Predicting Maintainability of Object-Oriented System using Fuzzy Logic

Palak Diwan
Galgotias University, Uttar Pradesh
Greater Noida, India
Email Id: dwnplk083@gmail.com

Abstract – With the increasing everyday demand of software the need of its maintenance has become pivotal. Maintainability, which is the system's ability to retain its original form and to restore back in case of failure, helps the new operator in easy understanding of the system. The Object-Oriented System is a popular system, based on the object-oriented paradigm which deals with objects and is used to make modular system. In this paper, a Fuzzy model has been proposed to obtain the empirical value of maintainability of object-oriented system. The model deals with crisp values and performs various operations to provide output as a single value.

Keywords – Object-Oriented System, Maintainability, Fuzzy logic.

I. INTRODUCTION

Software Engineering has become the core of our society by making us largely depend on its quality. McCall [1], Boehm [2], Dromey [3] have proposed various quality models, and each one of them suggests a different quality framework. The most followed model is ISO/IEC 9126 [4], which proposes six main factors that determine overall quality are maintainability, usability, efficiency, portability, functionality, and reliability. In our paper, the focus is on maintainability and its sub-characteristics like analyzability, testability, changeability, and stability.

In the evolution period, the software developed were procedure-oriented. They were difficult to maintain or modify, analyze, and lacked modularity. Therefore, another programming paradigm was introduced and named as an object-oriented system (OOP). The main features of OOP that make it better that procedure oriented systems are inheritance, polymorphism, data hiding, and encapsulation [5]. It overcame many shortcomings of procedure-oriented systems and had advantages in software reusability, portability, and maintainability. In further research, [6] [7] discovered that modularity and self- descriptiveness influence maintainability prominently. Maintainability of an object-oriented system can be assessed with either internal characteristics like cohesion, size, and coupling or on the basis of external characteristics such as analyzability, changeability, testability, stability, modularity, and self-descriptiveness.

Evaluating maintainability is challenging because the factors governing the system do not have fixed values; hence, a large number of techniques are required to calculate results. As factors vary over a wide range, we need a technique to deal with vague values rather than crisp values. Therefore, a soft computing technique called fuzzy logic [8] was proposed to assess maintainability. It processes vague input data and maps them into fuzzy sets, which are then aggregated to produce a single output.

In this paper, all the six factors: changeability, analyzability, stability, testability, modularity, and self-descriptiveness are considered to evaluate maintainability. To evaluate input factors, different metrics from many papers is used and is automated using a fuzzy logic toolbox of Matlab [9].

Section 2 details the work of various practitioners in related areas. In Section 3 describes an object-oriented system and Section 4 explains ISO/IEC 9126 and factors of maintainability. Section 6 shows the fuzzy logic approach to a system and the steps involved in the evaluation of maintainability. Section 7 describes the model with input parameters and their membership functions. In section 8 concludes the work.

II. RELATED WORK

Maintainability has developed tremendously from the initial days and several studies have been conducted for its effective evaluation.

According to Goel et al. [10], factors affecting maintainability hierarchy are analyzability, changeability, stability, and testability. These factors vary with design complexity metrics like NOC, CBO, DIT, WMC and RFC [11]. Ghosh et al. [12] discussed the sub-factors of maintainability namely size, complexity, coupling, and inheritance. NOC, CBO_NA, CBO_IUB, and CBO_U are taken as metrics for design description and changeability prediction in [13]. Kumar et al. [14] developed a three factor

model where complexity varies with complexity of attributes (CMPAt), complexity of operations (CMPOp), and complexity of nested components (CMPNested). Dubey et al. [22] suggested a framework where internal quality attributes of size, complexity, coupling, and inheritance are used as quality determining factors that involves object oriented metrics such as LOC, WMC, DIT, and CBO.

Viser et al. [16] added another factor as maintainability compliance and suggested that all factors of maintainability varies with volume, complexity per unit, duplication, unit size, and unit testing. According to Bruntink et al. [17], testability can be measured by comparing the java class C with the test class C_t and provided experimental design for the evaluation of testability.

All of the them considered four basic features for maintainability suggested by ISO/IEC 9126, but none of them considered the two features that have enormous influence on maintainability are modularity and self- descriptiveness. In the paper, a generic model is developed to include modularity and self-descriptiveness with already developed factors of maintainability.

III. OBJECT-ORIENTED SYSTEM

Object-oriented systems are rapidly replacing procedure-oriented systems as the latter are less maintainable and non-reusable. It is meant to address the difficulties with procedural programming with its aim to try and increase the flexibility and maintainability of programs. The design cycle of software partitions the problem to communicating entities called objects, which are called as the instance of class [18]. Objects are the basic unit through which we can access methods and variables of class and operate on it. Each class is a structure which consists of variables and methods used to operate on data of class. This system uses an approach to design modular and reusable software.

The goals of object-oriented programming are ease of maintenance, increased understanding, and ease of evolution. The features of OOP that make it maintainable are polymorphism, inheritance, data hiding, and encapsulation. Other two important features are coupling and cohesion. A well designed system should maximize cohesion and minimize coupling.

A software can be measured using software metrics [19], which are used to plan and predict its development. It states that software metrics are divided into two parts: software product metrics e.g. size and software process metrics e.g. efforts which are required to design a software system. It can also be used to evaluate software maintenance efforts.

According to Chidamber et al. [11], there are some object-oriented metrics like WMC (weighted methods per class), DIT (depth of inheritance), CBO (coupling between objects), NOC (number of children), RFC (response for a class), and LCOM (Lack of Cohesion of Methods) for internal quality attributes of a software.

Ghosh Soumi et al. [13] introduced one more metric called as LOC (line of code). It is the simplest metric used to compare different projects. It is referred to as lines of program excluding blank line or comments.

IV. SOFTWARE QUALITY

Many models have been developed to determine how various quality characteristics contribute towards software quality. These models are used to associate external view of users to the internal view of developers for the software products. Some of the popular quality models:

1. McCall Model

Jim McCall [1] proposed a software quality model in 1977 and classified the framework in three perspectives: Product Operation Factors, Product Revision Factors, and Product Transition Factors.

2. Boehm Model

It was introduced by Barry W. Boehm [2] in 1978. This model is similar to McCall Model with a difference that it structured the quality characteristics in a hierarchal tree and divided characteristics into further sub-divisions.

3. Dromey Model

The model was proposed by R. Geoff Dromey [3] in 1995, and it followed different approach to evaluate software quality than McCall's and Boehm's model. He focused on the relationship between the quality attributes and sub- attributes as well as tried to connect software quality attributes with product attributes.

4. FURPS+ Model

FURPS+ Model was proposed by Robert Grady at HP. FURPS. FURPS is expanded as Functionality, Usability, Reliability, Performance and Stability. He divided characteristics into 2 parts as functional requirements and non-functional requirements. The "+" in the FURPS+ acronym allow us to specify constraints including physical, interface, design and implementation.

5. IEEE

IEEE has released several standards. Few of the prominent ones are IEEE Standard 1063-2001, IEEE Standard 1028-1997, IEEE Standard 730-1998, and IEEE/IEC 12207:1995-(ISO/IEC 12207).

Out of all, IEEE standard resembles the most with other models and describes the processes like primary processes, supporting processes, and organization processes for the software life cycle

6. ISO/IEC 9126 [4] Software Quality Model

It is an international standard which is divided into four parts:

- ☐ ISO/IEC 9126-1 – Software quality model
- ☐ ISO/IEC 9126-2 – External Metrics
- ☐ ISO/IEC 9126-3 - Internal metrics

- ISO/IEC 9126-4 - Quality in use metrics

The characteristics of this model are functionality, reliability, usability, efficiency, maintainability, and portability. These are further characterized into 27 sub-characteristics.

V. MAINTAINABILITY

According to IEEE standard glossary of software engineering, maintainability is

"The ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment"

According to Hasia et al. [20], maintainability is greatly affected by the object-oriented approach of a system. A system is easily maintainable if at the design phase [21], an object-oriented system is decomposed into subjects and each subject is further decomposed into sub-systems. Thus, a system is said to be maintainable if the change affects less number of subsystems.

After analyzing different software metrics from several researches, six sub-characteristics are identified which can be used to estimate maintainability of software:

1. Analyzability: It can be evaluated by measuring the number of components and their relative size. A metric called Component Balance [15] was taken into account to calculate both the factors.
   It can be represented as:
   $$CB(S) = SB(|C|) * CSU(C)$$
   CB refers to component balance, SB refers to system breakdown and CSU refers to component size uniformity. Higher value of CB represents a better component decomposition, in turn, more analyzable.
1. Changeability: Kabali et al. [13] described changeability metrics like NOC (number of children), CBO_NA (CBO no ancestors), CBO_IUB (CBO is used by) and CBO_U (CBO using). The impact of change in one class to another class is calculated as:
   Impact (cl_i, ch_j) = F (S, G, H, I, L)
   Impact (cl_i, ch_j) means impact of ch_j class to ch_i class and S (association), G (aggregation), H (inheritance), I (invocation), and L (pseudo-link) are the links used to connect classes.
2. Stability: It is the property of a system to remain unaffected even after changes applied to it. Viser et al. [16] referred stability as the capability of a system to remain in a consistent state even after modification.
3. Testability: According to Bruntink et al. [17], software testability is affected by many different factors including the required validity, the process, and the tools used for the representation of requirements. Test class C_t is used to test class C by comparison.
4. Modularity: It means the decomposition of program into modules; thus, making the program more understandable.
5. Self-descriptiveness: It refers to the feature which enhances the readability of software by documentation and comments. It is used to reduce user's memory load.

VI. FUZZY LOGIC

Fuzzy logic is a mathematical tool for tackling the problem of uncertainty. It can be seen in two different perspectives. In a narrow sense, it is considered as a multivalued logic system and in a wider sense, it is the synonym for fuzzy sets [22].

In fuzzy logic tool, values are given as input and output is calculated by using a set of rules defined in rule base and fuzzy operators. The fuzzy inference system is shown in fig 1. Fuzzy inference process comprises of five parts:

- Fuzzification of input variables:
  In this step, the crisp values are taken as input and the degree by which they belong to a particular fuzzy sets is evaluated.
- Apply Fuzzy Operators:
  Now, we know the degree to which antecedent each rule belongs to. If the given rule has more than one part in the antecedent, fuzzy operator is applied to obtain one number that represents the result for that rule. This number is then applied to output function to obtain results.
- Apply Implication Method
  First, the weight for each rule is to be determined which is generally 1 then the implication method is applied. The input for the implication is a single number given by the antecedent of rule and output obtained is fuzzy set. Two built-in implication methods are min for truncation and prod for scaling.
- Aggregate all outputs:
  It is the process by which the fuzzy sets that represent the outputs are combined into a single fuzzy set.

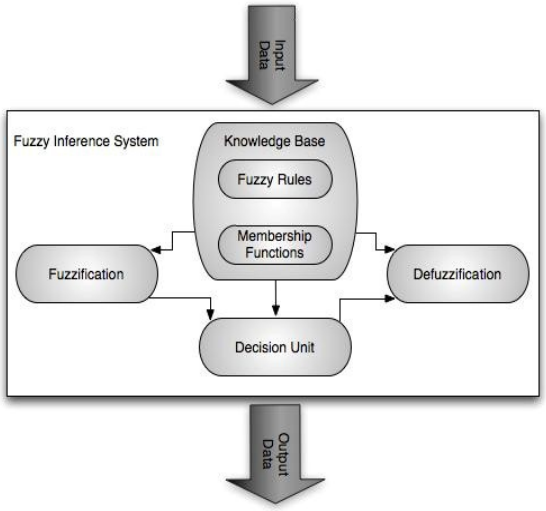

Fig. 1: Flow Diagram of input and output

- De-fuzzification:
  It is the reverse of fuzzification in which inputs are fuzzy sets and converted into crisp values to produce output

VII. PROPOSED FUZZY MODEL

As discussed above, there are several ways to evaluate the maintainability, but fuzzy logic proved to be better to obtain the exact value of the output [22]. Fuzzy model deals with fuzzy values rather than binary values and works even with small amount of data. Fig 2 shows the processes followed in fuzzy logic.

Fuzzy model can be implemented using Fuzzy Inference System (FIS) and the most commonly used FIS is Mamdani fuzzy inference method [22] as shown in fig 3.

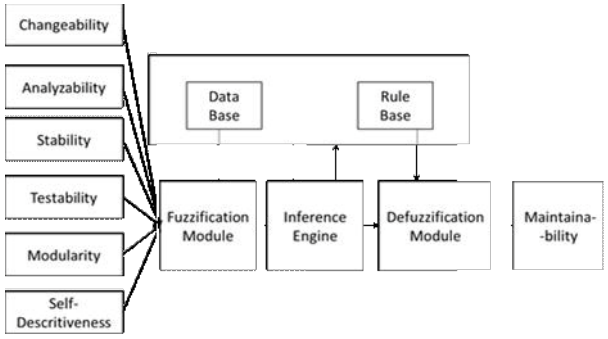

Fig 2: Process followed in Fuzzy Logic

According to process, input factors are taken as analyzability, changeability, stability, testability, modularity, and self-descriptiveness and output factor evaluated is maintainability.

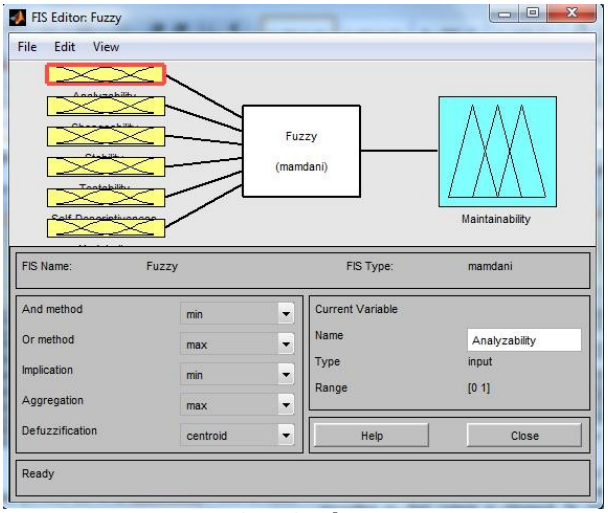

Fig 3: FIS Editor

In this process, fuzzification is done. To perform this, all the inputs are divided into three ranges of values as Low, Medium and High as shown in fig 4.

There are a number of Membership Functions (MF), but here, we are using triangular membership function to make model simple, and there is a little overlapping between these MF's.

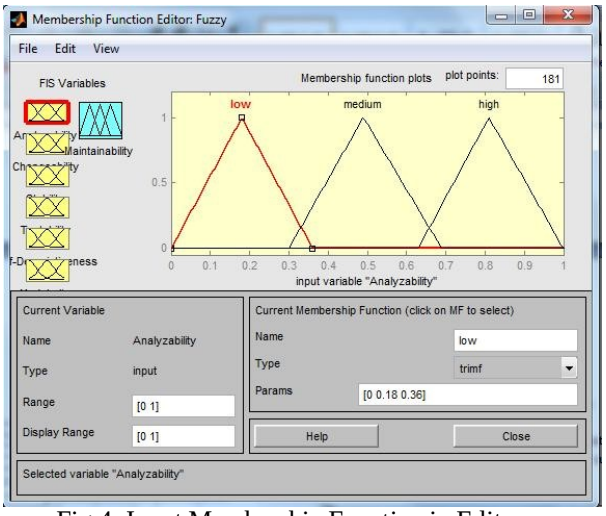

Fig 4: Input Membership Function in Editor

In the model, five membership functions as very low, low, medium, high, and very high are used to describe the range of for output as shown in the Fig 5.

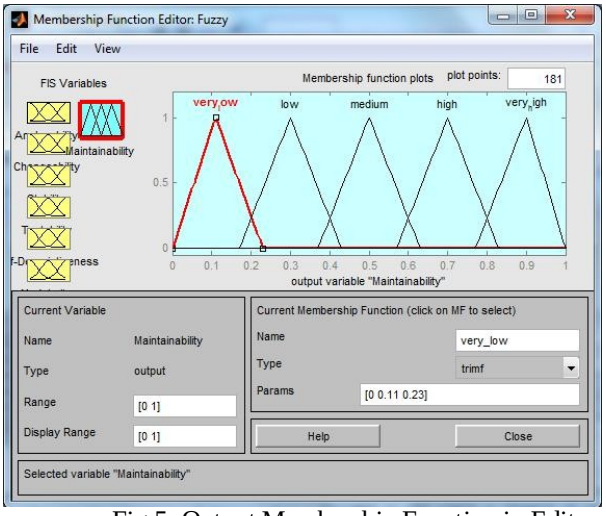

Fig 5: Output Membership Function in Editor

After that, rules are made in rule editor using fuzzy operator for each possible combination of inputs and output is obtained accordingly. In order to formulate rules, we conducted survey from 18 experts. Out of these experts, 9 are working in software industries and remaining 9 are PhD scholars. On the basis of opinion from experts, we formulated rule base. Here, the operator used is and. A total of 729 rules are made and it can be checked using formula n^m, where n is the number of membership functions used for inputs and m be the number of inputs [8]. A screenshot of rule editor is shown in fig 6

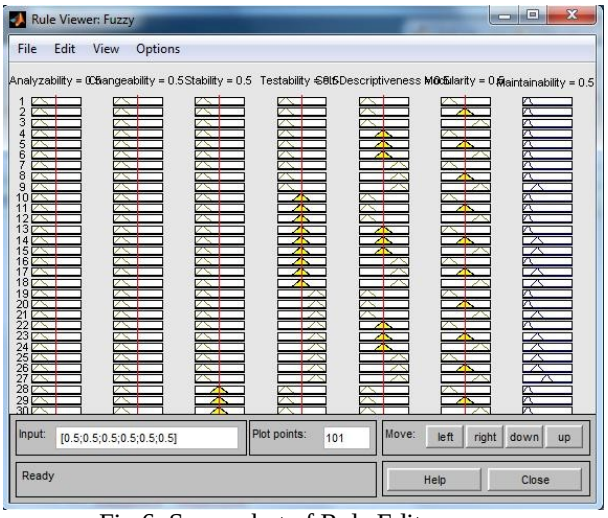

Fig 6: Screenshot of Rule Editor

After formulating all the rules, the implication method is applied. The most commonly used implication method is centroid method to calculate the aggregate of all the inputs. In this model, we have taken the range of inputs and output as 0-1. After that, de-fuzzification process is applied to obtain the final output as a crisp value.

Now, the variation of output with respect to input can be observed using rule and surface viewer. A screenshot of both is shown in Fig 7 and Fig 8 respectively.

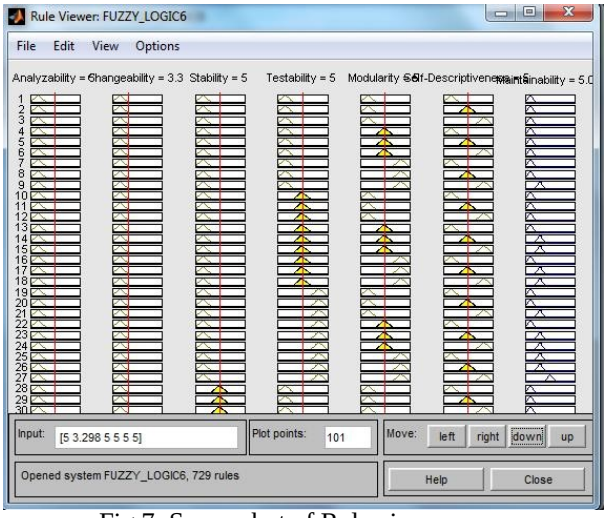

Fig 7: Screenshot of Rule viewer

From the rule viewer, it is seen that when values of analyzability, changeability, stability, testability, modularity and self-descriptiveness are 0.5, 0.5, 0.5, 0.5 and 0.5 respectively then Maintainability is 0.5, same can be checked from command prompt.

Surface view of the rules made:

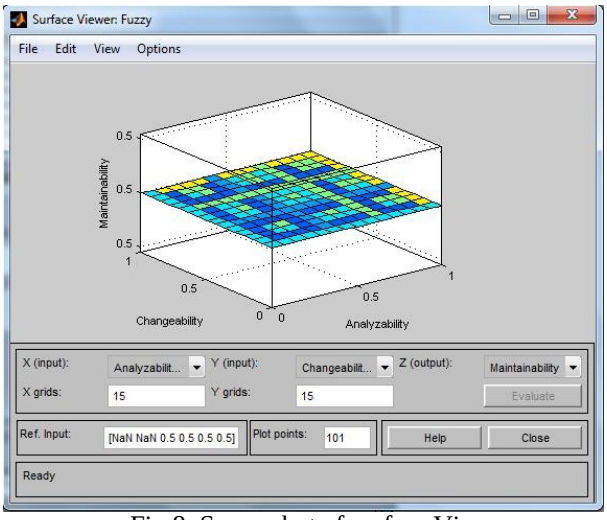

Fig 8: Screenshot of surface Viewer

VIII. CONCLUSION AND FUTURE SCOPE

Maintenance of software products highly important and is required after implementation phase of software lifecycle. In this paper, we proposed a framework to predict maintainability level of Object-Oriented Software system. First, we identified factors influencing maintainability of Object-Oriented System. These factors have been considered as input factors in Fuzzy Logic tool and maintainability as output factor. In order to set rule base of fuzzy logic, we have taken expert opinion. Our proposed model is an automated tool which can be used to predict maintainability of Object-Oriented System. This maintainability model can be used as guidelines by the software developers as they can develop Object-Oriented products in such a way that final product requires less maintenance.

REFERENCES

[1] J.A. McCall, P.K. Richards, and G.F. Walters, “Factors in Software Quality”, RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports, 1997.

[2] B. Boehm and H. In, "Identifying Quality-Requirement Conflicts," IEEE Software, Vol. 13, pp. 25-35, 1996

[3] R. Geoff Dromey's Model, “A Model for Software Product Quality”, IEEE Transaction On Software Engineering, Vol. 21, No. 2, Feb 1995.

[4] ISO 9126-1 Software Engineering - Product Quality - Part 1: Quality Model, 2001.

[5] P. Dhankar, H.K. Mittal, A. Mittal, and A. Rathee, “Maintainability Prediction For Object Oriented Software”, International Journal of Advances in Engineering Sciences, Vol.1, No. 1, Jan 2011.

[6] S. Ghosh, S.K. Dubey, and A. Rana, "Comparative Study of the Factors that Affect Maintainability", International Journal on Computer Science and Engineering (IJCSE), Vol.3, No.2, Dec 2011.
[7] K.K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Application of Artificial Neural Network for Predicting Maintainability using Object- Oriented Metrics", International Journal of Computer, Electrical, Automation, Control and Information Engineering, Vol. 2, No. 10, pp. 3552-3556, 2008.
[8] S. Rajasekaran and G.A. Vijayalakshmi Pal- Neural Network, Fuzzy Logic and Genetic Algorithms Synthesis and Applications; PHI Learning Private Limited; Rajkamal Electric Press, 2003.
[9] X. Li and L. Hendren, "Mc2FOR demo: A tool for automatically translating MATLAB to FORTRAN 95", IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), pp 458-463, Feb 2014.
[10] N. Goel, S.K. Dubey, and A. Rana, "Fuzzy Layered Approach for Maintainability Evaluation of Object Oriented Software System", International Journal on Computer Science and Engineering Research, Vol. 3, No. 6, June 2012.
[11] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp 476-493, June 1994
[12] S. Ghosh, S.K. Dubey, and A. Rana, "Fuzzy Maintainability Model for Object Oriented Software System", IJCSI International Journal of Computer Science Issues, Vol. 9, No. 2, July 2012.
[13] M.A. Chauman, H. Kabali, and P. Shrivastava, "Design Properties and Object-Oriented Software Changeability", IEEE, pp. 45-54, 2000.
[14] A. Kumar, R. Kumar, and P.S. Grover, "A Fuzzy Logic Approach to Measure Complexity of generic Aspect-Oriented Systems", Journal of Object Technology, Vol. 9, No. 3, May-June 2010
[15] E. Bouwers, J.P. Correia, A. Deursen, and J. Visser, "Quantifying the Analyzability of Software Architechtures", IEEE Conference on Software Architecture, pp. 83-92, 2011.
[16] I. Heitlager, T. Kuipers, and J. Visser J, "A Practical Model for Measuring Maintainability", IEEE, pp. 30-39, 2007.
[17] M. Bruntink and A. Deursen, "Predicting Class Testability using Object-Oriented Metrics", IEEE, pp. 136-145, 2004.
[18] Seban and R. Robert, "An overview of Object-Oriented Design", IEEE Conference on Aerospace Applications Conference, 1994.
[19] W. Li, "Software Product Metrics", Potentials, IEEE, Vol. 18, No. 5, pp 24-27, Dec 1999/ Jan 2000
[20] P. Hasia, A. Gupta, C. Kung, and J. Peng, "Study on the effect of architecture on maintainability of object-oriented systems", IEEE Conference on Software Maintenance, pp. 4-11, 1995.
[21] N. Tagoug, "Maintainability assessment in Object-Oriented System design", International Conference on Information Technology and e-Services (ICITeS), pp 1-5, 2012.
[22] S.K. Dubey and A. Rana, "A Fuzzy Approach for Evaluation of Maintainability of Object Oriented Software System", International Journal of Computer Applications, Vol. 49, No.21, July 2012
[23] Zadeh and A. Lotfi, "Fuzzy logic: issues, contentions and perspectives", IEEE International Conference On, Volume:vi, 1994
[24] R.D. Banker and S.M. Datar, "Software Complexity and Maintainability", ICIS Proceedings, pp. 247-255, 1989.
[25] M. Mattson, H. Grahn, and M. Frans, "Software Architecture Evaluation Methods for Performance, Maintainability, Testability and Portability", Second International Conference on the Quality of Software Architectures, 2006.
[26] R. Ubar, J. Heinlaid, J. Raik, and L. Raun, "Calculation of Testability Measures on Structurally Systhesized Binary Decision Diagrams", Proc. of the 6th Baltic Electronics Conference, 2006.
[27] E.P. Jeffery, T.A. Roger, and D.H. Charles, "Design-for-Testability for Object-Oriented Software", Object Magazine. Vol. 7, No. 5, 34-43, 1997.
[28] P. Dhankar, H. Mittal, A. Mittal, and Amita "Maintainability Prediction for Object-Oriented Software", International Journal of Advances in Engineering Sciences, Vol.1, Issue 1, January 2011.
[29] F. Abreu, M. Goulao, and R. Esteves, "Towards the design Quality Evaluation of Object-Oriented Software Systems" Proceedings of the 5th International Conference on Software Quality, 1995.
[30] R. Saini, S.K. Dubey, and A. Rana, "Analytical Study Of Maintainability Models For Quality Evaluation" Indian Journal of Computer Science and Engineering
[31] Peercy and D.E., "A Software Maintainability Evaluation Methodology", Software Engineering, IEEE Transactions on, Vol. SE-7, No. 4, July 1981.
[32] H.A. Al-Jamimi and M. Ahmed, "Prediction of software maintainability using fuzzy logic", IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS), pp. 702-705, 22, June 2012.
[33] B.L. Retterer and R.A. Kowalski, "Maintainability: A historical perspective", IEEE Conference on Reliability, Vol. R-33, No. 1, April 1984.
[34] A.F. Rosene, J.E. Connolly, and K.M. Bracy, "Software Maintainability - What It Means and How to Achieve It", IEEE Transactions on Reliability, Vol. R-30, No. 3, August 1981.