuring reusability based on different complexity based and other metrics as discussed in related work. Lately the focus of research in the field of software reuse has drifted towards the development of functions that could help in the establishment of meaningful relationships among various reusability attributes represented (measured) by reusability metrics for the evaluation of reusability levels (high, medium or low) and identification of reusable software components. For the generation of such functions domains like neural networks, data mining, machine learning, artificial intelligence etc are being explored by researchers. Many such proposed approaches are presented in a tabular form in section 2.

The presented work proposes an efficient and reliable automated solution to the problem of software reusability measurement and prediction of reusability levels (low, medium or high) of procedure based object oriented software components thereby easing and systematizing the software reuse process and reducing the risks associated with it. The proposed solution model follows a reusability metric framework that targets the requisite reusability attributes including maintainability (using the Maintainability Index) for the functional analysis of the software components. Further Multilayer perceptron (using back propagation) based neural network is applied for the establishment of significant relationships among these attributes for reusability level prediction. The proposed approach works at functional level rather than at structural level. The performance of the tool JRA^{2}M^2 (Java based Reusability Assessment Automation Model using MLP) is evaluated using the results generated by it using parameters like accuracy, classification error, precision and recall (based on the confusion matrix generated). The results generated by JRA^{2}M^2 indicate that the presented automation model can be effectively used as an efficient and reliable solution for the automated identification of reusable procedure based object oriented software components from the existing resources.

The remaining paper is organized as: the next section talks about the related work present in the literature of software reuse. Section 3 presents the problem formulation followed by the Proposed solution model for reusability measurement presented in section 4. The next section 5 presents an overview of the results generated by the proposed automation tool JRA^{2}M^2 implemented in Java. JRA^{2}M^2 is compared with other existing approaches for measuring reliability of various parameters, the results of which are presented in section 6. Finally section 7 and 8 discuss the applications and significance of the proposed system and the conclusion respectively.

II. Related Work

The literature of software reuse presents numerous approaches for the identification and assessment of reusability levels of the software components. Initial research primarily focused on: 1) identifying the attributes and qualities of a software component that could determine its reuse potential and 2) establishing metrics that could be used to measure the degrees or levels of these attributes by direct or indirect means in order to predict reusability. Further with the advancements in the field of computer science, software reuse researchers tried to explore relationships between these attributes to see how these attributes can be used collectively for reusability evaluation and identification of reusability levels of software components. Many fields like data mining, artificial intelligence, machine learning, neural networks and genetic algorithms etc have been explored by researchers lately with the motive of generating such functions that can establish relationships among various reusability attributes and can efficiently automate collective reusability evaluation using these metric relationships. Different works proposed using these domains are presented in table 1 below. The approaches presented in table 1 use functional or structural level metric measures to support the reusability evaluation using domains like data mining, machine learning, neural network and also genetic algorithms. Manhas et al. [8] and Kumar [10] explore the Caldiera and Basili (CB) metric suite along with neural networks for reusability assessment. Manhas et al. [8] experiment with five different neural network algorithms under their approach that included Batch Gradient Descent, Batch Gradient Descent with momentum, Variable Learning Rate, Variable Learning Rate training with momentum and Resilient Back propagation. They also provide for a performance based comparison of these algorithms based on mean accuracy, mean MAE and mean RMSE. Kumar [10] explores the reusability evaluation as an application of Support Vector Machine (SVM). Neural networks approaches like Neuro fuzzy neural network and multilayer perceptron based network for reusability prediction have been used by Sandhu et al. [14] and Singh et al. [18] respectively but at structural level. Both use the Chidamber and Kemerer (CK) metric suite for the structural analysis of the software components for testing reusability levels. Structural analysis using CK suite has been supported by different data mining techniques as well. Shri et al. [9] propose the use of a hybrid k- means and decision tree based approach along with this structural analysis (using CK suite). A very similar model has been proposed by Sandhu et al. [15] that uses the CK suite and k-means algorithm for software reusability prediction.

Sandhu et al. [19] also propose an approach that uses a hybrid of Neuro fuzzy and genetic algorithm for identifying reusable software components at the functional level with the help of the CB metric suite. Cheema et al. [11] and Saini et al. [12] experiment and explore the domain of machine learning for reusability evaluation at functional level. They use k-Nearest neighbors and Density Based Spatial Clustering of Applications with Noise (DBSCAN) respectively. The later highlights the concept of density of components as important for reuse evaluation. Goel et al. [16] achieve appreciable prediction results with function oriented software metrics [16] and expectation maximaization algorithm. Kanellopoulos et al. [17] employ a metric framework along with k-Means for evaluating the maintainability and hence reusability of OO software systems. The solution is semi automated as the parsing engine extracts the data from the source code and stores it on a database. Most of the approaches presented in this section lack proper automation support, that is provided by the automation model proposed in this paper.

Table 1. Literarure Survey

Approach Proposed By	Domain	Scope	Metrics & Algorithms Used
Manhas et al. [8]	Neural Network	Functional level Object Oriented	CB[20], Neural Network
Shri et al. [9]	Data Mining	Structural level Object Oriented	CK[21], Hybrid k-Means & Decision Tree
Kumar [10]	Neural Network	Functional level Object Oriented	CB, SVM
Cheema et al. [11]	Machine Learning	Functional level Object Oriented	CB, k-NN
Saini et al. [12]	Machine Learning	Functional level Object Oriented	CB, DBSCAN
Czibula et al. [13]	Data Mining	Object Oriented Paradigm	S/W Metric[13], Hierarchical
Sandhu et al. [14]	Neural Network	Structural level Object Oriented	CK, Fuzzy Neuro
Sandhu et al. [15]	Data Mining	Structural level Object Oriented	CK, k-Means
Goel et al. [16]	Data Mining	Function Oriented	S/W Metric[16], Expectation Maximization
Kanellopoulos et al. [17]	Data Mining	Object Oriented Systems	S/W Metric[17], k-Means
Singh et al. [18]	Neural Network	Structural level Object Oriented	CK, Multi Layer Perceptron
Sandhu et al. [19]	Genetic Algorithm	Functional level Object Oriented	CB, Hybrid fuzzy Genetic Algorithm

III. Problem Formulation

Software reuse offers a great deal of potential in terms of improvements in quality and productivity. Software reuse acts as a boon for the software industry as it provides for the development of error free software products along with gains in time of development and cost and effort required. The concept of software reuse can be realized using two different approaches: a) identifying reusable resources from existing software reservoirs and using them for the development of current projects/software, b) initiate the development of reusable software products that can be reused at later stages. Using the later approach towards reuse i.e. development of reuse libraries by creating reusable components from scratch involves an additional associated designing and development cost, which can be avoided using the earlier approach based on assessment of the existing software repositories in order to identify and extract the reusable software components and utilizing them in the current scenario. Numerous reusability metrics and measures based approaches for the assessment of reusable components have been proposed in the literature of software reuse as presented in section 2 on Related Work, however the issue of how these measures support each other and contribute collectively to the reusability level evaluation, is relatively unexplored. With the capabilities of a neural network like extracting and storing knowledge, reasoning like humans and training itself with experience, it can prove to be immensely helpful in the establishment of relationships among these metrics for the collective reusability level evaluation process.

Hence the paper proposes a procedure oriented reusability metric framework and neural network based efficient and reliable system for automated evaluation of components identifying them as high, medium or low reusable components. The model presented in the paper focuses on a Multi-layer perceptron based neural network for the mapping of exact relationships among the various reusability metrics for the correct identification of the reusability levels of the software components. The training process followed by the MLP based neural network model uses the back propagation algorithm thus minimizing the risks associated with wrong reuse and making the prediction far more accurate and efficient, thereby providing for promising results in predicting reusability levels. Software developers can thus use the proposed Java based automation model for efficient and reliable identification of procedure based reusable components, which works at functional level rather than at structural level for the precise prediction of reusability levels of procedure based object oriented software systems.

IV. Proposed Solution/ Methodology

The basic framework of the proposed tool for identification of reusable components is presented in fig 1 below. Our tool JRA^{2}M^2 extracts components from existing inventory of software resources, further a metric framework followed by a neural network based module is used for evaluating the reusability levels of different components.

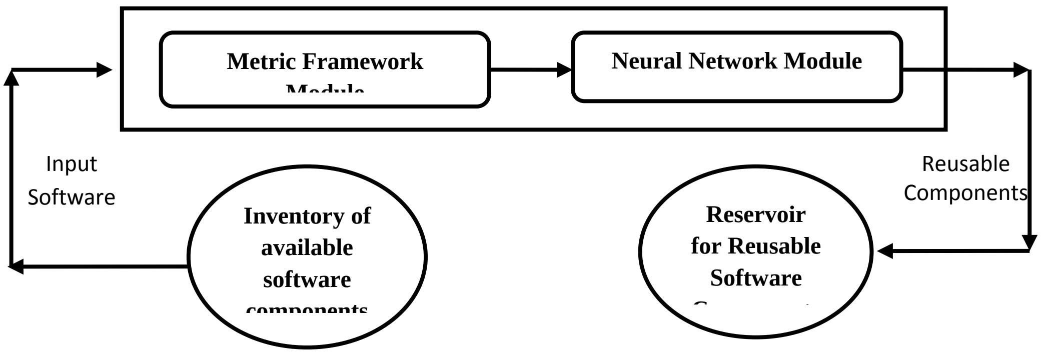


Fig. 1. Framework for proposed tool

A. *Software Metric Framework Module:*

The metric framework modules assess the input software components for various carefully selected reusability attributes. This module consists of sub modules that include: Metric Selection Module, Component Extractor and Analyzer and Metric Calculator. Fig 2a shows in detail the Metric Framework module with its sub modules.

- *Metric Selection Module:*

The first step of the system model is the selection of metrics for measuring reusability attributes. The evaluation model presented in our paper attempts to select metrics that either provide a direct measure for the reusability attributes or indirectly assess them by providing the evidences of the attribute's presence. These attributes that are responsible for making a component reusable in another system include its low reuse cost, its quality and its functional usefulness in the context of its application domain. These reusability attributes namely Quality, Cost and Usefullness are presented in the table 2 below along with the factors that affect these attributes respectively. The table 2 also presents with each factor, one or more associated metrics that either measure these factors directly or indirectly indicate their presence at functional level.

The metrics selected under our approach include Cyclomatic complexity (CC), Halstead Program Volume (HPV), Regularity Metric (RM), Reuse Frequency (RF) and Maintainability Index (MI). The values generated by these metrics are interpreted as high, low or medium using [20,22] in order to determine the reusability values.

Under the presented approach the maintainability index (MI) has been considered as an important metric for the precise and accurate prediction of reusability levels. Here in the proposed reusability prediction model we explore and use MI as our fifth metric in order to take into account maintenance efforts in case of partial reuse or reuse with modification. MI is an effective way of evaluating software by quantifying software's maintainability, as it acts as an excellent guide to direct human investigation thereby identifying maintainable components. It helps identify components having designs and plans in closer proximity with the problem domain and those with greater readability and reliability thus having a very positive impact on the reusability level of such software components [22]. With the help of MI the presented system is able to provide support for identifying components appropriate not only for direct reuse but also for partial reuse or reuse after modification easily and in a very time and cost effective manner.

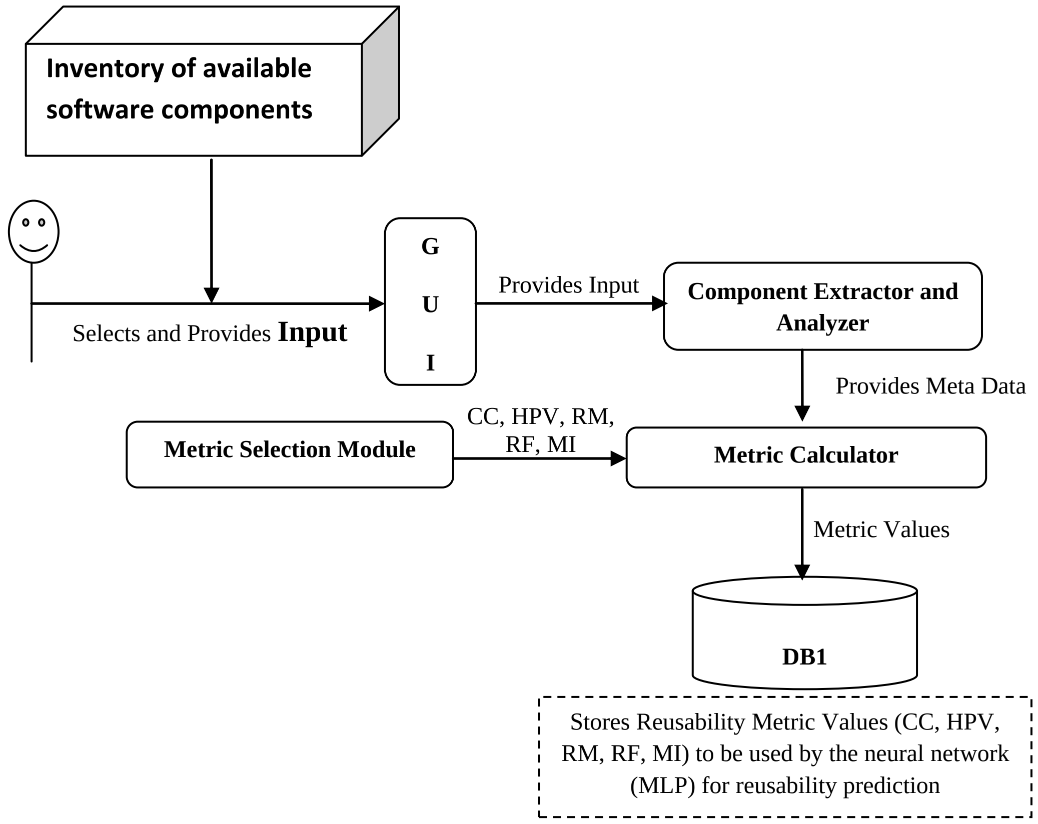

Inventory of available software components
G U I
Selects and Provides Input
Provides Input
Component Extractor and Analyzer
Provides Meta Data
CC, HPV, RM, RF, MI
Metric Selection Module
Metric Calculator
Metric Values
DB1
Stores Reusability Metric Values (CC, HPV, RM, RF, MI) to be used by the neural network (MLP) for reusability prediction

Fig .2 a. Metric Framework module and its sub modules

Table 2. Factors affecting reusability

Reusability Attribute	**Factors Affecting the attribute**	**Metric Measures for direct or indirect assessment these factors**
Usefulness	Variety of Functions Commonality of Functions - Within a System - Within a Domain - Overall	CC, RM RF RF
Quality	Ease of Modification Correctness Testability Readability Performance - Time - Space	MI HPV CC MI MI
Cost	Packaging Use in New Systems - Retrieval - Integration - Modification Extraction - Identification - Qualification	CC, HPV RM HPV, RM, MI HPV HPV

- *Component Extractor and Analyzer* and *Metric Calculator*:

The prediction model proposed is based on the automation of the selected metrics for the extracted components from the input object oriented software system. The component extractor and analyzer module, extracts modular class based components from the software system under test and analyzes them in order to calculate parameters required for the generation of the selected metric values. Further, a set of acceptable values for all these metrics are determined for reusability level prediction. These acceptable values can be either simple ranges or even more sophisticated relations among different metrics [20,23].

The five metrics selected for the reusablility level assessment model are explained in detail in the following section.

1) Cyclomatic Complexity – It is a software metric that reflects the complexity of a program, with the help of its control flow graph. The value of Cyclomatic complexity of a component is obtained using the following equation as proposed by Mc Cabes [24]:

$$\text{CC=Number of Predicate Nodes+1} \qquad (1)$$

Number of predicate nodes represent the decision nodes like if-else, for and while statements etc. in the software code components.

2) Halstead Software Science Indicator – This metric is used to indicate the program volume of the source code of the software components. The equation used to express the Halstead program volume based on Halstead software science indicator [25] is expressed as follows:

$$\text{Halstead Volume} = N1+N2\log2(\eta1+\eta2) \qquad (2)$$

Where, $\eta1$ is the number of distinct operators that appear in the program, $\eta2$ is the number of distinct operands that are present in the program, N1 is the total number of operator occurances and N2 is the total number of operand occurances in the component.

3) Regularity Metric – [20] The notion behind Regularity is to predict length based on some Regularity assumptions. Regularity is the ratio of the estimated length to the actual length. As actual length (N) is the sum of N1 and N2. The estimated length is shown in the following equation:

$$\text{Estimated Length} = N' = \eta1\log2\,\eta1 + \eta2\log2\,\eta2 \qquad (3)$$

The closeness of the estimate is a measure of the Regularity of Component coding is calculated as:

$$\text{Regularity} = 1 - \{ (N-N')/N \} = N'/N \qquad (4)$$

4) Reuse Frequency – Reuse frequency is calculated by comparing the number of static calls addressed to a component with a number of calls addressed to the component whose reusability is to be measured.

$$\text{Reuse-frequency} = \frac{\eta\,(C)}{\frac{1}{M}\sum_{i=0}^{M}\eta\,(S_i)} \qquad (5)$$

5) Maintainability Index – Maintainability Index is a software metric which measures how maintainable (easy to support and change) the source code is. The maintainability index is calculated as a factored formula:

$$MI = 171-5.2*\ln(V)-0.23*(G)-16.2*\ln(LOC) \qquad (6)$$

Where LOC is the lines of Codes, G is Cyclomatic complexity and V is volume of code.

The analyzer and metric calculator modules take an input system from the GUI, extracts and analyzes the components from the input system in order to obtain the metric values that can be further used as metadata for the next Neural Network module. Of the five selected metrics, for the first two metrics i.e. Cyclomatic Complexity and Halstead Program Volume, the values obtained have no specified range as they depend directly on the size of the component. Hence as part of the preprocessing of generated meta data for the next Neural Network module, these metric values are normalized to bring them into a particular range i.e. from 0-10. Although the values of the other three metrics – regularity, reuse frequency and maintainability index, already lie in a predefined range i.e. 0-1 for regularity metric and reuse frequency and 0-100 for maintainability index.

B. Neural Network Module:

The meta data generated in the above module in the form of pre processed reusability metric values of all the components, works as the input to the neural network model. The metadata of the various components is used to train and develop a Multilayer perceptron (MLP) based neural network for the prediction of readability levels as low, medium and high. Fig 2b represents the Neural network module. Further the generated levels (reusability classes) are evaluated using criteria like recall, precision, accuracy and classification error on the basis of the confusion matrix generated.

A neural network [26] is an information processing model influenced by the working and the behavior of the human brain. The numerous neurons from the key processing elements of the neural network model. A neural network with its capabilities like extracting and storing knowledge, reasoning like humans and training itself with experience, can be employed to solve difficult and complex problems under a wide diversity of applications including nonlinear system, optimization, functional approximation, pattern classification and recognition. Hence in the proposed approach neural network is worked upon in order to develop approximate functions that could represent relations among the selected reusability metrics for the prediction of reusability levels of the software components as high, medium or low.

The neural network model used here (MLP) is a perceptron [27] based model containing multiple hidden layers that uses backpropagation algorithm [28] for reusability evaluation. The neurons in a network are arranged into groups that form layers. A multilayer perceptron model [29] as the name suggests contains multiple layers of neurons: an input layer, one or more hidden layers and an output layer [29]. Fig 3 presents an MLP network with the three layers used in the presented approach. In the model presented in the paper the reusability metrics are taken as the input neurons of the first input layer and the reusability classes (high, medium and low) are generated as the output layer neurons.

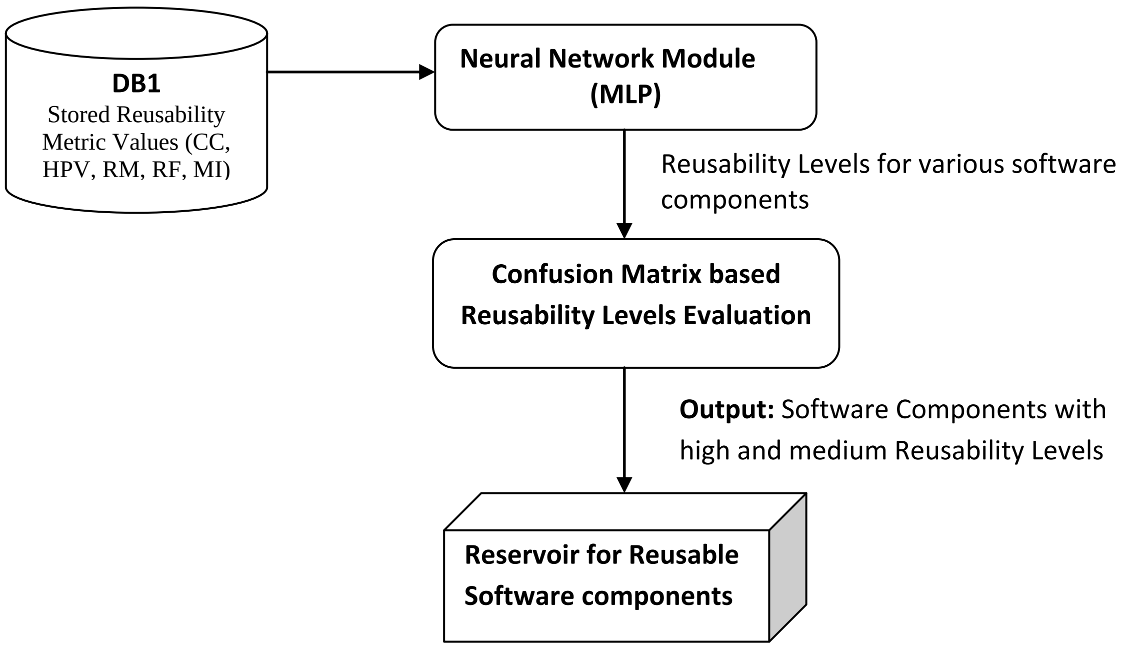

DB1
Stored Reusability Metric Values (CC, HPV, RM, RF, MI)
Neural Network Module (MLP)
Reusability Levels for various software components
Confusion Matrix based Reusability Levels Evaluation
Output: Software Components with high and medium Reusability Levels
Reservoir for Reusable Software components

Fig 2 b: Neural Network Module

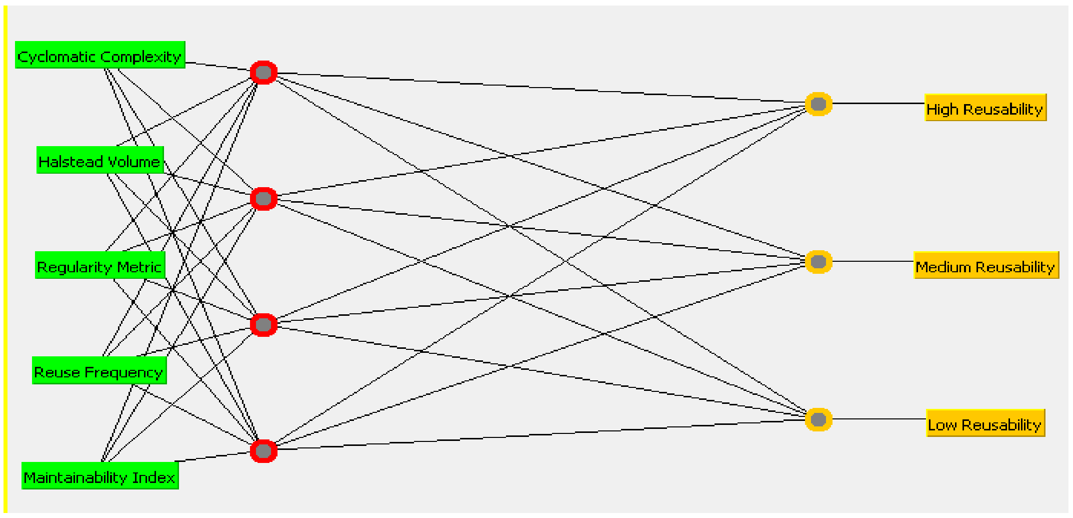


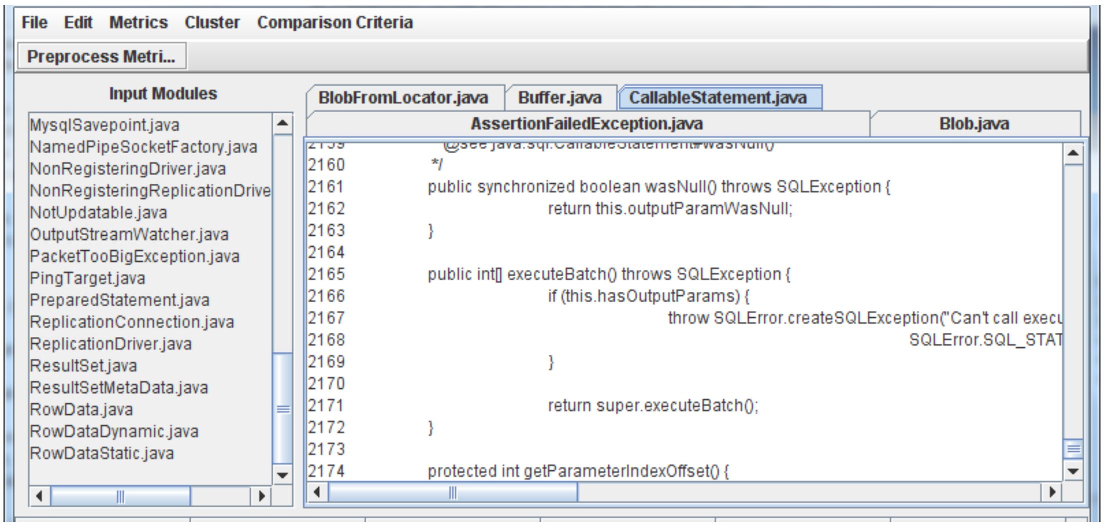
Fig. 3. MLP network with three layers

The output reusability levels generated from the above neural network are evaluated using the confusion matrix [30] which is an important tool for evaluating the classification results as it presents the results in an easy to understand way. Section 5 presents the confusion matrix generated from the results of classification, along with the values for evaluation parameters: precision, recall, accuracy and classification error.

V. Discussions And Results

In this section we present the steps that were followed for the collection of relevant data and generation of results in our study. For the tool proposed, the implementation is done in Java using NetBeans IDE 7.0.1. and for input, data collection is done over 175 object oriented Java based code fragments. A GUI based interface is provided by the tool for user interaction. Users can browse through and select any Java based system as input for the tool. JRA^{2}M^2 extracts different object oriented code fragments from the input system and generates meta data after analysis based on the metric framework discussed in section 4. The generated metadata is used by an MLP based intelligence system for predicting the reusability levels of the identified components. The resulting predictions are evaluated using parameters like precision, recall, accuracy and error rate. In the following section we discuss the step by step results of data collection, processing and result generation at functional level using JRA^{2}M^2. Next section followed by a comparison of results obtained using the proposed approach with already existing reusability prediction models.

Step 1: Individual components (classes or code segments) are identified from the system provided as input by the user. The identified components are then analyzed by the Component extractor and analysis module of the metric framework of the JRA^{2}M^2. Fig 4 below presents the interface of the tool displaying the extracted components.

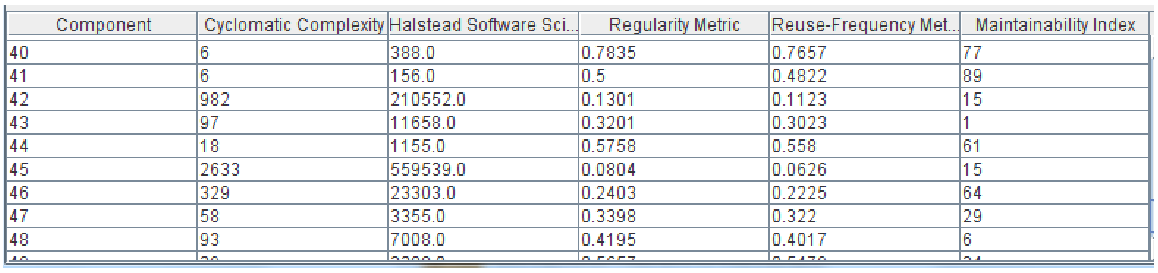


MysqlSavepoint.java
NamedPipeSocketFactory.java
NonRegisteringDriver.java
NonRegisteringReplicationDrive
NotUpdatable.java
OutputStreamWatcher.java
PacketTooBigException.java
PingTarget.java
PreparedStatement.java
ReplicationConnection.java
ReplicationDriver.java
ResultSet.java
ResultSetMetaData.java
RowData.java
RowDataDynamic.java
RowDataStatic.java

```
	 */
	public synchronized boolean wasNull() throws SQLException {
		return this.outputParamWasNull;
	}

	public int[] executeBatch() throws SQLException {
		if (this.hasOutputParams) {
			throw SQLError.createSQLException("Can't call execu
					SQLError.SQL_STAT
		}

		return super.executeBatch();
	}

	protected int getParameterIndexOffset() {
```

Fig. 4. Snapshot of interface of the tool displaying the extracted components

Step 2: Now the meta data is generated in the form of the values for the selected reusability metrics (as discussed in section 4) for all the identified components in the previous step. Fig 5 represents the meta data view generated for the identified components.



Component	Cyclomatic Complexity	Halstead Software Sci...	Regularity Metric	Reuse-Frequency Met...	Maintainability Index
40	6	388.0	0.7835	0.7657	77
41	6	156.0	0.5	0.4822	89
42	982	210552.0	0.1301	0.1123	15
43	97	11658.0	0.3201	0.3023	1
44	18	1155.0	0.5758	0.558	61
45	2633	559539.0	0.0804	0.0626	15
46	329	23303.0	0.2403	0.2225	64
47	58	3355.0	0.3398	0.322	29
48	93	7008.0	0.4195	0.4017	6

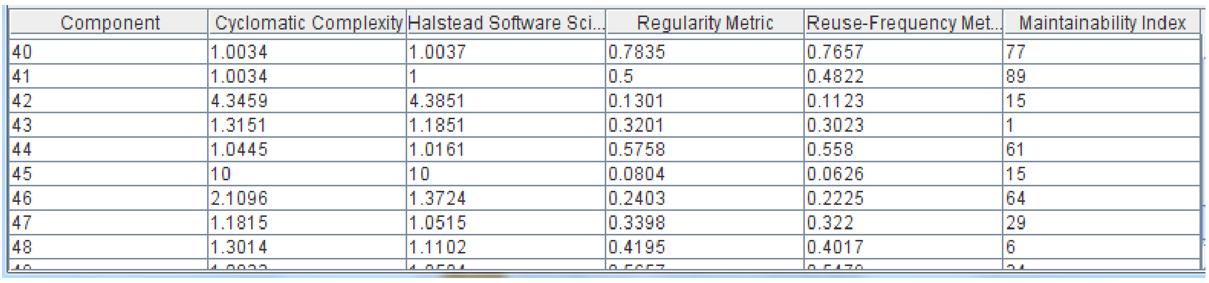
Fig. 5. Snapshot of meta data view generated for identified components.

In the next two steps, the meta data values generated in step 2 are pre processed such that they can be used for reusability prediction using the MLP based neural network approach.

Step 3: Firstly the meta data generated is normalized. Normalization need not be applied to metrics Reuse Frequency, Regularity Metric and Maintainability Index as already their values lie in specific ranges i.e. 0-1 for Reuse Frequency and Regularity metric and 0-100 for Maintainability Index (as can be seen from fig 5 above). Although the values of Cyclomatic Complexity and Halstead Program Volume do not fall in a pre defined range as they vary according to the size of the component. Hence we normalize these values to a range of 0-10. The normalized meta data view is presented in fig 6 .

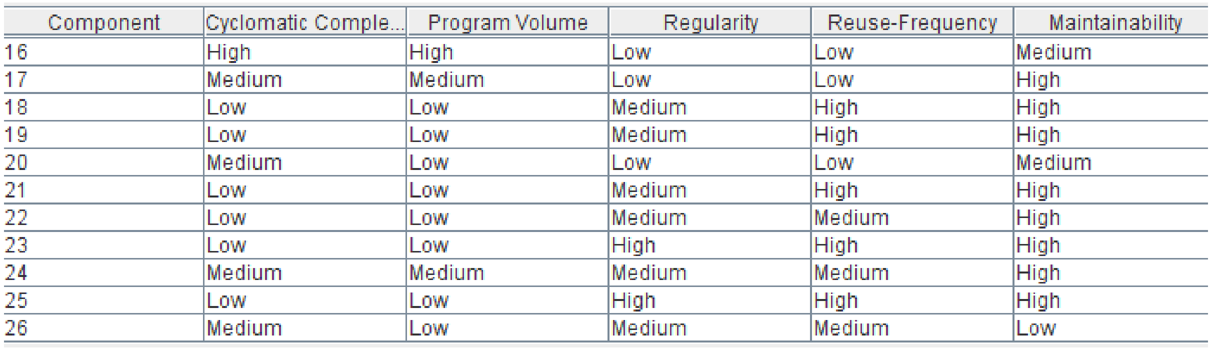
Step 4: After the metric values are normalized, they are raised in a hierarchial level of interpretation. The numeric metric values are transformed into categorical text (classes: low, medium or high) using [20] and [23]. This transformation is represented using fig 7 below. The reusability class labels are defined as low, medium or high based on the categorical metric values of the respective components.

Steps 2-4 are performed by the metric calculator module of the metric framework module of the proposed system JRA^{2}M^2.



Component	Cyclomatic Complexity	Halstead Software Sci...	Regularity Metric	Reuse-Frequency Met...	Maintainability Index
40	1.0034	1.0037	0.7835	0.7657	77
41	1.0034	1	0.5	0.4822	89
42	4.3459	4.3851	0.1301	0.1123	15
43	1.3151	1.1851	0.3201	0.3023	1
44	1.0445	1.0161	0.5758	0.558	61
45	10	10	0.0804	0.0626	15
46	2.1096	1.3724	0.2403	0.2225	64
47	1.1815	1.0515	0.3398	0.322	29
48	1.3014	1.1102	0.4195	0.4017	6

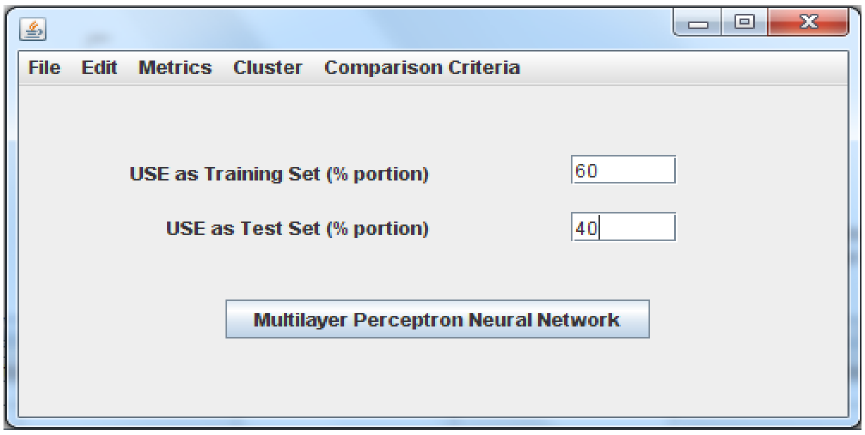
Fig. 6. Snapshot of normalized meta data view



Component	Cyclomatic Comple...	Program Volume	Regularity	Reuse-Frequency	Maintainability
16	High	High	Low	Low	Medium
17	Medium	Medium	Low	Low	High
18	Low	Low	Medium	High	High
19	Low	Low	Medium	High	High
20	Medium	Low	Low	Low	Medium
21	Low	Low	Medium	High	High
22	Low	Low	Medium	Medium	High
23	Low	Low	High	High	High
24	Medium	Medium	Medium	Medium	High
25	Low	Low	High	High	High
26	Medium	Low	Medium	Medium	Low

Fig. 7. Sanpshot of transformation into categorical text

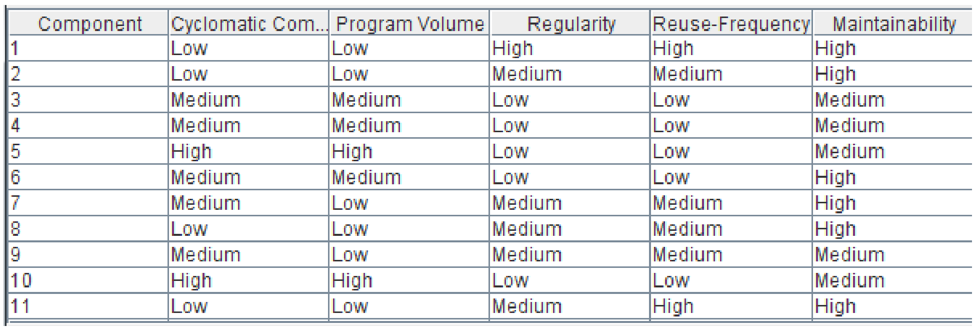
Step 5: For the establishment of meaningful relationships between the reusability metrics calculated in the previous steps JRA^{2}M^2 trains and develops a Multilayer perceptron based neural network using the generated metadata. The aim is to create a model that predicts the reusability values of a target variable (component) based on several input variables (metrics). We use MLP [29] algorithm in our model. The user is required to provide the percentage of inputs to be used as the training and test set as shown in fig 8 below. The instances of the training set are classified under three different class labels namely low reusability, medium reusability and high reusability, as per the combination of the metric values for the respective components. A snapshot of the extended reliability data set along with the enerated class labels is illustrated in fig 9.



USE as Training Set (% portion) 60

USE as Test Set (% portion) 40

Fig. 8. Sanpshot of percentage inputs to be used as training and test set

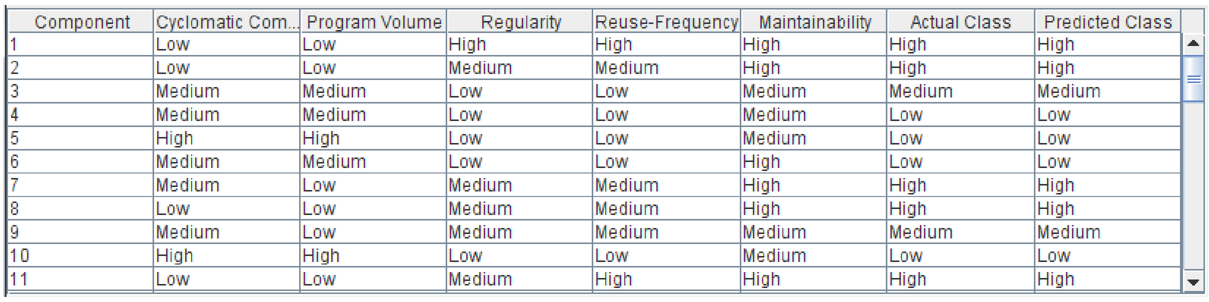


Component	Cyclomatic Com...	Program Volume	Regularity	Reuse-Frequency	Maintainability
1	Low	Low	High	High	High
2	Low	Low	Medium	Medium	High
3	Medium	Medium	Low	Low	Medium
4	Medium	Medium	Low	Low	Medium
5	High	High	Low	Low	Medium
6	Medium	Medium	Low	Low	High
7	Medium	Low	Medium	Medium	High
8	Low	Low	Medium	Medium	High
9	Medium	Low	Medium	Medium	Medium
10	High	High	Low	Low	Medium
11	Low	Low	Medium	High	High

Fig. 9. Snapshot of extended reliability data set along with the generated class labels

The multilayer perceptron model is trained using the instances from dataset presented in fig 9. The training is based on the error rate reduction using the backpropagation algorithm. Fig 10 shows the snapshot of JRA^{2}M^2 for test data result with actual and predicted classes and fig 11 presents the structure of the Multilayer Perceptron neural network model developed.

Step 5 and Step 6 are performed using the Neural Network Module of the proposed system model JRA^{2}M^2 for performance evaluation of the system.



Component	Cyclomatic Com...	Program Volume	Regularity	Reuse-Frequency	Maintainability	Actual Class	Predicted Class
1	Low	Low	High	High	High	High	High
2	Low	Low	Medium	Medium	High	High	High
3	Medium	Medium	Low	Low	Medium	Medium	Medium
4	Medium	Medium	Low	Low	Medium	Low	Low
5	High	High	Low	Low	Medium	Low	Low
6	Medium	Medium	Low	Low	High	Low	Low
7	Medium	Low	Medium	Medium	High	High	High
8	Low	Low	Medium	Medium	High	High	High
9	Medium	Low	Medium	Medium	Medium	Medium	Medium
10	High	High	Low	Low	Medium	Low	Low
11	Low	Low	Medium	High	High	High	High

Fig. 10. snapshot of JRA2M2 for test data result with actual and predicted classes.

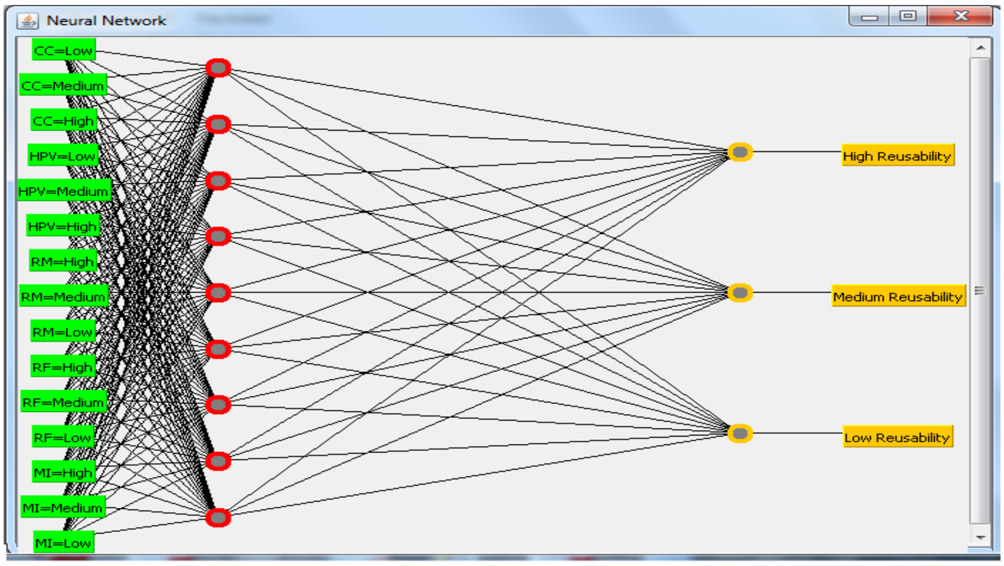


Fig. 11. Snapshot of structure of the Multilayer Perceptron neural network model

Step 6: The performance of the presents reusability evaluation model is assessed with the hel of confusion matrix for actual and predicted reusability classes for all the components. The confusion matrix hence obtained is resented in Table 3a. Based on the matrix parameters like precision and recall are calculated for the three reusability classes (low, medium and high) and accuracy and classification error of the system are also evaluated. The next Table 3b shows the values generated for these performance evaluation parameters.

Table 3a. Confusion matrix [30]

	Predicted Reusability Classes			
Actual Reusability Classes		Low	Medium	High
	Low	8	2	0
	Medium	1	3	1
	High	0	1	32

Table 3b. Values generated for performance evaluation parameters.

Reusability classes	Precision	Recall	Accuracy	Classification Error
Low	0.88	0.8	87.75%	12.25%
Medium	0.43	0.6		
High	0.96	0.94		

Using the reusability level prediction model presented here in our paper, the values of accuracy and classification error obtained (as can be seen from Table 3b) are 87.75% and 12.25% respectively. These values obtained clearly reflect the potential of the proposed automation tool/system JRA^{2}M^2, for detecting the components according to their correct reusability levels with a high probability. Hence ensuring that the proposed system is reliable enough for predicting reusability of software components, as it results in a low probability of false alarms (wrong usability level detection), thus avoiding the risks associated with wrong reuse (identifying a negligibly reusable component as reusable) thus preventing the wastage of time, effort and cost required in its reuse. Hence the developed tool JRA^{2}M^2 is a safe, reliable and effective approach for assessment and identification of reusable components from existing reservoirs for the development of reuse rpositories. Fig 12 below presents the number of components (used in our study, as per section 4) categorized under the three classes namely high, medium and low reusability.

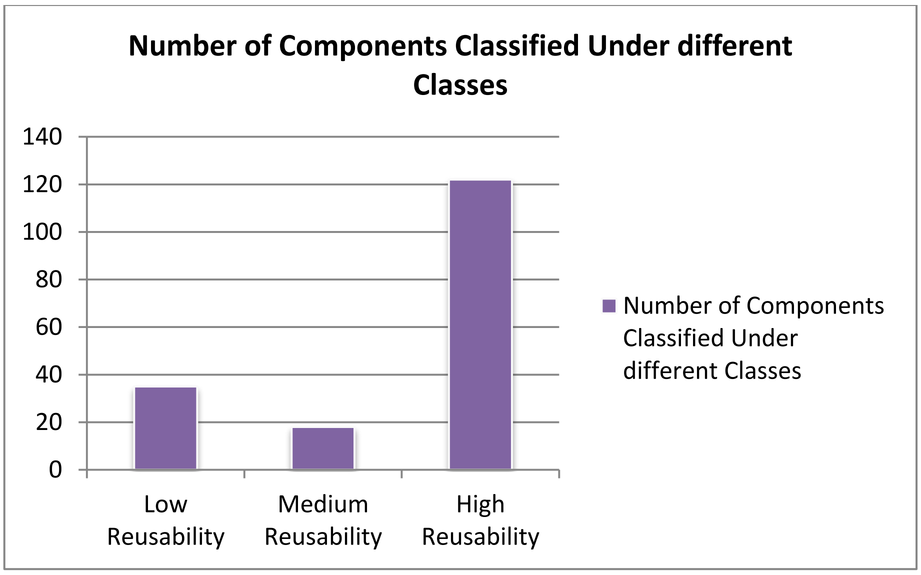


Fig. 12. Number of Components Classified Under different Classes

Hence it can be seen that the presented tool JRA^{2}M^2 can be used as a reliable, effective and efficient solution for reusability evaluation and prediction of function based object oriented software systems using Multilayer perceptron neural network, that provides for reduced error rates and better learning based on the feedback provided with the backpropagation learning algorithm. JRA^{2}M^2 not only estimates effective reuse of a component for complete or direct reuse but also provides reliable estimates for reuse after modification or partial reuse. Thus the presented evaluation model (JRA^{2}M^2) can be of immense potential use for software practitioners for assessing reusability levels and enhancing the software reuse repositories.

VI. Comparison With Other Existing Approaches

In this section the results generated by the presented tool JRA^{2}M^2 are compared with those of the other existing approaches for reusability prediction. Table 4 presents the comparison based on various criteria like tool support, approaches used, support for estimating partial reuse effort and performance parameters. Accuracy and Classification error rates for various approaches are compared using graph presented in fig 13a. Performance parameters include Precision, Recall values for each reusability class, low, medium and high for the techniques (presented graphically in fig 13b).

From the comparison presented using Table 4 and fig 13, it is clearly evident that the results generated by our proposed tool JRA^{2}M^2 are more reliable and promising as compared to those of the other existing techniques. With respect to techniques 1, 2 and 3, our tool JRA^{2}M^2 has higher accuracy levels and lower error rates. Technique 1 looks for structural aspects of reusability unlike our tool that works on functional basis for reusability evaluation. Techniques 2 and 3 although focus on the functional aspects, but do not provide support for the estimation of maintainability of the detected reusable components, which is an important aspect for reuse and has been incorporated in our approach via the use of Maintainability Index metric. Comparison graph from figure 13b reflects that JRA^{2}M^2 has better precision, recall results for high reusability class than those of technique 1, thus showing that highly reusable components are identified better by JRA^{2}M^2.

As compared to technique 3, the values of precision, recall for high reusability class may be almost equivalent to our approach, but the recall and precision values for low reusability level are zero, thus increasing the risk associated with wrong reuse (identifying non reusable components as reusable) with the use of technique 3. Hence, it can be concluded that JRA^{2}M^2 proves to be a more efficient and reliable solution as a reusability evaluation model.

Table 4. Comparision with other approaches

Attributes	Technique1[9]	Technique2 [13]	Technique3 [12]	JRA^{2}M^2
Tool Support	No	No	No	Yes
Metrics Used	Structural (CK Metric Suite [21])	Functional (CB Metric Suite [20])	Functional (CB Metric Suite [20])	Functional (CB Metric Suite [20] along with Maintainability Index)
Data Mining Approach	k-Means Hybrid	Hierarchical	DBSCAN	MLP Neural Network
Support for Estimating efforts of Partial Reuse / Reuse after modification	No	No	No	Yes (Using Maintainability Index)
RESULTS: Accuracy Classification Error	Figure 13 (a)	Figure 13 (a)	Figure 13 (a)	Figure 13 (a)
Precision Recall	Figure 13 (b)	NA	Figure 13 (b)	Figure 13 (b)

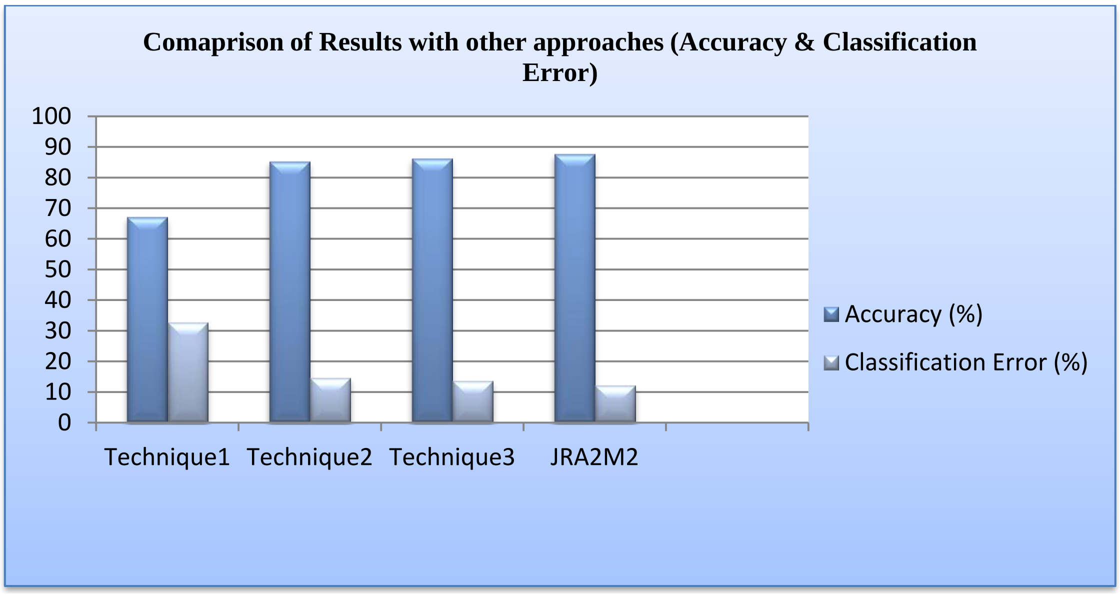


Fig. 13 a. Comaprison of Results with other approaches (Accuracy & Classification Error)

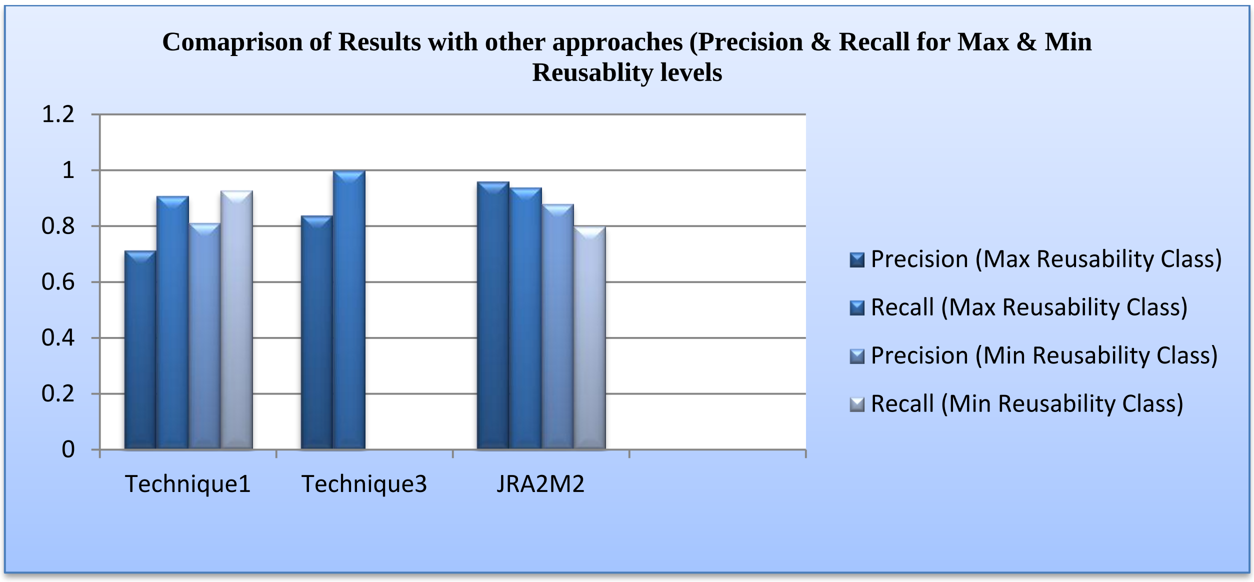


Fig. 13 b. Comaprison of Results with other approaches

VII. Significance/ Application

The presented automation system JRA^{2}M^2 effectively supports software practitioners, helping them increase the software development throughput by encouraging reuse of software components by providing them the knowledge about the levels of reusability of the already existing or newly developed software components. It can also help software development organizations apprehend time to market benefits for the "to be developed" software products. The acceptance of the approach presented in the industry can help software organizations strengthen their reuse repositories using the available inventories of resources while saving immensely on time, cost and effort. Also, it may lead to better reliability, maintainability, quality and effectiveness of newly developed software on the basis of better and safer reuse on account of its predictions.

VIII. Conclusion

In our paper a function based object oriented software metric framework is followed by an MLP based neural network training for the evaluation and assessment of reusability attributes of the input software components. The proposed model is implemented in Java with an automated and reliable tool support named using JRA^{2}M^2 NetBeans IDE. The metadata generated by the proposed metric framework (including support for partial reuse assessment via the Maintainability Index) is used to predict reliability by training and developing a multilayer perceptron based neural network that follows the backpropagation algorithm for error reduction. The model generates high precision and recall values for high reusability class, 96% and 94% respectively, thus helping in easy and efficient identification of highly reusable components. This implies 94% of them are identified and with 96% precision. Even for low reusability level the precision, recall values are good enough being 88% and 80% respectively, hence avoiding risks associated with wrong reuse. The precision and recall values for medium reusability class are 43% and 60% respectively. Performance results for JRA^{2}M^2 indicate how clearly it is able to differentiate between components belonging to different readability levels, especially between High and Low reusable components, at functional level. The overall accuracy of JRA^{2}M^2 is 87.75% with a classification error rate of 12.25%. Hence for such performance results of JRA^{2}M^2, it can be effectively used as an efficient and a reliable automation tool for identifying and evaluating reusable software components by software developers.

In future we plan to further extend the system JRA^{2}M^2 for identification or reusable components and their levels for more generic Object Oriented systems, i.e. independent of the language (presently Java). It can be extended to support other OO languages like Python, C++ etc. Also the application of the presented model can be investigated further in order to provide more precise levels of reusability classification by increasing the number of levels used for reusability classification.

References

[1] Mili H., Mili F., Mili A. (1995), "Reusing Software: Issues and Research Directions", IEEE Transactions on Software Engineering, Vol. 21, No. 6.

[2] Mcllroy D., (1968), "Mass Produced Software Components", Software Engineering Concepts and Techniques, 1968 NATO Conference on Software Engineering, pp. 88-98.

[3] Singh S., Singh S., Singh G. (2010), "Reusability of the Software", International Journal of Computer Applications (0975-8887), Vol. 7-No. 14.

[4] Sarbjeet Singh, Sukhvinder Singh, Gurpreet Singh, "Reusability of the Software", International Journal of Computer Applications (0975 – 8887)Volume 7– No.14, October 2010.

[5] J. F. Peters, W. Pedrycz, Software Engineering: An Engineering Approach, John Wiley & Sons, Inc., 2000. ISBN 0-471-18964-2.
[6] B.Jalender, Dr. A Govardhan, Dr. P Premchand, "A Pragmatic Approach to Software Reuse", Journal of Theoretical and Applied Information Technology.
[7] Jacob L. Cybulski, "Introduction to Software Reuse", Technical Report TR 96/4 The University of Melbourne Australia.
[8] Manhas S., Sandhu P.S., Chopra V., Neeru N. (2010), "Identification of Reusable software Modules in Function Oriented Software System using Neural Network Based Technique", World Academy of Science, Engineering and Technology, Vol. 67.
[9] Shri A., Sandhu P. S., Gupta V., Anand S. (2010), "Prediction of Reusability of Object Oriented Software System using clustering Approach", World Academy of Science, Engineering and Technology, Vol. 67, PP. 853-856.
[10] Ajay Kumar," Measuring Software reusability using SVM based classifier approach", International Journal of Information Technology and Knowledge Management, January-June 2012, Volume 5, No. 1, pp. 205-209.
[11] Amritpal Kaur, Rajbir Singh Cheema and Parvinder S. Sandhu, "Identification of Reusable Procedure Based Modules using kNN Approach", International Conference on Latest Computational Technologies (ICLCT'2012) March 17-18, 2012 Bangkok..
[12] Jagdeep Kaur Saini, Amitabh Sharma, Dr. Parvinder S. Sandhu, "Software Reusability Prediction using Density Based Clustering", 2006 "psrcentre.org".
[13] Czibula I. G., Serban G. (2007), "Heirarchial clustering for Software System Recnstructing", Babes bolyai University, Romania.
[14] Sandhu P. S., singh H. (2006), "A Reusability Evaluation Model for OO-Based software Components", International Journal of Electrical and Computer Engineering 1:4.
[15] Sandhu P. S., Singh J., Gupta V., Kaur M., Manhas S., Sidhu R. (2010), "A K-Means Based Clustering Approach for finding Faulty Modules in Open Source software Systems", World Academy of Science, Engineering and Technology, Vol. 72.
[16] Dr Himani Goel, Gurbhej Singh," Evaluation of Expectation Maximization based Clustering Approach for Reusability Prediction of Function based Software Systems", International Journal of Computer Applications (0975 – 8887) Volume 8– No.13, October 2010.
[17] Kanellopoulos Y., Dimopulos T., Tjortjis C., Makris C. (2006), "Mining source code Elements for Comprehending Object-Oriented systems and Evaluating Their Maintainability", SIGKDD Explorations, Vol. 8, Issue 1.
[18] Singh S., Singh P., Mohan N. (), "Identification of Object Oriented Reusable Components Using Multilayer Perceptron Based Approach", International Conference on Computer Engineering and Multimedia Technologies (ICCEMT'2012) September 8-9, 2012 Bangkok (Thailand).
[19] Sandhu P. S., Salaria D. S., Singh H. (2008), "A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation", World Academy of Science, Engineering and Technology 15 2008.
[20] Caldiera, Gianluigi and Victor R. Basili, "Identifying and Qualifying Reusable Software Components,," IEEE Software, Vol. 24, No. 2, February 1991, pp. 61-70.
[21] Chidamber, S.R., Kemereer C.F., "A metrics suite for object oriented design", IEEE Transactions on Software Engineering June 1994.
[22] Kurt D. Welker, "The Software Maintainability Index Revisited", The Journal of Defense Software Engineering august 2001.
[23] Code Metric Values," http://msdn.microsoft.com/en-us/library/bb385914.aspx.
[24] Thomas J. McCabe, "A Complexity Measure", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-2, NO.4, DECEMBER 1976.
[25] Halstead, M. H., Elements of Software Science, 1977, New York: Elsevier North-Holland.
[26] Hebb, Donald (1949), The Organization of Behavior. New York: Wiley.
[27] Rosenblatt, F. (1958). "The Perceptron: A Probalistic Model For Information Storage And Organization In The Brain". Psychological Review 65 (6): 386–408.
[28] Werbos, P.J. (1975), Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.
[29] Haykin, Simon (1998), Neural Networks: A Comprehensive Foundation (2 ed.), Prentice Hall. ISBN 0-13-273350-1.
[30] "Wikipedia - Confusion matrix", http://en.wikipedia.org/wiki/Confusion_matrix#cite_note-0.

Author's Profile:

Surbhi Maggo: She has done Masters of Technology and Bachelor of Technology from Jaypee Institute of Information Technology, India in Computer Science & Engineering and her area of interest is Software Engineering, Software Testing, Data Mining and Machine Learning. Currently she is working in a research and development company in India.

Chetna Gupta: She is Assistant Professor at Jaypee Institute of Information Technology, India. She obtained her Doctorate in the area of Software Testing. She also holds a Masters of Technology and a Bachelor of Engineering degree in Computer Science and Engineering. Her areas of interest are Software Engineering, Requirement Engineering, Software Testing, Software Project Management, Data Structures, Data Mining and Web Applications. She has many publications in international journals and conferences to her credit.