Software Quality Prediction using Hybrid Approach

Pragati Sharma
Dept of CSE
MSIT, GGSIPU
New Delhi

ABSTRACT
Quality of a software system depends on not only its functional but also its non-functional attributes. The prediction and determination of software quality of a component based system (CBS) becomes all the more important as the comprising components should be reusable. For that they must be reliable as well as reusable. Since quality is not something which is easily quantifiable it becomes a tedious task for conventional statistical models to predict software quality. Fuzzy logic can act as a great asset in these cases, where entities are closely related to the real world. An artificial neural network when combined with fuzzy inference system provided an architecture which can be trained and hence, is capable of predicting values. The said system has been employed for the purpose of quality prediction. Based on various factors several approaches have been proposed for determining and predicting software quality. But none of them use the combination of factors proposed in this paper.

Keywords
Component Based System (CBS), Software Quality, Fuzzy Logic

1. INTRODUCTION
In component based systems, the whole of as new systems is designed using predefined or already developed components .the use of pre-existing components make this technique better than any other technique. Components are basically reusable entities that can be used again and again for developing the new systems. Components are often called as black box, this is because we are not aware of the inner content of a component we are just concerned with its functionality. They are pretested and high quality entities. And also they are very restrictive in nature i.e. not compatible everywhere. Component based software engineering (CBSE) focusses on dividing a system into components, which are groups of code each of which provides a particular functionality. Using CBSE proves to be advantageous as it reduces the time and effort required to build a software from scratch. Hence promoting reusability. Also as these components are meant to be reused they are created in such a manner that they conform to the norms of reliability. Apart from that, many enterprises require software which is customized for their environment; therefore, customizability becomes a must for these components[1].

Mainly software engineering practitioners consider components as part of the platform for service orientation. Components play this role for example in web services and more recently in service oriented architectures(SOA) where a component is converted by web service into a service and subsequently inherits further characteristics beyond that of an ordinary component.

To estimate the quality of components, complexity, reusability, dependability, and maintainability are the key aspects. The overall quality of a system depends on the individual functionalities of these components. Therefore it is important to judge that which software will be good for use and which not. The phase of selection plays a major role. These aspects include main quality factors: complexity, coupling, cohesion, customizability, correctness and feasibility.

Soft computing is basically a technique use to find out unpredictable solutions .It will provide us with a range of values. Soft computing basically is different from the other used methods as it has high level of tolerance, imprecision and approximations. Neuro fuzzy system is a system that uses learned algorithm from neural networks to predict various parameters. A neuro-fuzzy system is as a 3-layer neural network. The first layer is for input variables, the middle (hidden) layer represents fuzzy rules and the third layer represents output variables. Fuzzy sets are encoded as (fuzzy) connection weights. The learning procedure of a neuro-fuzzy system takes the main properties of the fuzzy system into account. This results in restrictions on the possible modifications that are applicable to the system parameters. Neuro fuzzy system is used here to predict the range of values for our various factors. In this paper a method to predict the quality of software is proposed using fuzzy inference system and Halstead metrics

2. RELATED WORK
Various approaches to predict software quality have been proposed by researchers which can broadly be classified into three categories:

- Mathematical models
- Architecture based models
- Soft Computing based models

Architecture based models utilize a white box approach in order to take into account architecture of software using the consisting components and their interactions with each other. This can be classified into state based and path based approach. In state based models the control flow graph of the system is taken into consideration [2]. They consider current behaviour of the component to be independent of the component's past behaviour. In path based models existing execution paths are used to estimate software reliability [3] [4].

Another framework for estimating reliability is by using a component composition mechanism [5]. This approach introduces five component composition mechanisms. It also provides techniques to estimate their reliability. Reliability of the software is calculated by using "component composition mechanism and component utilization frequencies."

Soft Computing based approaches use fuzzy inference system alone or in combination with other techniques in order to predict software quality. This includes calculation of various factors determining the software quality, applying the

obtained result to a fuzzy inference system or neuro-fuzzy system [6]. In some cases quality is predicted by predicting the defect density in the code [7].

Bo Yang, Lan Yao, Hong-Zhong Huang used artificial neural network as well as fuzzy inference system in order to predict software quality during early stages of software development. They considered data gathered during the development process as well as expert opinion for this [8].

Mie Mie Thet Thwin, Tong-Seng Quah in their paper proposed a model of software quality prediction using neural networks. They used object oriented parameters for this purpose. Some metrics like cohesion, coupling, inheritance, and memory allocation [9].

Software quality prediction by determining the faults in legacy codes was done by Yuan, X., Khoshgoftaar, T.M., Allen, E.B., Ganesan, and K. They used fuzzy subtractive clustering in order to find the number of faults in the code. Then module ordering model was used in order to predict if a module is faulty or not [10].

K. Ganesan, Taghi M. Khoshgoftaar, And Edward B. Allen used case based software quality prediction. Case-based reasoning uses data from the past in order to predict the quality of the software. They are one the pioneers of software quality prediction using case based reasoning. [11].

3. FRAMEWORK FOR OUR PROPOSED SYSTEM

The paper proposes a Neuro-fuzzy reliability model wherein set of if-else statements are used .The statements become handy in devising a Fuzzy Inference System. The use of extensive software has been made to calculate the Quality of the code that is taken under consideration.

In Fig 1. Fuzzy System is explained; here Logical rules and If-then rules are applied on the Fuzzy sets so as to formulate the implementation rules.

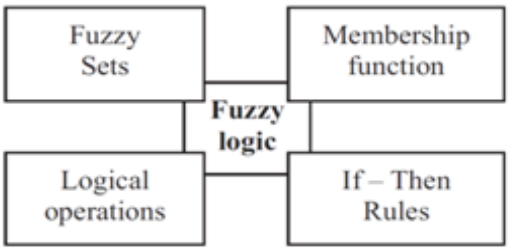

Figure 1: Fuzzy System

3.1 Factors for Quality Prediction

3.1.1 Cohesion

Cohesion tells us closely related a functionality is to other functionalities in the same module. It is more advisable to have modules with high cohesion. This is so because high cohesion implies reliability reusability etc. On the other hand low cohesion implies code which lacks ability to be comprehended, and is difficult to maintain.

Criterion for Selection
Cohesion is directly proportional to the quality of a component.

3.1.2 Coupling

Coupling is the degree of interrelationship between software modules or components. It is a measure of how closely connected two modules are. Low coupling is preferred over high coupling as low coupling as this means that a module or a component is complete on its own and hence it can be easily reused.

Criterion for Selection
Coupling is inversely proportional to software quality of a component.

3.1.3 Complexity

Complexity refers to difficulty in understanding with time and effort ultimately knowable. It describes the interactions between a numbers of entities. Complexity can be measured by various methods. Most easy and comprehensible way is using lines of code. Others include effort, number of errors, etc.

Criterion for Selection
As the complexity of the module increases, its quality decreases.

3.1.4 Feasibility

A feasibility study aims to reveal the strengths and weaknesses of the various business or proposed venture, opportunities and danger present in modules physically, logically and economically.

Criterion for Selection
Feasibility is directly proportional to software quality.

3.1.5 Customizability

Customization means the degree to which we can make changes in a module. It can be a user interface change or addition of another language or adding automation routines, etc. Customization is done to improve the software in order to satisfy your needs.

3.2 Halstead Metrics

The first step involves the use of Halstead Metrics for determination of software quality factors. Halstead Complexity measures were introduced for calculation of the various parameters that could help us judge the effectiveness of Software. The efficacy is can be amounted to the measurable properties of software and the relations between them .The various set of formulae used is stated below:

η_1 = the number of distinct operators
η_2 = the number of distinct operands
N_1 = the total number of operators
N_2 = the total number of operands
From these numbers, several measures can be calculated:

Program vocabulary: $\eta = \eta_1 + \eta_2$
Program length: $N = N_1 + N_2$
Calculated program length: $N = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$
Volume: $V = N \times \log_2 \eta$
Difficulty: $D = \frac{\eta 1}{2} \times \frac{N2}{\eta 2}$
Effort: E = D x V

Time required to program: $T = \frac{E}{18}$ seconds
Halstead's delivered bugs (B) is an estimate for the number of errors in the implementation.

3.3 Weka Tool

Once the factors are devised, they are fed into a spreadsheet which is then converted into a format suitable for the WEKA tool. WEKA is used in order to come out with the set of best possible Rules. For this, Apriori Algorithm is used on the data after the data is discretized. WEKA stands for Waikato Environment for Knowledge Analysis which is a Java software and is used for data mining so as to develop certain relationships and associations. Direct usages of dataset or directly loading java code are the two methodologies for analysis through WEKA. The 'dataset aspect' of it has been used. Pre-processing, classification, regression, clustering, association rules, and visualization are various aspects for which tools are available in WEKA. For ciphering the Association rules, Apriori Algorithm has been used. APRIORI ALGORITHM is an algorithm for study of association rules over transactional databases and learning of frequent item set mining. It progresses by combining smaller frequent item sets into larger and larger item sets as long as those item sets appear adequately in the database. General trends in database can be identified by using the frequent item sets determined by Apriori.

3.4 MATLAB

The Rules that are devised by the WEKA tool are put into a file that can be used by MATLAB in order to predict the Deviation of the Test data from the Training data that further helps in figuring out the appropriate Ranges for the Quality Factors**.** MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and a high-level language and multi-faceted environment used by millions of scientists and engineers worldwide. It lets us explore and visualize ideas and collaborate across disciplines including signal and image processing, communications, control systems, and computational finance.

3.5 Neural Network And Fuzzy Logic

Neural Network mimics the axon with dendrites where information is passed on from one axon to the other axon via the dendrites. Likewise, in a neural network input flows in through various sources so as to zero in on a particular output by propagation of only valid results from one set of conditions to another.

In Fig 2, Fuzzy logic is explained. It explains the fuzzification and defuzzification. The inference rules are laid down and the decision is made based on these if-then rules. The input values are defined and then output is obtained based on the FIS rules defined. [12]

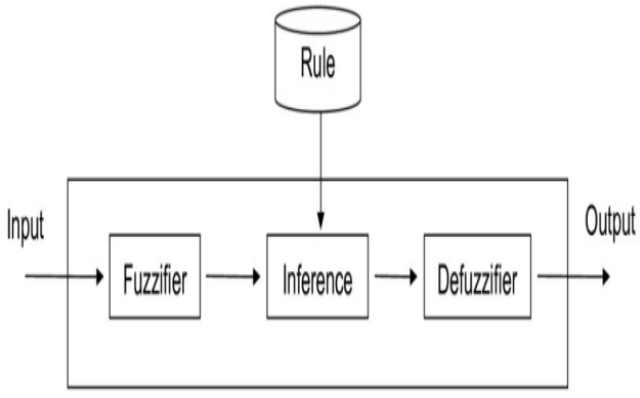

Figure 2: Fuzzy logic

3.6 ANFIS

Neuro Fuzzy approach as a cumulative one is precise and effective in calculating the Software Quality Prediction. As it amalgamates the Neural Network and Fuzzy principles, it combines the advantages of both the systems into a single framework which offers precision in decisions.

In the proposed approach the neuro-fuzzy system is build, trained and tested in MATLAB:

- Create and Build Fuzzy Inference System as FIS in MATLAB (See Figure 3). FIS contains the member functions for every input and output variable. Input variables are the pre-requisites that are required to reach to the output.

- Membership Function defines the variables for suitable ranges. (See Figure 4). A **membership function** (MF) is a curve that defines how each point in the input space is mapped to a **membership** value (or degree of **membership**) between 0 and 1. The **membership function** of a fuzzy set is a generalization of the indicator function in classical sets. In fuzzy logic, it represents the degree of truth as an extension of valuation.

- The training of the data is done with the help of the best found Rules obtained by the application of Apriori Algorithm. **Apriori** is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

- The FIS Rules are formed using the Association Rules which gave a definite result. On the basis of the Best found rules obtained from WEKA, FIS variables were selected against which the Final Rules were established (See Figure 5)

- Load the Training data in neuro-fuzzy system using MATLAB toolbox. Similarly, load the testing data as well (See Figure 6). Training data is used to train the FIS to generate valid outcomes by using the established set of rules.

- When training is completed, trained neuro-fuzzy system is validated with the testing data (See Figure 7). The intended outcome is to generate the desired outcome for the Testing data by using the Trained FIS which has rules and membership functions embedded in it.

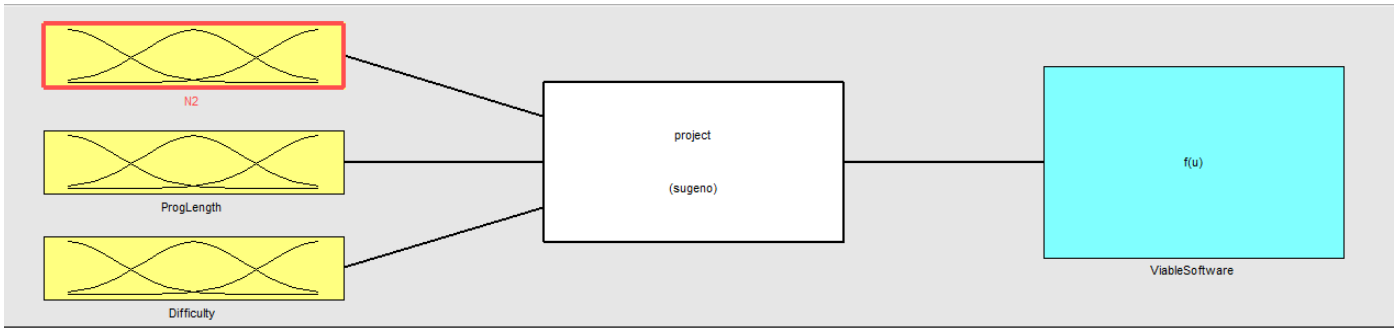

Figure 3: FIS system in MATLAB

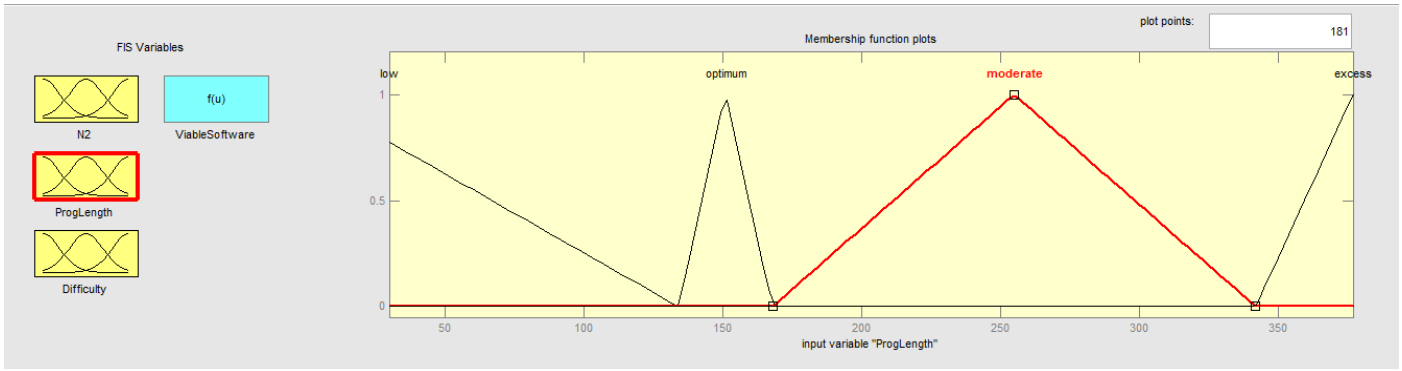

Figure 4: Member Functions

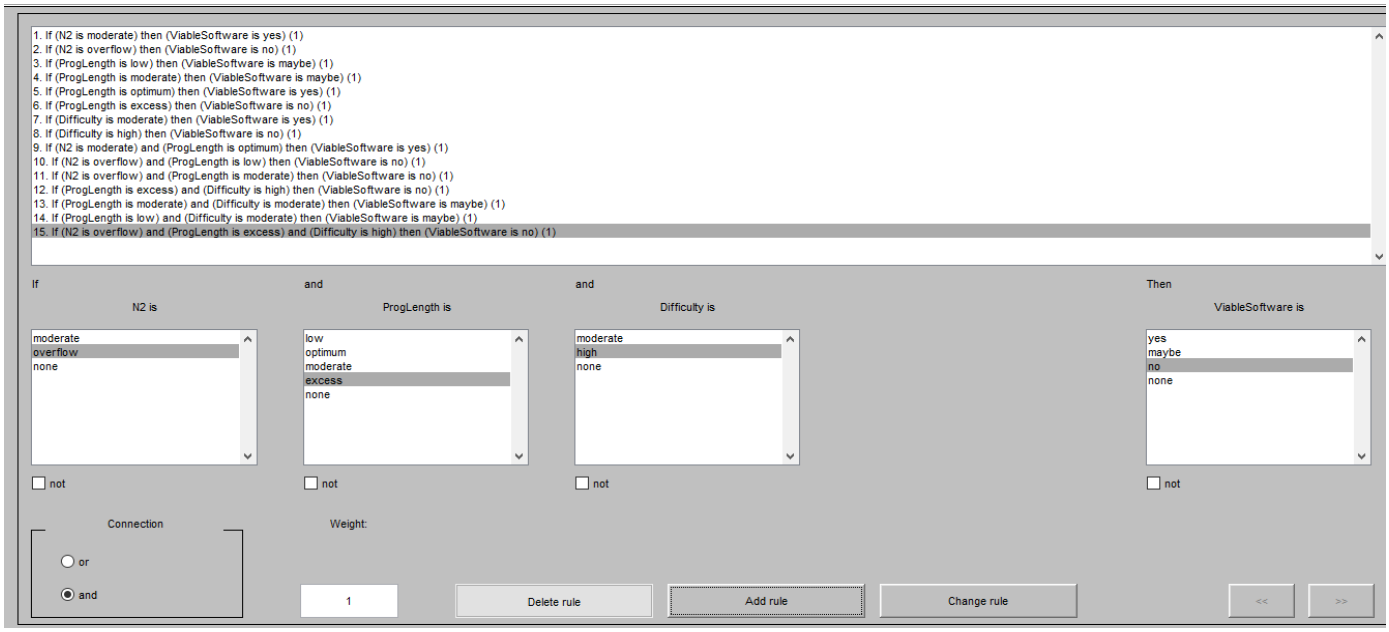

Figure 5: Rules obtained from Weka

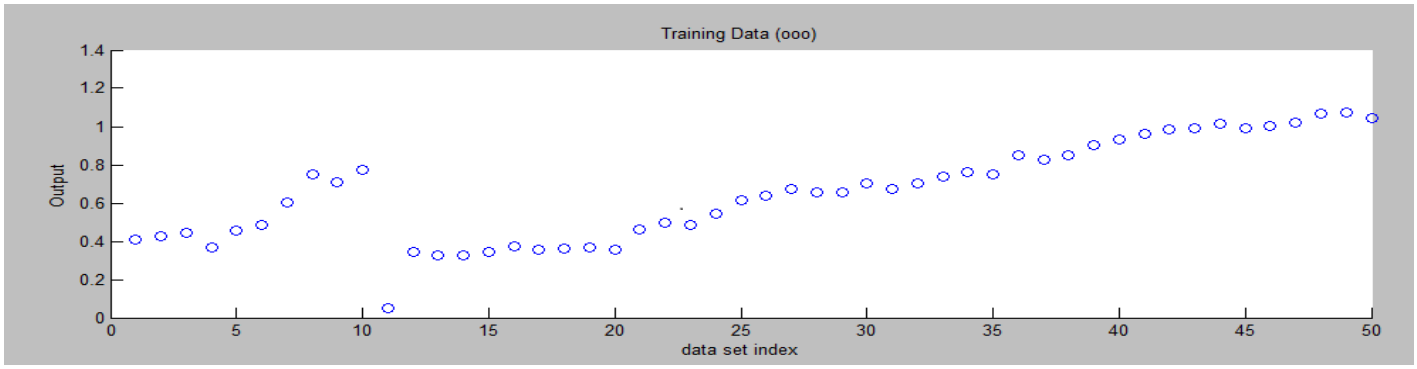

Figure 6: Training data in ANFIS

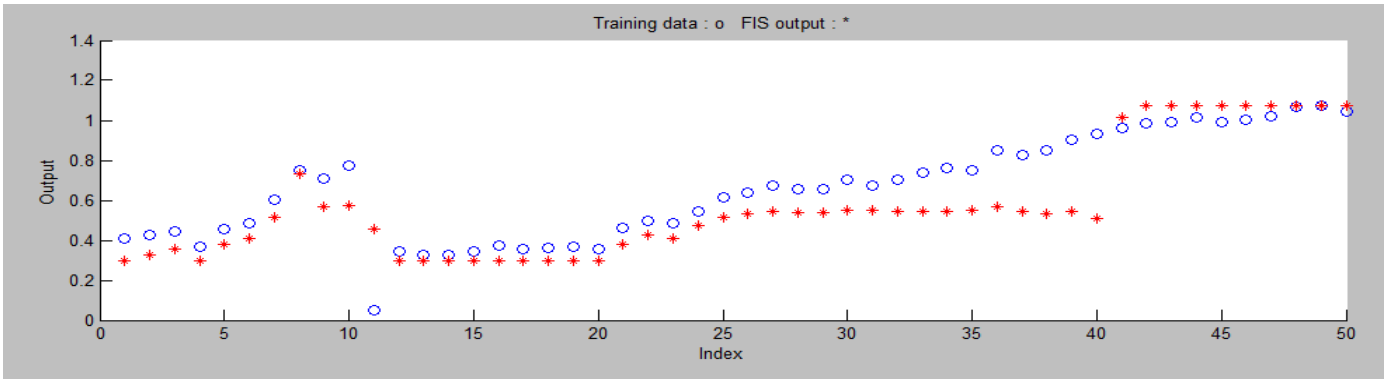

Figure 7: Plot of testing data (in red) against training data (in blue)

4. CONCLUSION

The present paper critically examined various metrics of five important quality factors: cohesion, coupling, complexity, feasibility and customizability for component-based systems. For characteristics like cohesion and coupling analytical method was used and for the rest a proper procedure using Halstead metrics was taken into consideration. The progress includes formulating methods for calculation of various software factors and devising a procedure for testing the input data against the training data. Creation of member functions was implemented for specifying the low, medium and high ranges for the various quality factors. The quality factors were measured on the sub-factors like Difficulty, Program length and Total no. of operands (N2). On the basis of the ranges decided and Rules established, the software's can be classified as whether they are fully viable, partially viable or not viable.

5. REFERENCES

[1] R. K. A. S. Vijai Kumar, Applying Neuro-fuzzy Approach to build the Reusability Assessment Framework across Software Component Releases - An Empirical Evaluation, International Journal of Computer Applications (0975 – 8887), May 2013.

[2] K. S. T. Katrina Goseva-Pospstojanova, "Architecture-based approach to reliability assessment of software systems," Performance Evaluation , vol. 45, pp. 179-204, 2001.

[3] M. Shooman, "Structural models for software reliability," in International Conference Software Engineering, 1976.

[4] A. M. S. Krishnamurthy, "On the estimation of reliability of a software system using reliabilities of its components," in Internatonal Symposiam Software Reliability Engineering, 1997.

[5] X. Y. X. W. C. H. Yuanjie Si, "Architecture based reliability estimation framework through component composition mechanism," in Computer Engineering and Technology, Chengdu, 2010.

[6] A. S. Kirti Tyagi, "An adaptive neuro fuzzy model for estimating their reliability of component-based software sytems," Applied Computing and Informaics, 2014.

[7] K. R. K. Vijai Kumar, "Appplying Soft Computing approaches to predict defect density in software product releases: An emperical study," Computing and Informatics, vol. 32, pp. 203-224, 2013.

[8] Yang, L. Yao and H.-Z. Huang, "Early Software Quality Prediction Based on a Fuzzy Neural Network Model," in Natural Computation, 2007. ICNC 2007. Third International Conference on (Volume:1), Haikou, 2007.

[9] T.-S. Q. Mie Mie Thet Thwin, "Application of neural networks for software quality prediction using object-oriented metricsApplication of neural networks for software quality prediction using object-oriented metrics," Journal of Systems and Software, vol. 76, no. 2, pp. 147-156, 2005.

[10] X. A. E. G. K. Yuan, "An application of fuzzy clustering to software quality prediction," in Application-Specific Systems and Software Engineering Technology, 2000. Proceedings 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, Richardson, TX, 2000.

[11] T. M. E. B. A. K. Ganeshan, "Case-based Software quality prediction," International Journal of Software Engineering and Knowledge Engineering, vol. 10, no. 2, 2000.

[12] L. a. H. H. B.Yang, "Early software quality pediction based on a Fuzzy Neural Network Model," in Natural Computation Third International Conference, Haikou, 2007.