International Journal of Computer Applications (0975 – 8887)
Volume 65– No.25, March 2013

Validating Extendibility of the Object-Oriented Software using Fuzzy Computing Techniques

Vibhash Yadav
Computer Science & Engineering Department
Krishna Girls Engineering College
Kanpur, India

Raghuraj Singh
Computer Science & Engineering Department
Harcourt Butler Technological Institute
Kanpur, India

ABSTRACT

A large number of metrics have been proposed for measuring quality of object-oriented software from its code. These include size, inheritance, cohesion and coupling, abstraction, hierarchies, encapsulation, composition, polymorphism, messaging etc. These object-oriented metrics affect the design quality of object oriented software as they are related with the design attributes like Reusability, Functionality, Effectiveness and Extendibility. In this paper, a fuzzy logic based model have been proposed that analyses object oriented metrics for one of the important attributes i.e. Extendibility. The model can be used to validate the precise role of design quality metrics in Extendibility of a software design. On the basis of results obtained, it has been concluded that the design quality of Object Oriented Software can be best assessed by fuzzy analysis of design quality metrics.

General Terms

Object Oriented Design Quality

Keywords

Design Quality, Product Quality, Design Quality Metrics, Design Properties, Design Quality Attributes, Fuzzy Computing.

1. INTRODUCTION

Object-oriented design and programming is the dominant development paradigm for software systems today. With the growing complexity and size of object oriented systems, the ability to reason about quality attributes based on automatically computable measures is becoming increasingly important(1,2). Set of metrics have been suggested which are independent and are useful to compute the design quality of software. Table 1 lists some of the important identified metrics for design quality.

Table 1. Metrics For Design Quality

Sr. No.	Metric	Name of the Metric	Description
1.	DC	Design size in class	It counts the total number of classes
2.	NH	Number of hierarchies	It is a count of total number of class hierarchies
3.	ACA	Average count of ancestors	This metric count the average number of classes from which a class inherits the information
4.	DAM	Data access metrics	This metric is ratio of number of protected/private attributes to the total number of attributes defined in the class
5.	CC	Class coupling	It counts the value through which different classes are directly related to
6.	CAM	Cohesion among methods in class	This compute the value through which methods of a class are related
7.	MA	Measures of aggregation	It count the number of data declarations whose types are user defined
8.	CPM	Count of polymorphic methods	This count the methods of polymorphic behavior
9.	CIS	Class interface size	It counts the number of public methods in a class
10.	TNM	Total number of methods	It counts the total number of methods in a class
11.	MOA	Measure of abstraction	It is the ratio of the number of the methods inherited by a class to the total number of methods accessible by member methods of the class

Each of the above metric is related with one specific design property. The standard properties for the metrics are given in the table 2.

Table 2. Design Property

Design property	Definition
Design Size	The total number of classes used in a design.
Hierarchies	These are used to represent different generalization-specialization perceptions in a design. It is the total number of classes containing children in a design and are non inherited.
Abstraction	A measure of a generalization-specialization on aspect of the design classes in a design which has one or more descendants exhibit this property of abstraction.

Encapsulation	Defined as the bundling of data and behavior within a single module. In object oriented design it means that the internal representation of an instance is hidden from view outside of the instances definition.
Coupling	Defines the interdependency of an object on other object in a design. It is a measure of the number of other object that would have to be accessed by an object in order for that object to function correctly.
Cohesion	Accesses the relatedness of methods and characteristics in a class. Strong cohession is indicated by strong overlap in the module parameters and characteristics.
Composition	Measure the aggregation relationship in an object oriented design like "part-of", "has", "consist-of", "part-whole" relationship.
Inheritance	It is a method to find the relationship amoung/between classes which is related to the level of nesting of classes in an inheritance hierarchy.
Polymorphism	Defines the substitution of objects with matching interfaces at runtime.It also measures dynamically determined services at run time.
Messaging	A count of the number of public methods that are available as services to other classes.it is the services measure which a class provieds to others.
Complexity	The difficulty level in understanding and interpreting the entire structure of classes along with their relationships.

Six higher order design quality attributes are described in the Table 3 below. These attributes can be treated as major decisive factors for the design quality Object Oriented Software.

Table 3. Standard Design Quality Attributes

Quality Attributes	Definition
Extendibility	It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension.
Effectiveness	It is the method to get the functional working and its behaviour for object oriented systems.
Functionality	The obligations assigned to the classes of a design, that are made accessible by the classes by using their public interfaces.
Reusability	Reusability is object oriented design property that allows a design to be reapplied to a problem without any significant effort.
Flexibility	This characteristic allows the incorporation/ changes in the design. It is inversly proportional to the software quality.
Understandability	This is the property of design that enables understandability to be easily learned. It is directly related to the complexity of the software design .

2. EXISTING QUALITY MODELS

For evaluating quality indirect models have been developed by researchers to measure software product quality by using a set of quality attributes, characteristics and metrics (1,3). The main assumption for determining the quality models is that external product attributes are influenced by characteristics of internal products and also evaluation of internal characteristics concludes about external quality attribute of products (2, 3, 4).

Please use a 9-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

2.1 McCall's Quality Model (1977)

The quality model presented by Jim McCall et al,(also known as the General Electric Model of 1977) is one of the most renowned predecessor of today's quality models.

For determining the product quality of a software, McCall quality model gives three major aspects:

a) Product revision- ability to undergo changes.
b) Adaptability to new environments
c) Product operations its operation characteristics.

The main idea behind this model is that a complete software quality picture should be provided by synthesized quality factors. Answers to yes and no questions accomplish the actual quality metric that then is put in relation to each other.

2.2 Boehm's Quality Model (1978)

Barry W. Boehm's quality model is the second basic and commencing predecessors of today's quality models. Boehm addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the software quality. Software quality is qualitatively defined by a given set of attributes and metrics through his models (3,5,6). Overall quality level contributed by a hierarchical quality model, structured around primitive, intermediate and high level characteristics presented in both Boehm's as well as in McCall's quality model..

2.3 Dromey's Quality Model

R. Geoff Dromey's quality model is the most recent model that is similar to McCall's and Boehm's quality model. A product based quality model proposed by Dromey identifies that quality evaluation differs for each product and also there is a need for a more dynamic idea for modeling the process that can be applied for product transition in different systems (4,7). In this model the relationship between the quality attributes and the sub-attributes is focused along with the attempt to connect software product properties with software quality attributes.

Elements of this model are:
a) Product properties that influence quality
b) High level quality attributes
c) Means of linking the product properties with the quality attributes.

2.4 ISO 9126

ISO 9126 is the software product evaluation standard from the International Organization for Standardization. It defines six properties which describe software quality and minimum overlap.

ISO 9126 also introduces another type of quality-quality in use-having following elements:-

a) Effectiveness

b) Productivity

c) Safety

d) Satisfaction

ISO 9126 describes the following quality parameters:

a) Functionality

b) Reliability

c) Usability

d) Efficiency

e) Reusability

f) Portability.

3. FUZZY COMPUTING

Concept of Fuzzy Logic began in 1965 with the proposal of fuzzy set theory by Lotfi Zadeh. Fuzzy logic is a form of many-valued logic; it deals with reasoning that is fluid or approximate rather than fixed and exact.[8] In contrast with "crisp concept", where binary sets have two-valued: 1 for true or 0 for false, fuzzy logic variables may have a true value that lie between 0 and 1. Fuzzy logic has been again extended to handle the concept of partial truth, where the value of truth may exist between completely true and completely false. Furthermore, for linguistic variables, these degrees may be managed by specific/defined functions.

3.1 Using Fuzzy Logic to represent Metric Data

The proposed solution to the problem of boundary definition is fuzzy logic. Fuzzy logic provides the ability for a machine to perceive the world as humans do by representing vague and ambiguous knowledge (Negnevitsky, 2005).The intention is to blur the boundaries of binary thinking by allowing the classification of objects into multiple sets. Fuzzy sets are intended to blur boundaries so that various items can belong to any set (Hopgood, 2001).

3.2 Mamdani's Method

The process can be described using four steps [9]:

Step 1. Evaluate the antecedent for each rule

Given the inputs (crisp values), we obtain the membership values for those one (inputs). This whole process of finding membership function is known as 'input fuzzification'. If the rule antecedent has more than one part, a fuzzy operator (t-norm or t-conorm) is applied to obtain a single membership value.

Step 2. Obtain each rule's conclusion

Given the consequent of each rule (a fuzzy set) and the antecedent value obtained from step 1, we proposed/apply a fuzzy implication operator to obtain a new fuzzy set. Two of the most commonly used implication methods are:

1) minimum, which truncates the consequent's membership function, and

2) product, which scales it.

Step 3. Aggregate conclusions

In this step, we combine the outputs obtained for each rule in step 2 (obtain conclusion) into a single fuzzy set, using a fuzzy aggregation operator. Some of the most commonly used aggregation operators are the sum, the probabilistic sum and the maximum.

Step 4. Defuzzification

When we try to solve a decision problem, we want the output in crisp value set and not a fuzzy set. Therefore, we need to transform the fuzzy set we obtained in step 3 into a single value. The most famous methods of defuzzification are the centroid, which returns the area of the center under the fuzzy set obtained in the step 3.

It is worth mentioning that Mamdani's method is useful when there is a small number of a variable. Otherwise, the following difficulties may rise:

a) The number of rules increases exponentially with the number of variables in the antecedent.
b) The more rules we construct, the harder is to know if they are suitable for our problem.
c) If the number of variables in the antecedent is too much large, then it would be difficult to grasp the casual relationship between the antecedents and the consequents. So, constructing new rules will become too harder.

3.3 Fuzzy Logic Implementation in Matlab

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. Mamdani-type inference, as defined for the toolbox, expects the output membership functions to be fuzzy sets. After that for the aggregation process, there exist a fuzzy set for each output variable that requires defuzzification. It is possible, and in many cases it is much efficient, to use a single spike as the output membership functions instead of distributed fuzzy set.

The Fuzzy Inference System in MATLAB consists of the FIS Editor, Rule Editor, Rule Viewer, Surface Viewer and Membership Function Editor. In the FIS Editor, The drop-down lists let you modify the fuzzy inference functions. The Current Variable Area displays the name of an input or output variable, its type, and default range. A status line at the bottom displays information about the most recent operation.

The Membership Function Editor is the tool that lets display and edit all of the membership functions associated with all of the input and output variables for the entire fuzzy inference system. The Rule Viewer displays a roadmap of the whole fuzzy inference process.

4. MODEL DEVELOPMENT

Here two approaches have been used for the assessment of Software Quality. The first approach calculates Design Quality from UML diagrams of Object Oriented Software as input. In the second approach, code base is taken for assessment of the product quality. The results of both the approaches are compared. Regression and other analysis approaches are used to show the relationship between design quality and product quality and establish the co-relation between the two metrics. Design quality and Product quality are assessed for three category of software; i.e. Good, Medium and Bad which is known priori. Thus, fuzzy logic can be applied at its best for the analysis of design quality metric data

As specified in table 3, six quality attributes have been identified which affect the overall quality of an object oriented design software system. In order to evaluate their individual impact on software quality, identification of various design properties & corresponding metrics for each individual quality attribute have been done. Then the rules for quality attributes defined and generated membership function through which we get rule view for each one. From these rule views the metric values corresponding to the attribute can be analyzed. In the following subsection, work shows for one of these qualities attributes i.e. Extendibility.

4.1 Extendibility

This refers to the presence and usage of properties in an existing system that allows for the incorporation of new requirements in the design (3,9). This property of software quality is measured by the following design properties:

Table 4. Design Properties For Extendibility

Sr. no.	Design property	Metric
1	Abstraction	ACA
2	Coupling	CC
3	Inheritance	MOA
4	Polymorphism	CPM

Combined impact of these metrics on Extendibility is in proportionately relation which has been established as [5]:

Extendibility = 0.5*Abstraction - 0.5*coupling + 0.5* Inheritance +0.5* Polymorphism

Metrics that are connected with extendibility have been detailed in Table 6.

Table 6: Metrics used for predicting Extendibility

Metric	Name	Category	Description
ANA	Average number of Ancestors	Abstraction	This Metric value signifies the average number of classes from which a class inherits information.
DCC	Direct class coupling	Coupling	Counts of different no of classes that a class is directly related to.
MFA .	Measure of functional abstraction	Inheritance	The ratio of the number of methods inherited by a class to the total no of methods accessible by member methods of class
NOP	No. of polymorphic methods	Polymorphism	Count of the methods that can exhibit polymorphic behavior.

The FIS Editor (Figure 1) for extendibility takes the four metrics i.e. ANA, DCC, MFA and NOP as input. Membership function for extendibility has been defined as follows: Low (0 - 0.7 with 0.3 as Peak Value), Fuzzy Value (0.4-0.7), High (0.4 – 0.9 with 0.8 as Peak Value). The Membership Function definition for Extendibility is shown in Figure 2.

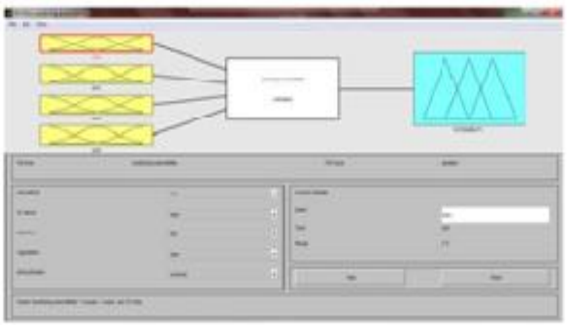

Fig 1: The FIS Editor for Extendibility

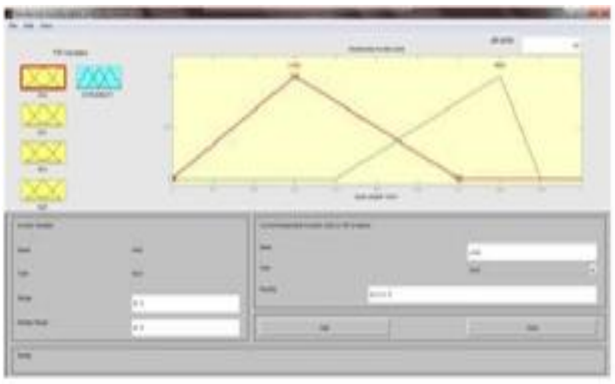

Fig 2: Membership Function Definition for Extendibility

Combined impact of extendibility related metrics on Software Quality has been analyzed using fuzzy logic. Rules view for the same is given in Figure 3.

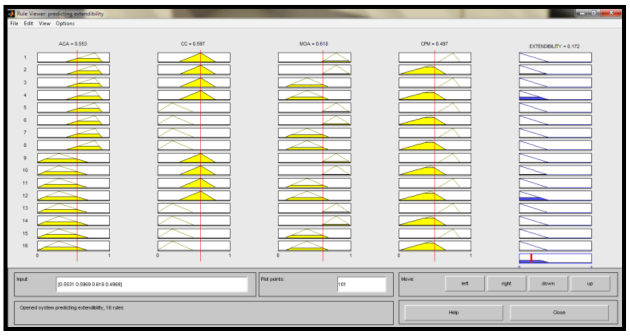

Fig 3: Rules View for Extendibility

This work is a part of a bigger project related to the assessment of design quality of Object Oriented Software from UML Design and Product Code. Fuzzy logic technique has been used for validating the design quality metrics. In this paper, we have considered one of the six design quality attributes i.e., Extendibility. The details of software selected for this purpose are given in the Table 5.

Table 5. Chosen Open Source Software with Priori Known Design Quality

Project No.	Name of Software	Level of Quality
Project1	Student project	Low design quality
Project2	EviewApplet	Low design quality
Project3	Bonforum	Medium design quality
Project4	Taming Java Thread	High design quality
Project5	Jdom	High design quality

5. RESULTS AND DISCUSSIONS

Our Initial results confirm that fuzzy logic based design quality assessments have given better validation in terms of good, intermediate and poor design quality of Object Oriented Software. Values of Extendibility for the five chosen open source software obtained using fuzzy logic based approach are better confirming with the results of product-based approach as compare to the traditional approach. Figure 4 summarizes the results obtained from the two approaches with reference to product- based approach.

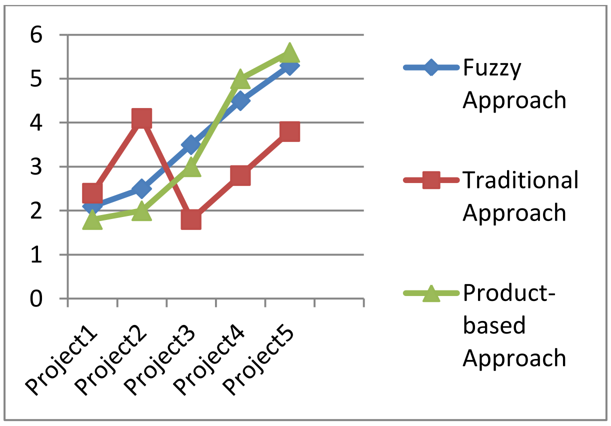

Fig 4: Extendibility Values: Fuzzy vs. Traditional Approaches

6. CONCLUSIONS

This work makes the analysis of one of the important design quality attributes i.e., Extendibility using the fuzzy analysis of its related design quality metrics. The same approach shall be used for the validation of the other remaining design attributes in our ongoing project i.e. assessment of design quality of Object Oriented Software.

The evaluation of software quality is an important aspect for software providers as well as users. The model proposed by us is based on fuzzy logic which could be an exclusive tool for making decision at design level. Since the model is not subjective, the prediction of software quality can effectively be done by it using object oriented metrics. The quality of software has the linear relation with the metrics. These metrics provide the approximate values on which the desired quality can be achieved. The evaluation results are based on the relationship of the metrics with the parameters of quality.

7. REFERENCES

[1] K K Agarwal, Y Singh, Arvinder Kaur and Ruchika Malhotra: Empirical study of Object Oriented Metrics, Journal of Object Technology, Vol. 5, No. 8, Nov-Dec 2006.

[2] L. C. Briand, W. L. Melo, and J. Wust, “Assessing the applicability of fault-proneness models across object-oriented software projects", IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 706-720, July 2002.

[3] W.J. Brown, R.C. Malveau, H.W. McCormick, III, and T.J. Mowbray, “AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis”, John Wiley Press, 1998.

[4] Yuming Zhou and Hareton Leung, “Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults,” *IEEE Trans. Software Eng.*, vol. 32, no. 10, pp. 771-789, Oct. 2006.

[5] Jagdish Bansiya and Carl G Devis: A Hierarchical Model for Object Oriented Design Quality Assessment, IEEE transactions of Software Engineering, Vol 28, No. 1, January 2002.

[6] W.J. Brown, R.C. Malveau, H.W. McCormick, III, and T.J. Mowbray, “AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis”, John Wiley Press, 1998.

[7] S.R. Chidamber, C.F. Kemerer,” A metrics suit for object oriented design”, IEEE Trans. On Software eng., vol. 20, no. 6, p.p. 476-493, June 1994.

[8] Zadeh, L.A. (1965). "Fuzzy sets", Information and Control 8 (3): 338–35

[9] MATLAB tutorial on Fuzzy Logic