Asian Journal of Information Technology 15 (15): 2611-2620, 2016
ISSN: 1682-3915

Fuzzy Based Approach for Object-Oriented (OO) Software Maintainability Measurement

[1]P.R. Therasa and [2]P. Vivekanandan
[1]College of Engineering, Anna University, Chennai, India
[2]Department of Chemical Engineering, Anna University, Chennai, India

Abstract: The quality of the software is the very important factor in the field of software development which can be determined by many quality attributes of the software. Thus, the quantification of the quality parameters and incorporating them into the quality models are essential for software maintainability. The ISO/IEC 25010:2011 standard is developed to integrate the quality model based on software attributes. In this study, a fuzzy based model is proposed to predict the software maintainability from UML class diagram. The outcomes are presented and the knowledge modeling using fuzzy logic is discussed. The development of this model is based on the factors that affect the maintainability like other software quality factors. This hierarchy consists of factors, attributes and metrics which are used for measuring the maintainability of object-oriented software. This proposed model captures the factors that determine maintainability at design level and expressed by coupling and size attributes. Some of the metrics which quantify these attributes (NA, NM, NAssoc, NAgg etc.) are then considered as the input parameters to the proposed model. The process is applied to a case study where the Mamdani fuzzy inference engine is used to predict software maintainability. The performance of the model was evaluated using RMSE. In order to estimate the software maintainability, the analysis of different membership functions defined in fuzzy inference system by MATLAB for the mentioned metrics are presented.

Key words: Software maintainability, fuzzy logic, object oriented metric, structural complexity, coupling, size metric

INTRODUCTION

Software quality is not only based on the performance of functional requirement of a system but also based on the internal structural quality of the system. According to the ISO/IEC: standard definition, "software quality is the completeness of features and attributes of a software product that bear on its ability to satisfy stated or implied needs".

Software attributes can be classified in two different ways internal and external attributes. Internal attributes such as coupling, cohesion which are measurable to the concern with the developer of the product and external attributes such as stability, maintainability which are concerned with developer, customers, users and project managers of the product. Internal quality attributes are typically directly measured during different stages of the software lifecycle. We can measure the external quality attributes by means of measuring the internal quality attributes.

The software quality model is predestined to describe the various external attributes which draw the attention of the stakeholders along with their level of importance. Also, it expresses the dependencies between the significance of external attributes and the corresponding internal attributes which we can measure. Such internal attributes act as judges to predict future external quality attributes at the initial stages during the development life cycle of the software.

To design the metrics which depend on internal quality characteristics such as size, coupling, cohesion and complexity by the internal attributes. In turn, researchers and practitioners have proposed a large number of new metrics and valuation frames for quality plan principles such as size, complexity, low cohesion and high coupling. High complexity and high coupling are a sign of low quality or low maintainability of software systems.

Several software quality models were proposed by many researchers such as McCall's model Boehm's model (Boehm *et al*., 1978), ISO/IEC 9126 Models [21]. Chen and Liu (2009) presented an method to select the appropriate method of Software Development (SD). They also explained the important features of OOSDM and visual programming and also offered the results of the empirical validation. Dormey (1995) proposed a model for software

Corresponding Author: P.R. Therasa, College of Engineering, Anna University, Chennai, India

product quality. It is defined and formulated by combining a set of quality-carrying properties which correspond to the structural forms that are used to express the statements in a programming language. These quality carrying properties are consecutively connected to the high-level quality attributes such as functionality, reliability, usability, efficiency, maintainability and portability of the International Standard for Software Product Evaluation ISO-9126. As a result, they divide the quality carrying properties related to structural forms of programs and categorize into four basic forms. In order of priority, they are categorized as follows:

- Correctness properties which are based on basic minimal requirements for correctness
- Structural properties which are caused by low-level and intra-module design issues
- Modularity properties which are based on high-level and inter-module design issues
- Descriptive properties which are based on different forms of specifications

Distinguishing these models give us to define and measure the software quality in different viewpoints by describing and estimating a set of software internal attributes to evaluate the external software quality. In such software models, maintenance performance of a program has usually been assessed by measuring maintenance effort. That is the effort needed to complete the maintenance task by the entire programming team. The maintenance effort or maintainability of software can impact by the following object-oriented metrics which are structural complexity metrics such as coupling, cohesion and size metrics that recognize the ease of maintenance which is not directly measureable.

Literature reveiw: Presently, the UML class diagram complexity measurement is important in object-oriented analysis. Most of the researchers have carried out some progressive work in the area of UML class metric with a different number of principles and procedures. There are two way of approaches, namely, class diagram complexity metrics based on statistics method and entropy-distance method (Yi, 2006, 2010), respectively.

Muthanna proposed a maintainability model for Industrial software systems from the design level metrics by using polynomial regression techniques. They have used six metrics such as module level function point metric, module level information flow metric, module level global data flow, average fan-out, average knot count and average cyclomatic complexity metric. Correlation analysis is performed between each metric with subjective ratings provided by the software developers. The polynomial regression analysis is a statistical method for predicting values of one or more dependent variables from a collection of independent variables. Here, single maintainability prediction value was obtained by combining a minimal set of metrics (Table 1).

Genero *et al.* (2007) design a controlled experiment with 28 UML class diagram and found out value for UML class metric and maintainability sub-characteristics. Evaluated maintainability of the software based on statistics focus on calculating number of occurrence value or depth of properties (attributes), methods and relationships among classes. A statistical process is followed to test the relationship between the size metric NC, NA, NM, MaxDIT and MaxHagg and relational metric NAssoc, NAgg, NaggH, NGen, NGenH, NDep, NDepIn and NDepOut with the maintainability sub-characteristic. The above metrics conforms that the properties, methods and relationships will affect complexity and size of class diagrams, focused on counting number of the relationship among classes by considering the inheritance and inter-dependencies of the class. In fact, relationships among different classes influence the complexity of class diagrams with different degree.

Ahmed and Jamimi (2013) proposed a post-training procedure to enforce model transparency. They conducted a case study using maintainability as an external attribute and a quality prediction model is manifested in a form of a set of fuzzy rules which relate the internal attributes to the external attributes. It is worth noting here that there are two popular fuzzy inference engines for building fuzzy systems, namely, Mamdani Method (Mamdani and Assilian, 1975) and Takagi-Sugeno (T-S) Method (Takagi and Sugeno, 1985). Mamdani's engine offers very relevant capabilities when it comes to transparency and accommodating imprecise data. They compared its performance with other Machine Learning (ML) techniques' performance. They found that the Mamdani-based model seemed to be superior to all other models (Jamimi and Ahmed, 2013).

Evaluation of software maintainability based on the indices system. It has a basis of quantitative procedural information and it has used Analytic Hierarchy Process (AHP) to determine the weight of the evaluation indices, uses the fuzzy evaluation method to deal with the quality of the indices. This indices system consists of three level of indices; the first level indices is maintainability, second indices are analyzability, changeability, stability and testability, third level indices are basic metrics such as number of levels in the inheritance graph, average coupling between objects, attribute inheritance factor

Table 1: Maintainability value for different system

				Different weighting factor for Nc, Na and Nm											
				Nc	Na	Nm		Nc	Na	Nm		Nc	Na	Nm	
Diagram Id	Size	Coupling	Maintainability score	0.3	0.3	0.3	Size	0.7	0.7	0.7	Size	0.5	0.5	0.5	Size
1	2.6	0.6	-	3.0	3.4	3.0	4.2	3.8	6.6	4.6	9.8	3.4	5	4	7
2	3.9	1.0	9.67	4.5	5.1	4.5	6.3	5.7	9.9	6.9	15.0	5.1	8	6	11
3	5.2	1.4	9.70	6.0	6.7	6.1	8.4	7.6	12.7	9.7	20.0	6.8	11	9	14
4	4.1	1.8	-	4.7	5.3	4.8	6.6	5.9	10.1	7.6	15.0	5.3	8	7	11
5	7.5	1.8	8.57	8.5	9.6	8.9	12.0	10.5	18.0	14.5	28.0	9.5	15	13	20
6	3.9	1.2	-	4.5	5.1	4.5	6.3	5.7	9.9	6.9	15.0	5.1	8	6	11
7	4.4	2.2	9.70	5.2	5.6	5.2	7.2	6.8	10.4	8.4	17.0	6.0	9	8	12

(MOOD), method inheritance, etc., in leaf node to map to data. They got the quantitative result of software maintainability evaluation and solve the multi-index evaluation problems effectively.

Basically Fuzzy logic controller is used input as linguistic variable and gives output as a crisp value by mapping the fuzzy inference rules with input parameter. An experiment on the linguistic synthesis of a controller for a model industrial plant (a steam engine) is described by Mamdani (Mamdani and Assilian, 1975). Developed an Fuzzy logic controller to convert heuristic control rules specified by an operator into an automatic programmed control strategy. The experiment was initiated to investigate the possibility of human interface with a learning controller. Though, the control strategy set up linguistically proved to be far better than expected and reported a non-learning controller based on experiment of linguistic control synthesis.

MATERIALS AND METHODS

Approach: This model follows the methodology that implements the maintainability as a function of measurable attributes from the UML diagram. It is based on the amount of time needed to extend or modify the source code in future based on the size of the system and inter connection among the modules. These attributes are obtained from various types of UML class diagrams which demonstrate the coupling and size metric that determine the maintainability of the resulting software. The coupling is expressed in terms of internal and external relationships such as association, inheritance and aggregation with other classes in modules.

The size variable is expressed in terms of number of variables, methods, classes and depth of inheritance tree in the UML class diagram. These features will be measured using UML class metrics which are given in Table 2. Thus, coupling and size which are given as inputs to the proposed fuzzy based model and a single output is gained which is the estimated maintainability of the class diagram.

Table 2: UML class metric

Name of metric	Definition
NAssoc	The number of association metric
NAgg	The number of aggregation metric
NDep	The number of dependencies metric
NGen	The number of generalization metric
NGenH	The number of generalization hierarchies metric
NAggH	The number of aggregation hierarchies metric
MaxDIT	The maximum DIT metric is defined as the maximum between the DIT values obtained for each class of the class diagram
MaxHAgg	The Maximum HAgg metric is defined as the maximum between the HAgg values obtained for each class of the class diagram

Coupling = NAssoc×0.6+NDep×0.4+NAgg×0.4+NGen×0.3

Development platform: These calculations will be done using MATLAB while the design of the model will be implemented using Fuzzy Toolbox which is add-on tool in MATLAB. MATLAB is a high performance language for technical computing which integrates computation and programming. It makes use of mathematical notations to express the problems and solutions being modeled. It has several add-on toolboxes which are suited for various specialized technology. Among them, fuzzy logic toolbox is a collection of functions built on the MATLAB technical computing environment. It enhances creation and editing of fuzzy inference systems within the framework.

Fuzzy logic: Fuzzy logic was devised by Zadeh (1965, 1983). From that it has been began as a powerful technique for the controlling industrial and business processes, entertainment electronics and other expert systems. Most of the applications of fuzzy logic are in the area of industrial control. The inspiration for Fuzzy Logic was expressed in the following way. The ability of the human mind to deal with the information that is based on task relevant connected, though it has a marvelous amount of information is existing to the human minds in a given situation on particular piece of information that would fill a classic computer system. Likewise, the human mind has the ability to discard most of this information and to focus on the information that is only on task relevant. By concentrating only on the task-relevant information, the extent of information that the brain has to deal with is reduced to a manageable level.

Fuzzy logic is mostly a multiple valued logic that permits intermediate values to be well-defined between predictable approximations like yes or no, true or false, etc. Thinking like rather good or not bad can be expressed in precisely or mathematically and algorithmically processed. In this way, we can do challenge to apply like or a more human-like way of thinking in the programming concept of computers based on soft computing approach. Fuzzy logic systems are the method to address the fuzziness of the input and output variables by defining fuzzy numbers and fuzzy sets that can be expressed in linguistic variables like small, medium and large. Fuzzy rule-based approach to modeling is based on words to formulate the rules overlapped throughout the limitation area. They use mathematical exclamation to handle the complex non-linear relationships.

Fuzzy rule-based systems: Fuzzy rules are linguistic IF-THEN constructions that have the general form of 'IF A AND B THEN C" where A, B and C are suggestions containing in the linguistic variables. A and B are called the basis of the rule and C is the significance of the rule. In the outcome, the use of linguistic variables and fuzzy IF-THEN- rules activities are the tolerance for vagueness and disbelief. In this opinion, fuzzy logic simulators are the critical facility of the human mind to summarize the data and focus on decision-relevant information.

Factors affecting software maintainability: In software engineering, maintainability is defined as the ease with which a software system can be modified and extended in the operation phase. A software product is modified in order to correct the defects, enhance the new requirements and meet forthcoming maintenance easier. These activities are commonly called as maintenance task. There are several factors that influence the ease of these activities. The factors include software features at design level, coupling and size of the product. The staffing also very important in field of maintenance, one who develop the software, will be very easy to modify and extend the same. This study focuses on features of class diagram as an early artifact that may determine maintainability of the software. These features contain inter dependency or coupling among modules and size of the module. Coupling will occur when one module changes or depends on the internal workings of another module or other classes in a module. Evaluation of maintainability in terms of inheritance depth (Daly *et al*., 1996). This commonly denotes that changing the module will lead to change in the dependent module. It is inflexible to perform maintenance activities on highly coupled program code since a change in one module usually forces a ripple-effect on the other modules. On the other hand when a system has more number of variables, method, classes and high inheritance depth, it is very time consuming task to maintain that system. The size of the system also influences the maintenance activity. In large extent of software which reflects difficulty in comprehending the code and it leads the code complexity.

Metrics and data for the study: UML class metric was proposed by Genero *et al.* (2007) for structural complexity and size of the UML class diagram and proved the theoretical validation of these UML class metric as long as the empirical validation of their proposed metric by the controlled experiment. Based on the hypothesis formulation of the structural complexity measures (NAssoc, NAgg, NDep, NGen, NaggH, NGenH, MaxHAgg, MaxDIT) and the size measures (NC, NA, NM), it has found that it is a good predictors of maintainability sub-characteristics. Briand *et al.* (1991) gave an idea of a unified framework for coupling measurement in OO systems. All this studies focus on the structural complexity and size of the object oriented software based on UML class diagram. Data used for this study have been taken from the controlled experiment (Genero *et al*., 2007). It contains 11 UML class metric values from UML class diagram listed in Table 2 for 28 class diagrams.

Fuzzy based maintainability model: This section defines the fuzzy based model in detailed design in this fuzzy model as shown in Fig. 1, we have taken two inputs as coupling and size to provide a crisp input value of maintainability using rule base system. Fuzzy Inference System (FIS) uses fuzzy inference rules to map the input into crisp output. Here, we have used Mamdani fuzzy inference method. After the fuzzification process has been completed, we have to take the fuzzy sets for output variable that requires defuzzification process. For the defuzzification, the input will be a fuzzy set and output will be a crisp value. Here, we have used centroid method which gives the center of area under the curve. It is the most common method widely used for defuzzification process. The triangular membership function is used to represent the input as well as output.

Linguistic variables: There are four phases of development; these are first definition of the linguistic variable with linguistic terms low, medium and high for coupling and small, medium and large for size variable. The model has one output variable that is maintainability which will be described using four linguistic terms to give more precision output to be obtained. These terms are namely, low, medium, high and very-high.

Membership functions: The membership functions are used due to their ease and simplicity of understanding. It graphically describes a situation in real world. These standard membership functions include trapezoidal membership function and gaussian membership and triangular shaped membership functions named after the shape of their plots. The variables are fuzzified as shown in Fig. 2. Figure 3 shows the maintainability indicator application system for OO software using fuzzy logic system with two input parameter such coupling and size. Based on the UML class metric the system will find out the maintainability level of the software.

Membership function for coupling: The input parameter coupling is measured from the UML class metrics. The coupling is an inter module dependency. We can derive coupling measure from some of the metric that are associated with coupling. There are several parameters

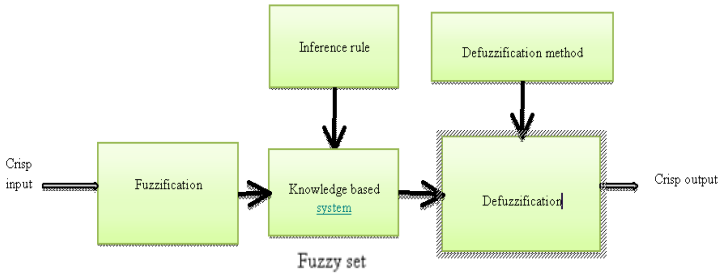

Fig. 1: Fuzzy inference system

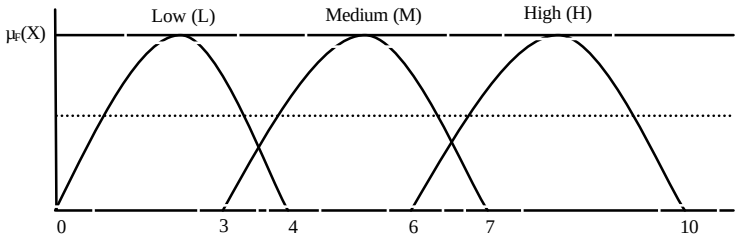

Fig. 2: An example of fuzzy logic membership function for coupling

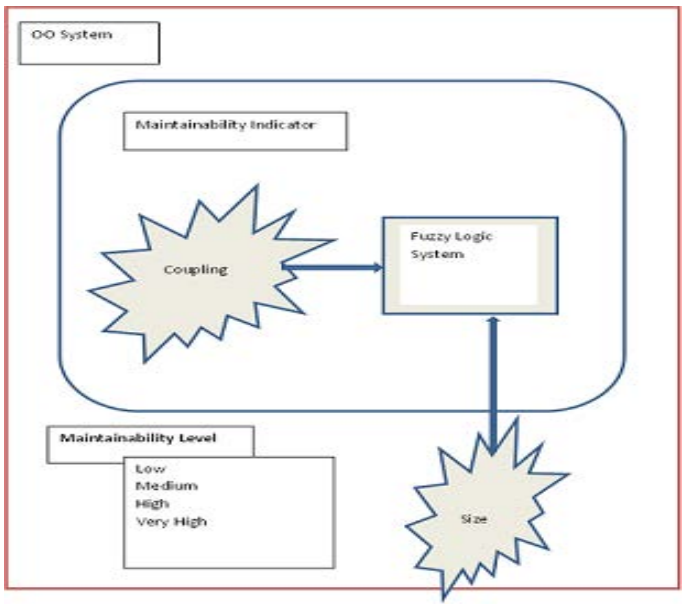

Fig. 3: Maintainability fuzzy logic system

that impacts the coupling between classes are Nassoc, Ndep, Nagg and Ngen. To normalize the input value by multiplying with the weighting factor assigned to it. The weighting factor is assigned by based on the impact of different parameter on coupling as shown in Table 3. Figure 4 shows the membership function for coupling with three fuzzy set low, medium and high with respective value 0-7.2, 5.4-12.7 and 11-36.

Membership function for size: The input parameter size is measured from the UML class metrics. As size measure is amount of code or content in each module, we can derive size measure from some metric that are associated with size. There are several parameters that impact the size of the UML class diagrams NC, NM/WMC, NA and NGenH and DIT. To normalize the input value with the weighting factor assigned to it is multiplied. The weighting factor is assigned by based on the impact of different parameter on size as shown in Table 4. To normalize the input value for size metric, multiply the input parameter with weight factor assigned to it.

Figure 5 shows the membership function for size parameter with three fuzzy set small, medium and large with respective value 0-26, 18-58 and 40-100.

Maintainability of the system: The maintainability of UML class diagram is measured based on two input parameter coupling and size. Figure 6 shows the membership function of maintainability. We consider here, low, moderate, high and very high easy of maintainability, i.e., if low maintainability means easy of maintainability which is low so it needs of higher maintenance effort.

Rule structure and rule matrix: The rule matrix is knowledge base rule used to describe fuzzy sets and fuzzy operators in form of conditional statements. The number of rules is a result of considering all the possible combinations of the two inputs with their linguistic terms and a rule is assigned to each combination. The rules are expressed in a single fuzzy if-then rule can be as follows,

Table 3: Weighting factors for coupling measure

Nassoc	NDep	NAgg	NGen
0.60	0.40	0.40	0.30

Table 4: Weighting factor for size metric

NC	NM/WMC	NA	NGenH	DIT
0.10	0.20	0.20	0.30	0.20

Size = NC×0.1+NA×0.2+NM×0.2+NGenH×0.3+MaxDIT×0.2

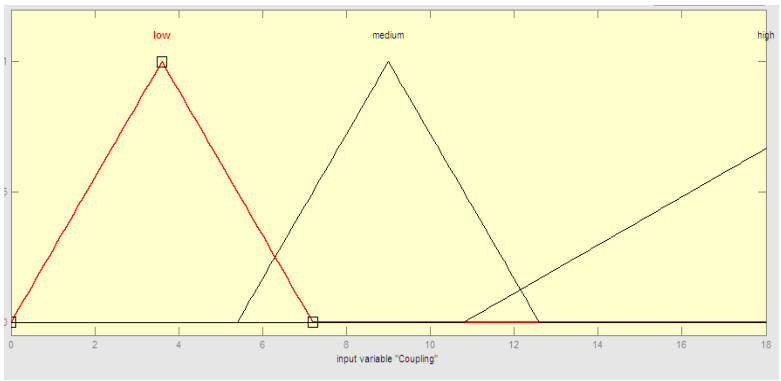

Fig. 4: Membership function for coupling

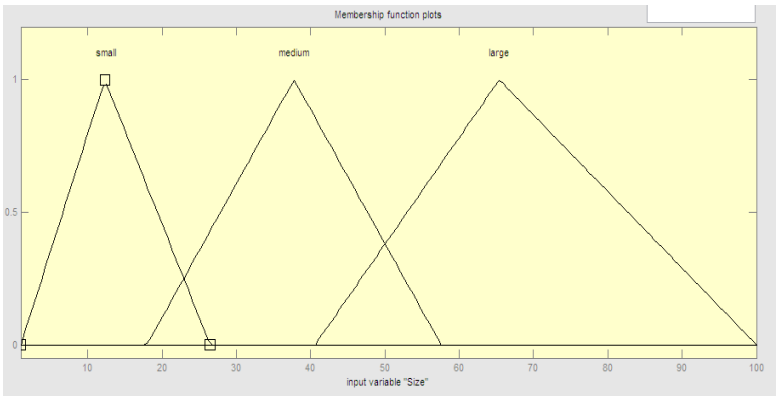

Fig. 5: The membership function of size parameter

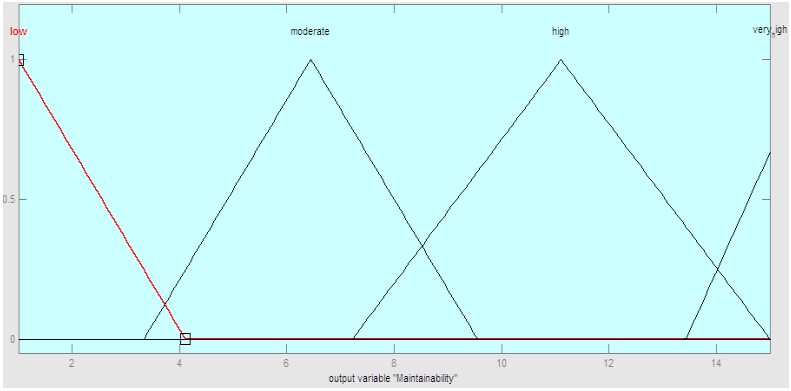

Fig. 6: Membership function for output maintainability

1. If (Coupling is low) and (Size is small) then (Maintainability is very_high) (1)
2. If (Coupling is low) and (Size is large) then (Maintainability is moderate) (1)
3. If (Coupling is high) and (Size is small) then (Maintainability is moderate) (1)
4. If (Coupling is medium) and (Size is large) then (Maintainability is moderate) (1)
5. If (Coupling is medium) and (Size is small) then (Maintainability is moderate) (1)
6. If (Coupling is medium) and (Size is medium) then (Maintainability is moderate) (1)
7. If (Coupling is low) and (Size is medium) then (Maintainability is high) (1)
8. If (Coupling is high) and (Size is medium) then (Maintainability is low) (1)
9. If (Coupling is high) and (Size is large) then (Maintainability is low) (1)

Fig. 7: Inference rules

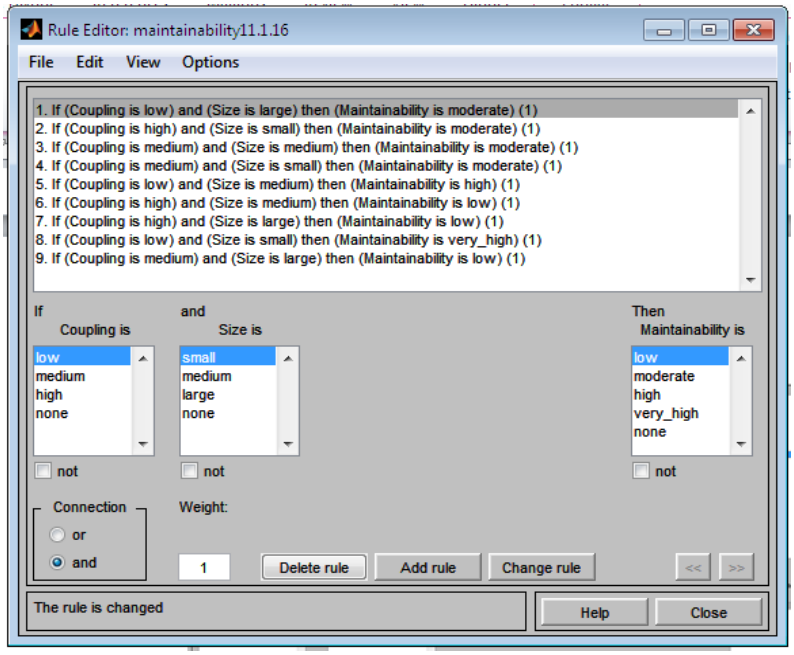

Fig. 8: Inference rules generated in MAT Lab

if x is A then y is Z, i.e., wordy format. If (Coupling is Low) and (size small) then (Maintainability is very good). Here the Fig. 7 shows the inference rules generated for the input parameters coupling and size with the output parameter maintainability. Figure 8 shows the inference rules generated in MAT Lab with

input and output parameters. Two inputs: coupling and size. One output: maintainability score.

Selection of appropriate defuzzification method: The selection of a defuzzification process is usually based on the considerations of strength of input parameter. The last step called defuzzification process, i.e., to convert the fuzzy inference into a crisp output. The centroid method is commonly used for defuzzification to get the single output. Here, we applied centroid method to get the crisp output.

RESULTS AND DISCUSSION

Suppose we have the following inputs to the model as shown in the sample interface below: rule viewer diagram for two input variable and one output as shown in Fig. 9. Figure 10 shows surface plot FIS showing relationship between inputs and output parameters after training. The plot results from a rule base with 3×3×4 rules. The area under the origin indicates a high maintainability towards deviations in either coupling or size near to the place. This leads if the high maintainability should be obtained with small size and low coupling.

Performance evaluation using Root Mean Square Error (RMSE): The performance of the fuzzy system was evaluated using the RMSE. The developed fuzzy model evaluated in MAT lab using command evalfis. This function has two argument the first one is the input array with two column and the second one is the FIS file created with membership function and set rules to achieve the desire result maintainability. Figure 11 show the comparison of actual and predicated value of the maintainability from fuzzy model. The RMSE is evaluate using the MAT lab Curve Fitting Tool with the degree of polynomial 3 and 95% confidence bounds. Linear model Poly3:

$$f(x) = p1*x^3+p2*x^2+p3*x+p4$$

Coefficients (with 95% confidence bounds):

- p1 = -0.04693 (-0.08582, -0.008044)
- p2 = 0.792 (0.194, 1.39)
- p3 = -2.896 (-5.491, -0.3016)
- p4 = 5.328 (2.164, 8.49)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(a_{obs,i} - p_{model,i})^2}{n}}$$

Where:

n = The number of sample
a = The actual value of maintainability
p = The predicated value of maintainability from the fuzzy model with two input parameter coupling and size with fuzzy rules

The evaluated Goodness of fit in R^2, Adj R^2 and RMSE value. SSE: 70.62; R^2: 0.6846; Adjusted R^2: 0.6451; RMSE: 1.715. Figure 12 shows the goodness of curve fitting diagram by Genero *et al.* (2007) with developed fuzzy model. The obtained RMSE and Coefficient of Determination R^2 value was 1.715 and 0.6846, respectively.

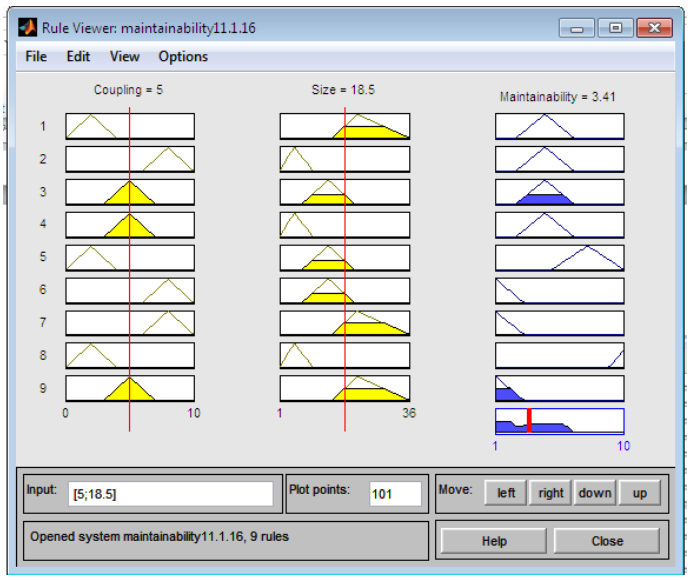

Fig. 9: Rule viewer diagram for two input variable

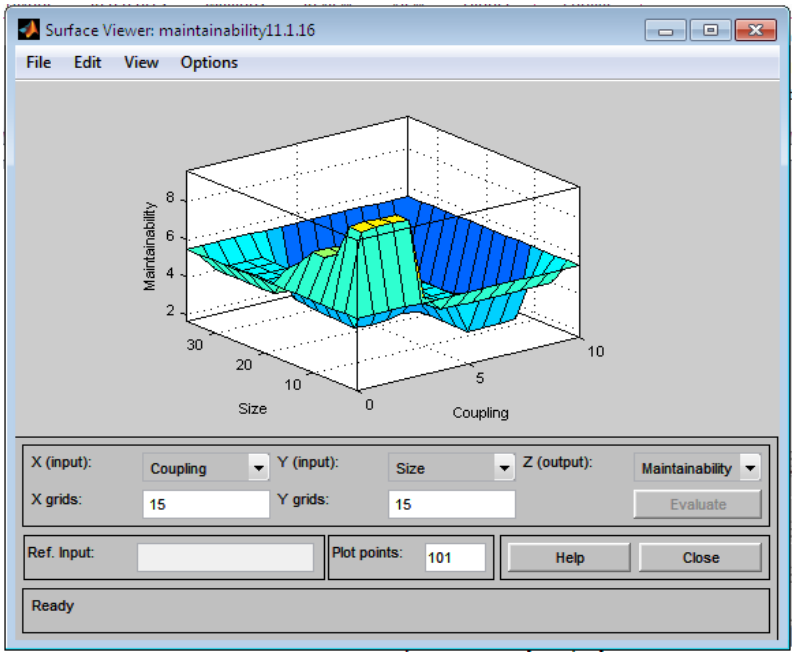

Fig. 10: Surface plot for FIS

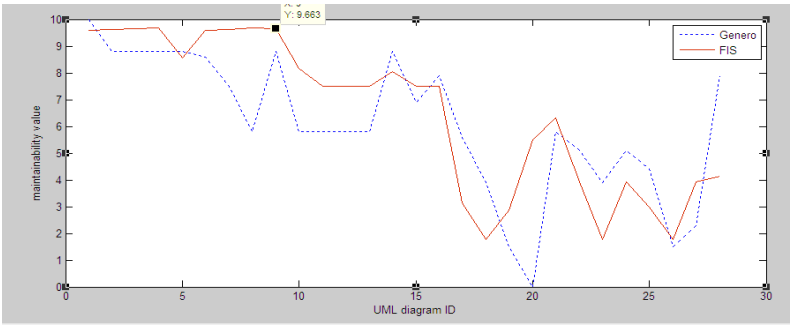

Fig. 11: Comparision of genero and fuzzy model

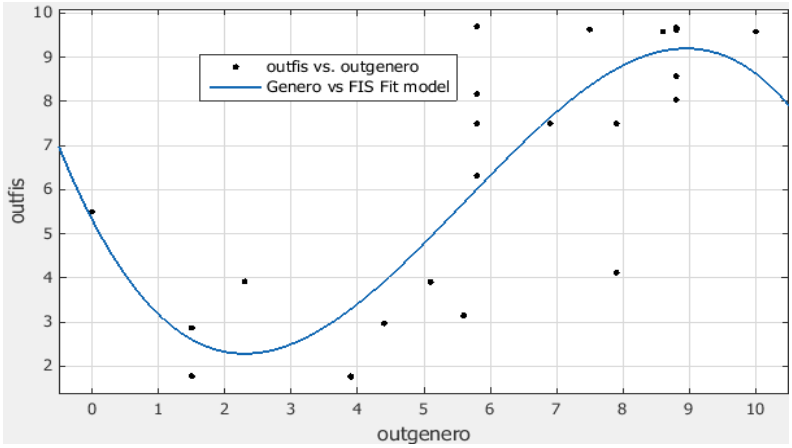

Fig. 12: Goodness of fit by Genero *et al*. (2007) and fuzzy model

CONCLUSION

In this study a fuzzy based method is proposed for predicting software maintainability using two parameters coupling and size. The Fuzzy Inference System (FIS) was developed with two input and one output, i.e., maintainability. The triangular membership function is used to specify the input as well as the output. The crisp inputs and specified fuzzy inference rules are given to FIS. The FIS will return the fuzzy output as result. Then to get the crisp output, the fuzzified input is given to the defuzzification process. The defuzzification process uses centroid method to find crisp output. The model is validated on select UML class metric data to check the effectiveness of the approach. Lower value of maintainability indicates the need for improvement in class diagram, so that the maintenance cost can be reduced. A crisp value for maintainability can help software leaders to review the maintenance efforts in the early phase of software and do the required s maintenance activities. The performance of the developed fuzzy model for maintainability was compared with Genero Model (Genero *et al*., 2007). The technique of fuzzy logic used in this study is sufficiently and easily applied in other areas of measureable (to quantify) software engineering systems to remove the vagueness in the results:

- Different weight factors may affect the system maintainability through the impact of the input
- As size is lower, we will have high maintainability so we can choose the lower size value
- The output is depending upon two factors, we have to choose the factor that affect two input variables
- Analyzed the output for single class diagram with various factors to train the system
- If we analysis the large number of sample data to train the system, we can achieve better result

The above table shows the different UML diagram with varying size and coupling factor with their maintainability. It shows that the different weighting factor affects the maintainability of the system. Thus, it tells that the considered two input parameters impact the maintainability of the system.

REFERENCES

Ahmed, M.A. and H.A.A. Jamimi, 2013. Machine learning approaches for predicting software maintainability: A fuzzy-based transparent model. IET. Software, 7: 317-326.

Briand, L., W. Daly and J. Wust, 1991. A unified framework for coupling measurement in object-oriented systems. IEEE Trans. Software Eng., 25: 91-121.

Chen, J. and X. Liu, 2009. Software Maintainability Metrics Based on the Index System and Fuzzy Method. Proceedings of the 2009 First International Conference on Information Science and Engineering, December 26-28, 2009, IEEE, Nanjing, China, ISBN: 978-1-4244-4909-5, pp: 5117-5120.

Daly, J., A. Brooks, J. Miller, M. Roper and M. Wood, 1996. Evaluating inheritance depth on the maintainability of object-oriented software. Empirical Soft. Eng., 1: 109-132.

Dormey, G.R., 1995. A model for software product quality. IEEE Trans. Software Eng., 21: 146-162.

Genero, M., E. Manso, A. Visaggio, G. Canfora and M. Piattini, 2007. Building measure-based prediction models for UML class diagram maintainability. Empirical Softw. Eng., 12: 517-549.

Jamimi, H.A.A. and M. Ahmed, 2013. Machine learning-based software quality prediction models: state of the art. Proceedings of the International Conference on Information Science and Applications, June 24-26, 2013, IEEE, Suwon, South Korea, ISBN: 978-1-4799-0602-4, pp: 1-4.

Mamdani, E.H. and S. Assilian, 1975. An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man-Mach. Stud., 7: 1-13.

Takagi, T. and M. Sugeno, 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE. Trans. Syst. Man Cybernetics, 15: 116-132.

Yi, T., 2006. Research on UML-Model-Oriented Dependence Analysis and its Applications. Chinese Science and Technology Press, Hefei, China.

Yi, T., 2010. Comparison research of two typical UML-class-diagram metrics: Experimental software engineering. Proceedings of the International Conference on Computer Application and System Modeling, October 22-24, 2010, IEEE, Taiyuan, China, ISBN: 978-1-4244-7235-2, pp: 12-86.

Zadeh, L.A., 1965. Fuzzy sets. Inform. Control, 8: 338-353.

Zadeh, L.A., 1983. A computational approach to fuzzy quantifiers in natural languages. Comput. Math. Appl., 9: 149-184.