Volume 3, Issue 2, February 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Modelling of Reusability of Procedure Based Software Components with Improved Bayesian Classifier Approach

Priyanka Kakkar
Mtech(CSE)*
SSCET, Pathankot
India

Prof. Meenakshi Sharma
HOD(CSE)
SSCET, Pathankot
India

ABSTRACT: - We have developed a highly flexible module to evaluate and access the reusability of software components. The purpose of this model is to do pattern recognition by discovering supervised features which can help us to measure the intangible aspects of software components in terms of reusability. There were several function based applications which were given due diligence for identifying their various degrees of reusability of their components. Once these projects were analyzed their software components were measured in terms of software metrics including (Volume, Coupling, Complexity, Reuse frequency, Regularity and Reusability). These measured metrics were carefully allocated a particular set of label which was based on the principals of software engineering and objectives to be achieved for doing the due research. Therefore, in this research work we are studying the degree of reusability by using six classes with proposed method which was able to give high precision value as compare to previous methods.

Keywords: Software Reusability, Software Metrics, Reuse-Frequency Metric

1. INTRODUCTION

1.1 Software Reusability

Software reusability more specifically refers to design features of a software element (or collection of software elements) that enhance its suitability for reuse. In software engineering, reusability is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required. Reusability is often a required characteristic of platform software. Reusability brings several aspects to software development that does not need to be considered when reusability is not required.

For a software component to become reusable, it has to be generalized from the situation at hand, thoroughly documented and tested, incorporated in a library and classification scheme, and maintained as a separate entity.

The software industry is moving toward large-scale reuse, resulting in savings of time and money. To develop a new system from scratch is very costly. It is generally assumed that the reuse of existing software will enhance the reliability of a new software application. This concept is almost universally accepted because of the obvious fact that a product will work properly if it has already worked before.

A component can be considered an independent replaceable part of the application that provides a clear distinct function. A component can be a coherent package of software that can be independently developed and delivered as a unit, and that offers interfaces by which it can be connected unchanged with other components to compose a larger system.

Software Metrics

Metrics is defined as "The continuous application of measurement based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products". Software metrics is all about measurement and these are applicable to all the phases of software development life cycle from initiation to maintenance. The IEEE Standard Glossary defines metric as a "Quantitative measure of degree to which a system, component or process possess a given attribute". Quantitative measure here means that modeling components are open source projects. Metrics are designed to provide information about the software quality and allow easier detection of possible error or bad design. It can be static and dynamic, where static metrics are collected during the software code analysis while dynamic metrics are collected during execution. Software complexity metrics are often used as indirect metrics of reliability since they can be obtained relatively early in the software development life cycle. Using complexity metrics to identify components which likely contain faults allows software engineers to focus the verification effort on them, thus achieving a reliable product at a lower cost.[6]

Metrics are grouped into three main categories:

• Product metrics measure the software product properties, such as its documentation, design and performance, regardless of its development stage. Product metrics are indicator of external software attributes such as cyclometric complexity for testability and coupling factor for maintainability.

•Process metrics emphasize on the software development process, such as development time, methodology used and quality assurance techniques.

•Resource metrics emphasize on the human, hardware and software resources such as developer skill level, hardware reliability, software component quality.

Naïve Bayes Classifier

It is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable.

The Naïve Bayes Probabilistic Model:

Abstractly, the probability model for a classifier is a conditional model

$$p(C|F_1,\ldots,F_n)$$

over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.Using Bayes' theorem, we write

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

The proposed five metrics for function Oriented Paradigm is as follows:

The proposed metrics for Function Oriented Paradigm are as follows:

According to Mc Cabe, the value of Cyclometric Complexity (CC) can be obtained using the following equation:

1. Cyclometric Complexity Using Mc Cabe's Measure

$$CC = Number\ of\ \ predicate\ nodes + 1 \quad (1)$$

2. Software Science Indicator

According to this metric volume of the source code of the software component is expressed in the following equation:

$$Volume = N1 + N2\log 2(\eta 1 + \eta 2) \quad (2)$$

3. Regularity Metric

The notion behind Regularity is to predict length based on some regularity assumptions. As actual length (N) is sum of N1 and N2. The estimated length is shown in the following equation:

$$EstimatedLenght = N' = \eta 1\log 2\,\eta 1 + \eta 2\log 2\,\eta 2 \text{、} \quad (3)$$

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

$$\text{Re}\,gularity = 1 - \{(N - N')/N] = N'/N \quad (4)$$

4. Reuse-Frequency Metric

Reuse frequency is calculated by comparing number of static calls addressed to a component with number of calls addressed to the component whose reusability is to be measured. Let N user defined components be X_1, X_2 … X_N in the system, where S_1, S_2 … S_M are the standard environment components e.g. printf in C language, then Reuse-Frequency is calculated as:

$$Reuse - Frequency = \frac{\eta(C)}{\frac{1}{M}\sum_{i=0}^{M}\eta(S_i)} \quad (5)$$

5. Coupling Metric

Functions/methods that are loosely bound tend to be easier to remove and use in other contexts than those that depend heavily on other functions or non-local data. Different types of coupling effects reusability to different extent.

Table 1: Confusion Matrix of Prediction Outcomes.

	Real Data Value of Reusability	
Predicted Value of Reusability	**1**	**0**
1	**TP**	**FP**
0	**FN**	**TN**

With help of the confusion matrix values the precision and recall values are calculated described below:

1) Precision

The Precision is the proportion of the examples which truly have class x among all those which were classified as class x. The technique having maximum value of probability of detection and lower value of probability of false alarms is chosen as the best fault prediction technique.

Precision = TP / (TP + FP) (1)

2) Recall

Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been) [8]. The recall can be calculated as follows:

Recall=TP (TP+FN)

3) Accuracy

The percentage of the predicted values that match with the expected values of the reusability for the given data.

The best system is that having the high Accuracy, High Precision and High Recall value.

2. Methodology

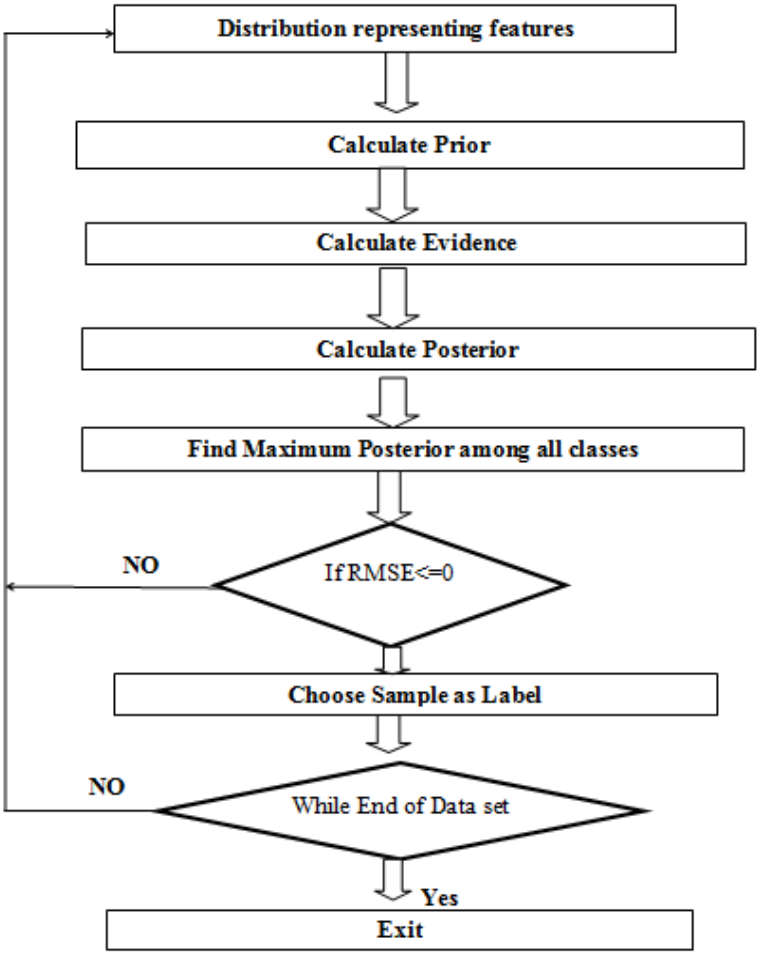

Figure1. : Methodology of Proposed Algorithm

3. Results and Discussion

The proposed methodology is implemented in NetBeans. The function oriented dataset considered have the output attribute as Reusability value.

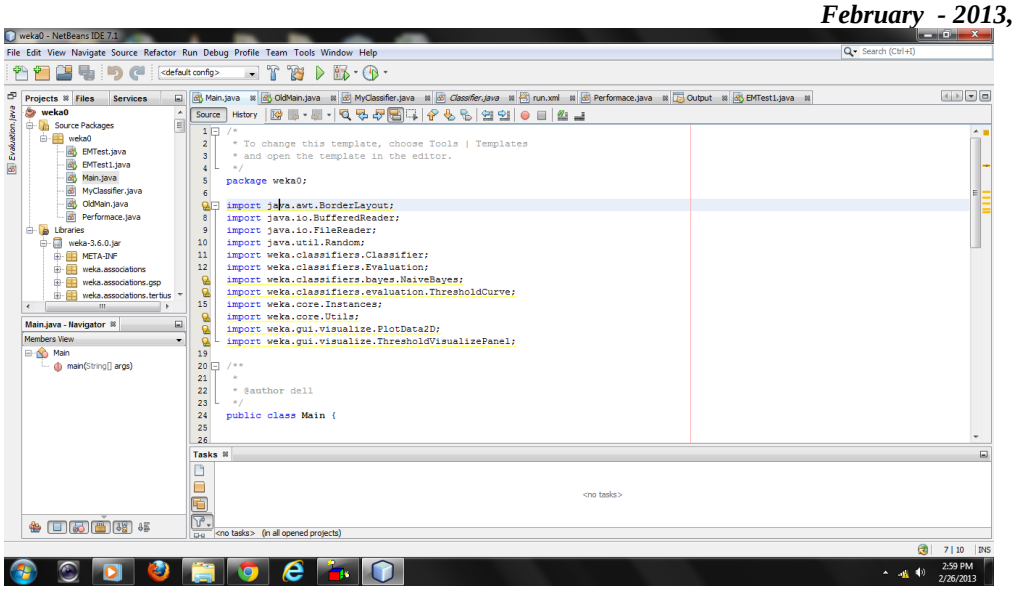

Figure 2: Snap shot of Net Beans to read the file

In the Figure 2 shows read the data from the source file. This is actual screenshot of the code which declare the file associated resource for the various output like Precision, Recall, True Positive Rate, F- measure, False Positive Rate etc. .

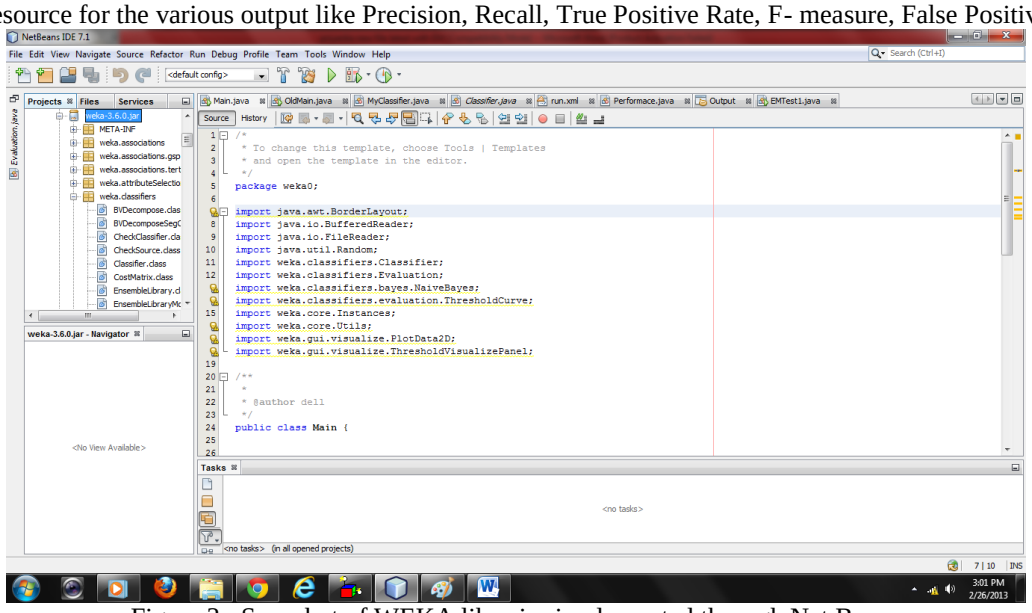

Figure 3.: Snapshot of WEKA libraries implemented through Net Beans

The above Figure 3 explains the library. The above figure contains the description of all implemented libraries and associated class.

Net Beans IDE is an open-source integrated development environment. Net Beans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box.

Output of proposed Algorithm:

When I compile and run the program using proposed classifier algorithm in Net Beans I find the various output like True Positive Rate, F- measure, True Negative, Recall , Precision etc as follows:

Table 2. : Detailed Accuracy By Class of proposed algorithm

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.2	0.01	0.5	0.2	0.286	0.961	1
0.778	0.063	0.538	0.778	0.636	0.958	2
0.462	0.064	0.706	0.462	0.558	0.78	3
0.862	0.187	0.641	0.862	0.735	0.831	4
0.647	0.011	0.917	0.647	0.759	0.806	5
1	0.035	0.857	1	0.923	0.969	6

Table 3.:Weighted Average Matrix of proposed algorithm

0.712	0.082	0.724	0.712	0.697	0.855

Table 4.: Confusion Matrix of proposed algorithm

A	B	C	D	E	F	<-- classified as
1	3	1	0	0	0	a=1
1	7	1	0	0	0	b=2
0	3	12	11	0	0	c=3
0	0	3	25	1	0	d=4
0	0	0	3	11	3	e=5
0	0	0	0	0	18	f=6

The other parameters calculated by the weka in the Net Beans using the proposed classifier are shown as under:-

Correctly Classified Instances	74	71.15%
Incorrectly Classified Instances	30	28.85%
Kappa statistic	0.6344	
Mean absolute error	0.1446	
Root mean squared error	0.2731	
Relative absolute error	54.44%	
Root relative squared error	75.01%	
Total Number of Instances	104	

Comparative Study :

Table 5: Precision Value of Different Classes of the Reusability Values of Previous Method and Proposed Method

Class	Precision(Proposed method)	Precision(Old method)
1	0.5	0.33
2	0.538	0.44
3	0.706	0.5
4	0.641	0.66
5	0.917	0
6	0.857	0.78

Table 5 shows the precision values of the different classes claculated by the EM Clustering and the proposed classification algorithm.

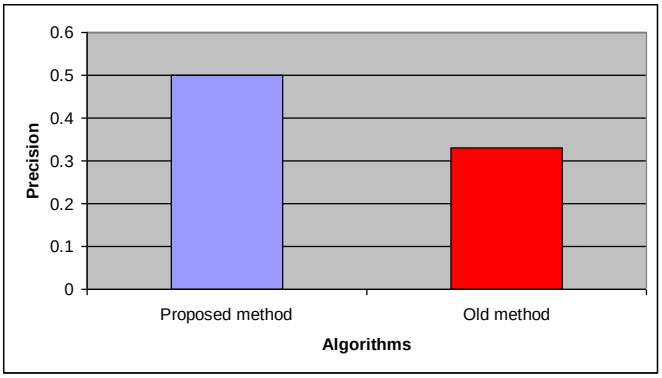

Figure 4: Comparative study of Reusability of class 1

It is apparent from the graph that the value of precision due to EM algorithm is quiet low as compare to the value of precision due to proposed algorithm . In this class the value of precision for proposed scheme is quiet relevant to the whole scenario due to their high precision value.

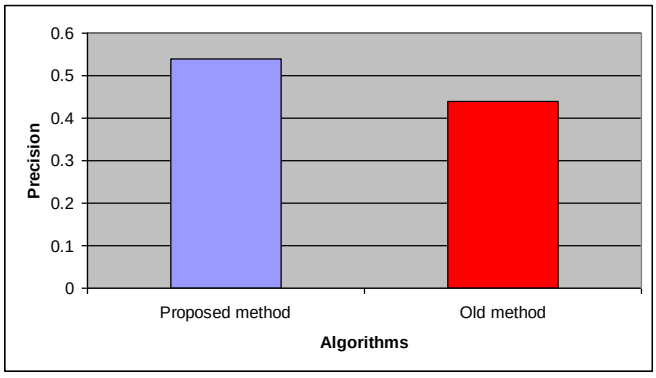

Figure 5: Comparative study of Reusability of class 2

Higher will be the value of class higher will be the precision value . This actually means with increase in the value in the class there will be the increase in the value of precision for both schemes.

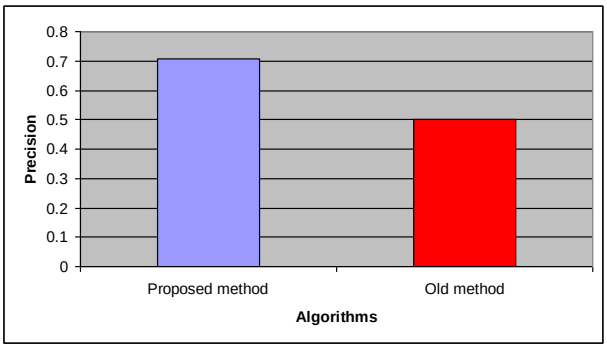

Figure 6: Comparative study of Reusability of class 3

It is apparent from the graph that the value of precision for EM algorithm is 0.5 which is far less as compare to propsed algorithm which is 0.706 in reusability level of class 3. So, the proposed algorithm classification based scenario is much better as compare to EM based cluster.

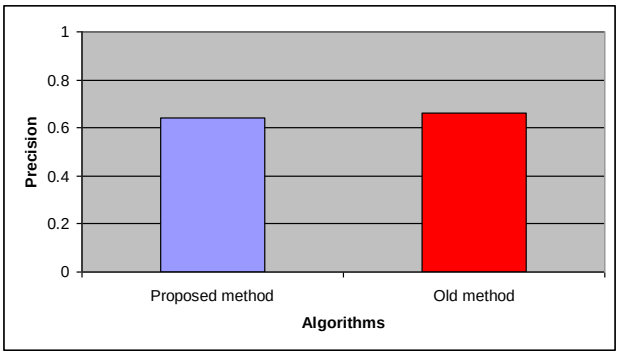

Figure 7: Comparative study of Reusability of class 4

In the above figure the value of precision of old scheme which is EM cluster is shown through pink bar and the value of precision of proposed scheme is shown. For class 4 , the value of old scheme is 0.66 and of proposed scheme is 0.641.

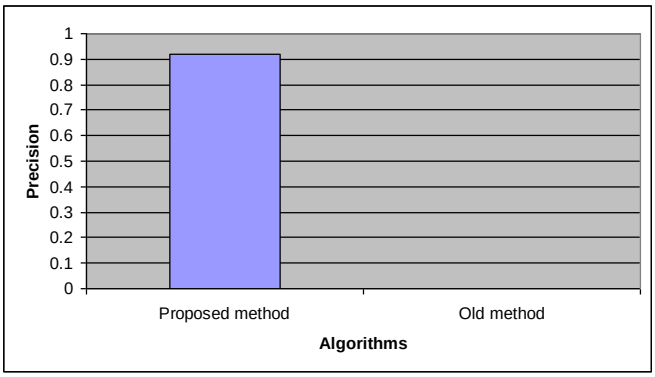

Figure 8: Comparative study of Reusability of class 5

It is apparent from the above figure 8 the Precision value of EM is 0 in the class 5 of reusability and the Precision value of proposed schema is also 0.917.

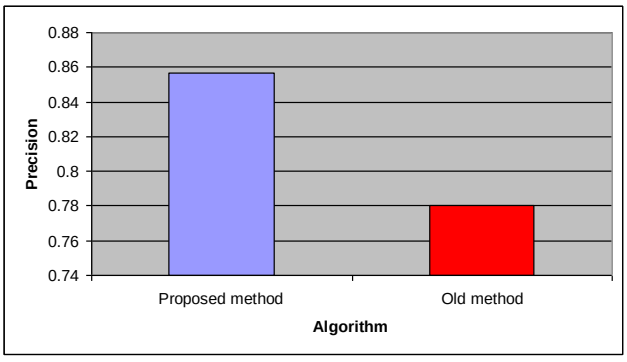

Figure 9: Comparative study of Reusability of class 6

It is clear that in the last class of reusability the value of Precision due to EM algorithm is 0.78 and of Proposed Algorithm Classification is 0.857.

In the last all six figure of Comparision , it is clear that the value of Precision through our Proposed Scheme Proposed Algorithm is higher for all class as compare to the value of Precision due to old scsheme EM based cluster.

Accuracy of EM Method and Proposed Method

Accuracy of EM Clustering Algorithm= 60%

Accuracy of the Proposed Algorithm= 71.15%

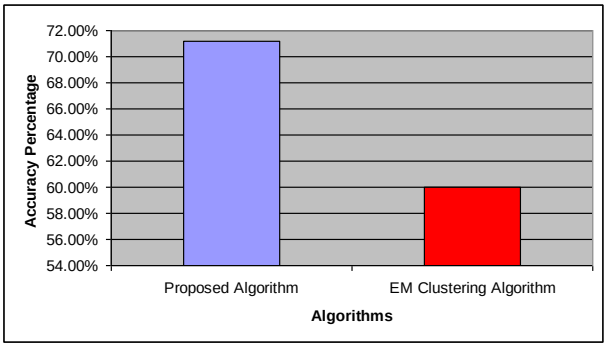

Figure 10: Comparative study of Accuracy of EM and Proposed Algorithm**.**

4. Conclusion

In this study, algorithm is proposed for evaluation of Reusability of Function based Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the six linguistic values. Six input metrics are used as Input and classifiers are formed using Naive Bayes algorithm, thereafter 10 fold cross validation performance of the system is recorded. As deduced from the results it is clear that Precision and Recall values of reusability class in the level, it means the system is able to detect the "Excellent" components precisely.

Measuring components is a challenging and costly process activity, and in its fullest definition it provides strong support for the development of high quality software. Existing maturity models do not adequately address the issue of component at coarse level. I have developed an assessment model to address these issues. Our research effort has resulted in an Assessment with five fully defined parameters of quality. Research can also be done on assessment model that will allow measuring some other parameter like compatibility. Finally, future research efforts must address the issue of components at fine grain level of function level component's assessment model can be created to measure the component quality at fine grain level of the functions /sub functions also.

The proposed technique is showing Accuracy value approximately equal to 71%, so it is satisfactory enough to use proposed algorithm classification technique for the prediction of the function based reusable modules from the existing reservoir of software components.

5. Future scope

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. So assessment of component at fine grain level (functions) can be achieved in future as further implementation of assessment model

- Intelligent Component Mining or Extraction algorithms can be developed
- Early prediction of the quality of component based system
- Characterization of Software Components for easy retrieval

REFERENCES

1. Basili, V. R. and Rombach, H. D. (1988) "The TAME Project: Towards Improvement Oriented Software Environments", IEEE Trans. Software Eng., vol. 14, no. 6, June 1988, pp. 758-771.
2. Selby, R. W. (1988) "Empirically Analyzing Software Reuse in a Production Environment", Software Reuse: Emerging Technology, W. Tracz, ed, IEEE Computer Society Press, 1988.
3. Basili, V.R. (1989) "Software Development: A Paradigm for the Future", Proceedings COMPAC'89, Los Alamitos, California, IEEE CS Press, 1989, pp. 471-485.
4. Arnold, R.S. (1990) "Salvaging Reusable Parts From Ada Code: A Progress Report", SPC Technical Report, SALVAGE_ADA_PARTS_PR-90048-N, September 1990.
5. Arnold, R.S. (1990) "Heuristics for Salvaging Reusable Parts From Adav Code", SPC Technical Report, ADA_REUSE_HEURISTICS-90011-N, March 1990.
6. Esteva, J. C. and Reynolds, R. G. (1991) "Identifying Reusable Components using Induction", International Journal of Software Engineering and Knowledge Engineering, Vol. 1, No. 3 , 1991, pp. 271-292
7. Mayobre, G. (1991) "Using Code Reusability Analysis to Identify Reusable Components from Software Related to an Application Domain," Proceeding of the Fourth Workshop on Software Reuse, Reston. VA, November, 1991, pp. 87-96.
8. Stender (1994) "Introduction to genetic algorithms", IEEE Colloquium on Genetic Algorithms, Volume 2, March 15, 1994 pp. 1-4.

9. Jang, J-S. R. and Sun, C.T. (1995) "Neuro-fuzzy Modeling and Control", Proceeding of IEEE, March 1995, pp. 123-135.
10. Klir, G. J. and Yuan, B. (1995) "Fuzzy Sets and Fuzzy Logic" Prentice-Hall, New Jersey.
11. Kartalopoulos, S. V. (1996) "Understanding Neural Networks and Fuzzy Logic-Basic Concepts and Applications", IEEE Press, 1996, pp. 153-160.
12. Jerome Feldman (1996) "Neural Networks - A Systematic Introduction" Berlin, New-York, 1996.
13. Succi, G., Benedicenti, L., and Vernazza, T., "Analysis of the Effects of Software Reuse on Customer Satisfaction in an RPG Environment", IEEE Trans. Software Eng., vol. 27, no. 5, May 2001, pp. 473-479.
14. Anderson, J.A (2003) "An Introduction To Neural Networks", Prentice Hall of India.
15. Frakes, W.B. and Kyo Kang (2005) "Software Reuse Research: Status and Future", IEEE Trans. Software Engineering, vol. 31, issue 7, July 2005, pp. 529 - 536.
16. Parvinder Singh and Hardeep Singh (2005) "Critical Suggestive Evaluation of CK METRIC", Proc. of 9th Pacific Asia Conference on Information Technology (PACIS-2005), Bangkok, Thailand, July 7 – 10, 2005, pp 234-241.

I.

