An Analysis of Software Complexity using Fuzzy Logic for Object-Oriented Software

MukeshBansal[1*], C. P Agrawal[2]
Affiliations
1. **Department of Computer Engg. Govt Polytechnic Hisar- 125001, Haryana,India, mukeshbansal76@gmail.com**
2. **Department of Computer Science and Applications, M. C. N. U. J. C, Bhopal - 462039, Madhya Pradesh, India**

Abstract: Measurement is fundamental to any engineering discipline. There is considerable evidence that object-oriented design metrics can be used to make quality management decisions. In this paper we proposed a system based on fuzzy logic to assess the software complexity of OO design, uses the CK metric suite and Mamdani Inference Engine. A new model is proposed using fuzzy inference system for tuning the performance of software complexity. Fuzzy logic is a form of many-valued logic, designing a knowledge base model with four input metrics WMC, NOC, CBO, and RFC. These metrics are closely related to the factor i.e Reusability, Maintainability, Testability, Understandability and Efficiency. These factor are independent to each other and used for to assess the complexity of the software. we have defined and evaluated factors combination which is used for the assessment of software complexity of object oriented software.

Keywords: Quality, Fuzzy Logic, CK Metrics, Mamdani Model, Object Oriented Software.

1. Introduction

Object oriented systems continue to share a major portion of software development and customer base for these systems is on the rise. This is because there are huge incentives in taking the object oriented approach. The drawback though is that most object oriented systems tend to be quite complex. Hence, the quality of such systems takes precedence and lots of time, money and effort is spent in ensuring it. One such method that predicts quality of a software system is by evaluating key attributes of the software through the use of metrics. Software quality is especially a superior area when it comes to prediction based on metrics. The introduction and subsequent use of metrics as a means to evaluate the software quality has had deep and useful impact on the overall system. But the success of software quality assessment through metrics is hindered by the need for constant validation to ensure the accuracy of such predictions.

It is well known that the quality of software is an “easy-aware, difficult to define, and can not be measured”[12]concept. In order to explain such a concept, a lot of work has been done recently. ISO/IEC 9126 Standard provides a framework for assessing complexity of software and definition of five software quality properties, including Reusability, Maintainability, Testability, Understand ability and Efficiency. Different users focus on different software quality properties. Object-oriented software usually contains a large number of internal attributes, which can provide more accurate and comprehensive descriptions of software’s internal structure and nature. To date, there are large numbers of metrics have been posed, among which Chidamber&Kemerer (C&K) metrics[17] are proved and recognized to be atypical and useful set of Object-oriented (OO) software metrics, including DIT (Depth of the Inheritance Tree), NOC (Number Of Children), CBO (Coupling Between Object classes), LCOM (Lack of Cohesion in Methods), WMC (Weighted Methods for per Class), RFC(Response For a Class).

2. Review of Literature

Over the past years, with the advent of new methodologies, process driven management many approaches have been developed to address the problem of detecting and correcting design flaws in an OO software system using metrics. Moreover, with the ever increasing number of software metrics being introduced the project managers find it hard to interpret and understand the metric scores. Chidamber and Kemerer are the predominantly referenced researchers, they proposed 6 metrics- Weighted Methods per Class (WMC), Response sets for Class (RFC), Lack of Cohesion in methods (LCOM), Coupling Between Object Classes (CBO), Depth of Inheritance Tree (DIT), Number of Children of a class (NOC), with the help of which various software quality attributes (e.g. efficiency, complexity, understandability, reusability, maintainability and testability) can be measured. They claim that using several of their metrics collectively can help project managers and designers make better design decisions.

MOOD metric set model, proposed by Abreu [3] is another basic structural method of the object-oriented paradigm. They were defined to measure the use of object-oriented design methods such as inheritance (MIF (Method Inheritance Factor), AIF (Attribute Inheritance Factor)) metrics, information hiding (MHF (Method Hiding Factor), AHF (Attribute Hiding Factor)) metrics, and polymorphism (POF (Polymorphism Factor), COF (Coupling Factor)) metrics. Abreu firmly suggested that metrics definitions and dimensions should be justified as they play important role in designing the object oriented metrics. Within the framework that, many metrics that are applied to traditional functional development are also applicable to object-oriented development, Rosenberg et al. [3] developed nine metrics for object-oriented system, from which three were traditional metrics viz. Cyclomatic Complexity (CC), Lines of Code (LOC), Comment Percentage (CP) and rest six metrics were same as CK metrics. They validated the six CK metrics at SATC and gave the relation between important object oriented software quality concepts, quality metrics and object oriented features as shown in Table 2 [3]. Amjan Shaiket. al. in [4] performed statistical analysis on the CK metric suite for Object oriented systems. They found that if properly used, metrics could lead to a significant reduction in cost of the overall implementation and quality improvement. L. Rosenberg et al. in [5] have identified five attributes for analysing design and code of the software. These are efficiency, complexity, understandability, reusability and testability/maintainability. Dr. Thapalyal, G. Verma in [6] performed an empirical study of two metrics – CBO,WMC of CK metric suite to extract the relationship of these metrics with defects.

3. Software Quality Factors

In reality object oriented development has proved its value for systems that must be maintained and modified. The concepts of object oriented design metrics are well established and many metrics relating to product quality have been developed and used. With object oriented analysis and design methodologies gaining popularity, it is time to start investigating object oriented design metrics with respect to software quality. Measuring quality in the early stage of software development is the key to develop high quality software. There must be a way to assess object oriented software complexity as early as possible in the development cycle [7]. McCall proposed a useful categorization of factors that affect software quality as shown in figure 1.

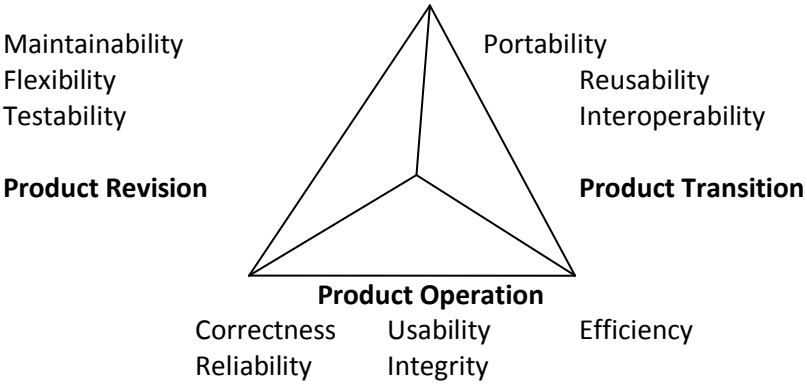

Figure 1: McCall's Quality factors [Source Pressman Fifth Edition]

4. Object Oriented Metrics

Chidamber and Kemerer (CK) [8] are the mostly referenced researchers. They defined six metrics viz., Depth of Inheritance Tree of a class (DIT), Number of Children of a class (NOC), Coupling Between Object Classes (CBO), Lack of Cohesion in Methods (LCOM), Weighted Methods per Class (WMC), Response sets for Class (RFC). CK metrics were defined to measure the complexity of the software in relation to their impact on quality attributes such as Reusability, Maintainability, Testability, Understand ability and Efficiency etc. Several studies have been conducted to validate CK metrics.

Table 1. Metric Suite of Chidamber and Kemerer

CK Metric	Definition
WMC	Number of methods of a certain class without inherited methods
RFC	Number of methods that can be performed by a certain class regarding a received message
LCOM	Number of disjunctive method pair of certain class
CBO	Number of coupling between a certain class and all other classes
DIT	Maximum depth of a certain class in an inheritance structure
NOC	Number of direct subclass of a certain class

CK metrics are aimed at assessing the design of object oriented system rather than implementation. This make them more suited to object-oriented paradigm as object-oriented design put great emphasis on the design phase of software system. The relation between important object-oriented software quality concepts, CK metrics and object-oriented (OO) features is given in Table 2 [5].

Table 2.Relationship among CK metrics, OO software Quality Concepts and OO Features

CK Metric	Concept/Factor	OO Feature
DIT	Reusability, Efficiency, Understandability, Testability	Inheritance
NOC	Reusability, Efficiency, Testability	Inheritance
CBO	Reusability, Efficiency	Coupling
LCOM	Reusability, Efficiency, Complexity	Cohesion
WMC	Maintainability, Understandability, Usability, Reusability	Class/Method
RFC	Understandability, Usability, Testability	Class/Method

5. Object-Oriented Design Quality Metrics

Our approach for evaluation and identification of testability of software is based on software quality factors and their related metrics. The aim of this work is to develop a model for the assessment of complexity that can be measured using these metrics. Some factors have been extensively recognized in OO like Inheritance, Coupling and Structure. We have identified four main metrics for the complexity evaluation in OO software.

It is concluded in our previous research paper that four out of six metrics WMC, NOC, CBO and RFC are suitable for complexity and quality measurement [2].

A. Inheritance Metrics

Number of children can be defined as the number of immediate subclasses of a class. NOC measures the breadth of a class hierarchy. The higher the value of NOC, the fewer the faults, which is desirable [18]. NOC, therefore, primarily evaluates efficiency, reusability, and testability [19]. Inheritance metrics is Number of Children (NOC).

B. Coupling Metrics

Coupling between objects [9] can be defined as no. of coupled classes within all classes. Low coupling is desirable for better design. In [10], a metric suite for OO is provided and coupling is considered as one of the main metrics in framework. Coady*et al.* [11] provides separate set of metrics for couplings in object oriented. Coupling metricis Coupling between Objects (CBO).

C. Structural Metrics

It is a combination of the sum of the complexities of all methods of a class and the set of methods that can potentially be executed in response to a message received by an object of that class [1].

- The large no. of methods in a class, the greater the potential impact on children. Classes with large no. of methods are likely to be more application specific, limiting the possibility of reuse. So WMC has negative impact or reusability of a class.
- The larger the no. of methods that can be invoked from a class through message, the greater the complexity of the class. So RFC has negative impact on reusability of a class.

Structural Metric are as follows: (i) Weighted Methods per Class (WMC) (ii) Response for a Class (RFC)

As, complexity of OO Software mainly depends on the metrics such as (i) Inheritance (ii) Coupling (iii) Structural. These metrics relates to the factors i.e. Reusability, Maintainability, Testability, Understandability and Efficiency. In order to access complexity of OOS, it is also very difficult to determine percentage of contribution of these metrics in complexity. To overcome these difficulties, we adapted fuzzy logic technique considering the four metrics as input variable and complexity as output variable.

6. Implementation of Fuzzy Logic

Fuzzy logic is a systematic technique to solve the problems that are very complex to understand quantitatively. It is a tool which deals with uncertainty and imprecision [13]. It is less dependent on historical data and fuzzy model can be built with less data [14, 15].

The fuzzy system accepts vague statements and imprecise data using the available membership functions and gives decisions as shown in Fig.2.

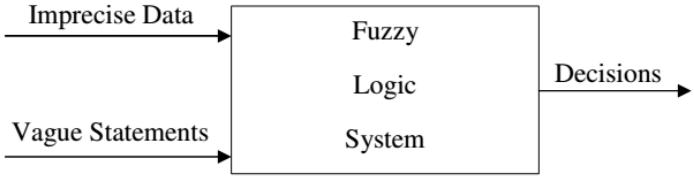

Figure 2: Fuzzy Logic System

The fuzzy model gives mapping from input to output. Architecture includes four different modules. The fuzzification module converts the crisp input values into fuzzy values. Fuzzy values are forwarded by an interface engine derived from rule base in the knowledge base given by domain experts. Finally, defuzzification module converts fuzzy data to crisp values. The fuzzy model architecture is shown in Fig. 3.

In this proposed work, Complexity of OO system is a measure of different factors: Reusability, Maintainability, Testability, Understand ability and Efficiency. These factors are independent to each other. These joined factors used in the measurement for the dependent variable i.e. complexity because we cannot directly measure the complexity.

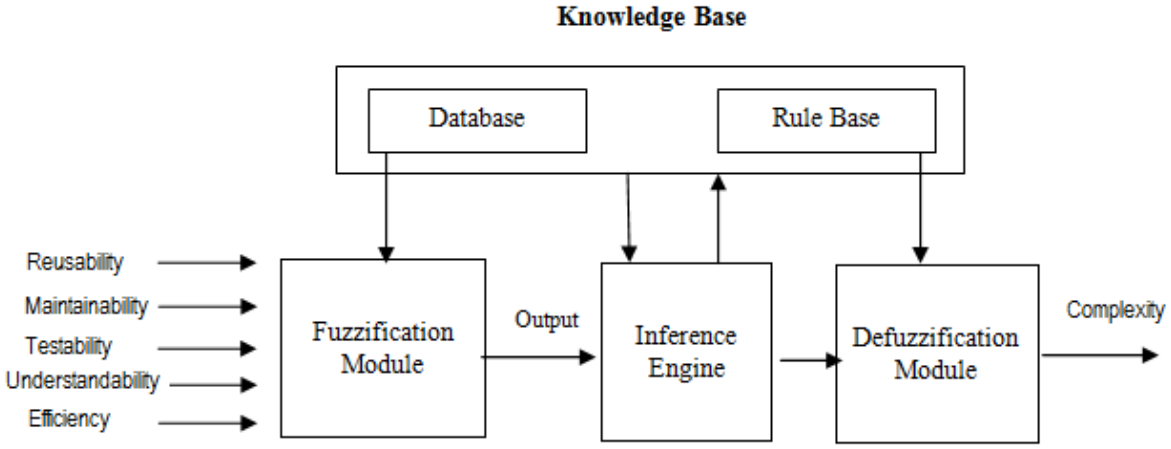

Figure 3: Fuzzy Model

The proposed fuzzy logic considers all these factors as inputs and finally gives a crisp value of complexity using rule base. All input values are categorized as low, medium and high. The output complexity is categorized as very low, low, medium, high and very high. A rule base is created using all feasible arrangements of inputs. Fuzzy Inference System (FIS) includes the following module which is shown in Fig. 4.

1) FIS Editor: It shows information related to fuzzy inference system to handle the complex problems for the system.
2) Membership Function Editor: It describes the shapes of all the membership functions related to every factor.
3) Rule Editor: It is used for editing the rules which determines the behavior of the problem.
4) Rule Viewer: It is used to see the rule base i.e. how an individual rule affects the results.
5) Surface Viewer: It is used to see the graph on the basis of given inputs and the output for the system.

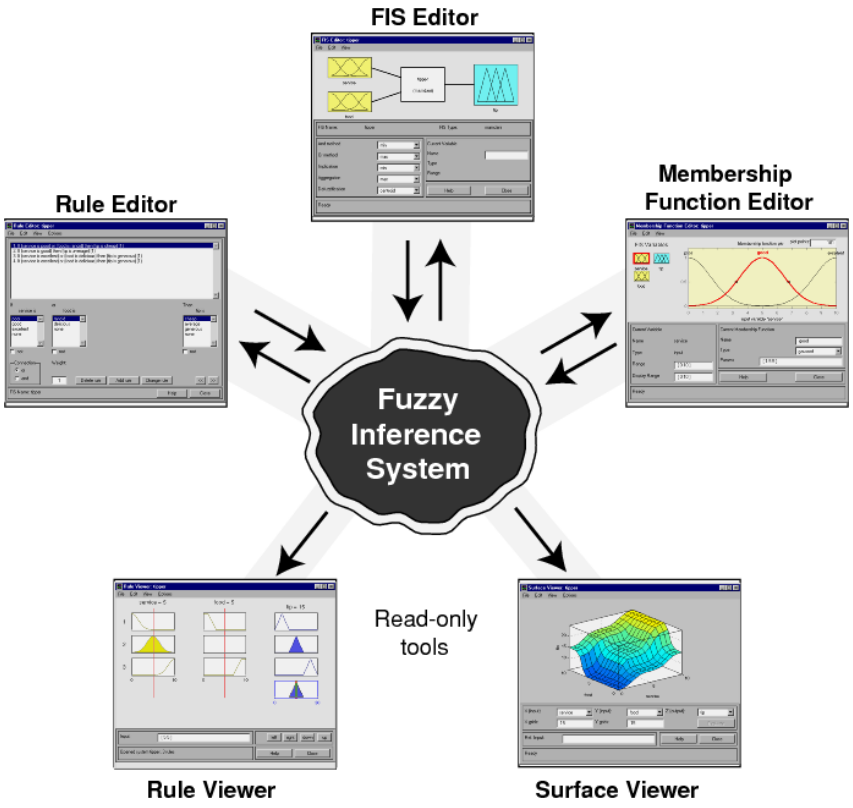

Figure 4: Fuzzy Inference System [16]

7. Analysis

All inputs are categorized into membership function i.e. low, medium and high and the software complexity as output is categorized as very low, low, medium, high and very high. Triangular membership function is used to categorize the inputs and output scaled between [0 1] scale. All 243

rules created and inserted in rule base, which represents all possible combinations of inputs i.e. 3^5 (243) sets.

The values for inputs, Low [ ], Medium [ ], High [ ] and For output, Very Low [ ], Low [ ], Medium [ ], High [ ], Very High [ ] may be find out in our new research paper with help of given proposed model.

8. Conclusion

This paper discusses complexity in relation to OO Software. It identifies the factors which affecting complexity and sets up a relationship on these factors for complexity. Proposed model based on five factors: Reusability, Maintainability, Testability, Understandability and Efficiency for accessing Software Complexity levels using AI techniques by Fuzzy Logic. Proposed model categorized inputs as low, medium and high and Complexity as a output which is categorized as very low, low, medium, high, and very high.

References

[1] Goel, B. M., Bhatia, P. K., 2013. ACM SIGSOFT Software Engineering Notes, ACM, New York, USA, Vol. 38 No. 4, July 2013, pp. 1-5.
[2] Bansal, M., Agarwal, C. P. 2014. Critical Analysis of Object Oriented Metrics in Software Development, 2014 Fourth International Conference on "Advanced Computing & Communication Technologies", IEEE Conference Publishing Services, R.G. Education Society, Rtk., 8-9 Feb., 2014 , pp. 197-201.
[3] J. Bansiya and C.G. Davis, 2002. A Hierarchical Model for Object-Oriented Design Quality Assessment, IEEE Transactions on Software Engineering, Vol. 28, No. 1.
[4] Boehm, B. W., Brown, J. R., and Lipow, M. L., 1976. Quantitative Evaluation of Software Quality. In IEEE Proceedings of the 2nd International Conference on Software Engineering, San Francisco, California, United States, pp: 592- 605.
[5] L. H. Rosenberg and L. Hyatt, 1997. Software Quality Metrics for Object- Oriented Environments, Crosstalk Journal.
[6] S. Jamali, 2006. Object Oriented Metrics, Sharif University of Technology.
[7] L. Rosenberg and L. Hyatt, 1995, Quality Metrics for Object-Oriented System Environments, NASA Technical Report.
[8] S. R. Chidamber and C. F. Kemerer, 1994. A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476–493.
[9] Aopmetrics project. http://aopmetrics.tigris.org/
[10] Chidamber S.R., Kemerer C.F., —A Metrics Suite for Object Oriented Design , Software Engineering, IEEE Transactions, Vol. 20, No. 6, 476–493, 1994.
[11] Coady Y., Kiczales G., —Back to the Future: A Retroactive Study of Aspect Evolution in Operating System Code , Published in proceedings of the 2nd International Conference on Aspect Oriented Software Development, ACM Press, pp. 50–59, New York, NY,USA, 2003.
[12] B. Kitchenham, “Software metrics in Software Reliaility Handbook [M],” London, New York: P. Rook. Elsevier Applied Science, 1990.
[13] Sivanandam, S.N., Sumathi, S., Deepa, S.N., —Introduction to Fuzzy Logic using MATLAB , Springer, Heidel-berg, 2007.
[14] MacDonell S.G., Gray A.R., Calvert J.M., —Fuzzy Logic for Software Metric Practitioners and Researchers , Published in Proceedings of the 6th International Conference on Neural Information Processing ICONIP, Perth, pp. 308-313, 1999.

[15] Ryder J., —Fuzzy Modeling of Software Effort Prediction , Published in Proceedings of IEEE Information Technology Conference, pp. 53-56, Syracuse, New York, 1998.
[16] http://www.mathworks.in/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html (lastaccessed on 12/02/14).
[17] M.H. Tang, M.H. Kao, and M.H. Chen, "An empirical study on object-oriented metrics [C]," Proceedings of 6th International Symposium on Software Metrics, 4-6 Nov 1999, pp.242-249.
[18] Henderson-sellers, B. 1996. Object-Oriented Metrics, Measures ofComplexity, rentice Hall.
[19] Bieman, J., and Karunanithi, S. 1993. Candidate reuse metrics forObject Oriented and Ada Software, In Proceedings of IEEE-CSFirst International Software Metrics Symposium.