A fuzzy approach for Integrated measure of object-oriented Software complexity

DR. K.K. AGGARWAL, DR. YOGESH SINGH, VANDANA GUPTA

Key Words - Complexity, Probability of Failure, Object Oriented

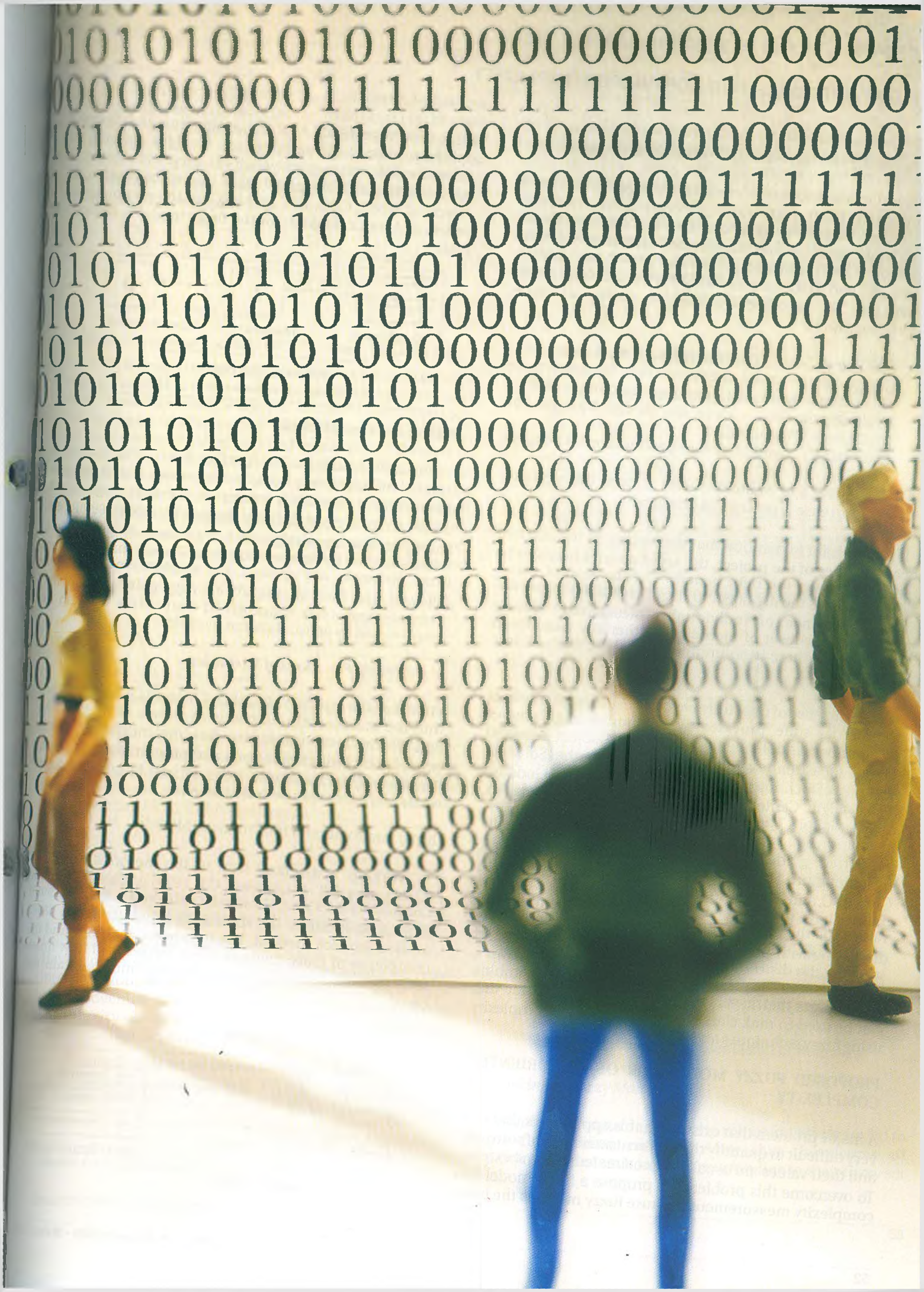
ABSTRACT

Many methods can be used to measure the complexity of a Object Oriented software, but none of them takes care of all the characteristics of the software. Various metrics used in measurement of OO software complexity are Weighted Methods per Class (WMC), Response for a Class (RFC), Lack of Cohesion of Methods (LCOM), Coupling Between Objects (CBO), Depth Inheritance Tree (DIT) and Number of Children (NOC), but none of them is alone sufficient to give an overall reflection of software complexity. Different metrics try to measure different characteristics of the software such as efficiency, complexity, understandability, reusability, testability and maintainability. Depending upon the characteristics of the software to be measured a subset of these metrics should be used. This paper proposes a fuzzy model for complexity measurement that integrates the effect of subset of these metrics i.e. WMC, RFC, CBO and DIT. The paper is about an approach to combine OO software metric values into a single overall value that can be used to rank classes according to their complexity. The approach uses fuzzy techniques and concepts, viz fuzzification of crisp metric values, inference and aggregation, defuzzification of fuzzy output etc.

INTRODUCTION

Software Complexity

Complexity has been used to characterize software. Banker et al. state, "Software complexity refers to the extent to which a system is difficult to comprehend, modify and test, not to the complexity of the task which the system is meant to perform; two systems equivalent in functionality can differ greatly in their software complexity" [D.D. Banker, S.M. Datar, D. Zweig (1989)]. Software complexity is defined in IEEE Standard 729-1983 as: "The degree of complication of a system or system component, determined by factors such as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, the types of data structures, and other system characteristics." According to McCall, a complexity relates to data set relationships, data structures, data flows and data being implemented and measures the degree of decision making logic within the system [J A McCall, P K Richards, G.F Walters (1977)].



Complexity Measurement of Object Oriented Systems

Object Oriented software development requires a different approach from traditional development methods, including metrics used to evaluate the software [3-10]. Chidamber and Kemerer [Chidamber and Kemerer, (1994)] have proposed the following 6 metrics - WMC, RFC, LCOM, CBO, DIT and NOC, for measuring the OO software quality attributes. The following OO software quality attributes can be measured:

* Efficiency -- are the constructs efficiently designed?
* Complexity -- could the constructs be used more effectively to decrease the architectural complexity?
* Understandability -- does the design increase the psychological complexity?
* Reusability -- does the design quality support possible reuse?
* Testability and maintainability -- does the structure support ease of testing and changes?

It is important to mention that depending on the size and complexities of the project, the set of metrics needs to be adjusted to get the right benefit. The decision of which ones to use is mostly down to the organizational practice and experience of individual professionals. Having selected the subset of metrics the other key issue faced today is that there is no unified measure of OO software complexity.

For the purpose of this study we are considering a subset of the metrics suite, provided by Chidamber and Kemerer [Chidamber and Kemerer, (1994)], for object oriented design measurement that are applied by the Software Assurance Technology Center (SATC) at NASA Goddard Space Flight Center [Linda H. Rosenberg, Ruth Stapko, Albert Gallo]. These metrics have been used in the evaluation of many NASA projects and empirically supported guidelines have been developed for their interpretation. The metrics used in this model are WMC, CBO, RFC, and DIT. These four metrics are selected as these metrics have the impact on the complexity. The threshold values for the individual metrics need to be derived by discussing with project managers and programmers and studying the distributions of the metrics collected over a period of time. The paper is about an approach to combine OO software metric values into a single overall value that can be used to rank classes according to their complexity using fuzzy techniques and concepts.

PROPOSED FUZZY MODEL FOR OBJECT ORIENTED COMPLEXITY

A major problem that exists with this approach is that it is very difficult to quantify the different attributes of software and their values are usually approximated to some extent. To overcome this problem we propose a fuzzy model for complexity measurement because fuzzy model is the best thing to represent such doubtful, ambiguous and divergin opinions. Fuzzy logic variables are used for defining the values for different metrics, which will serve as the inputs to our fuzzy model. This model takes into consideration the effects of WMC, CBO, DIT and RFC on the complexity of software. A block diagram for the fuzzy model is shown in figure1:

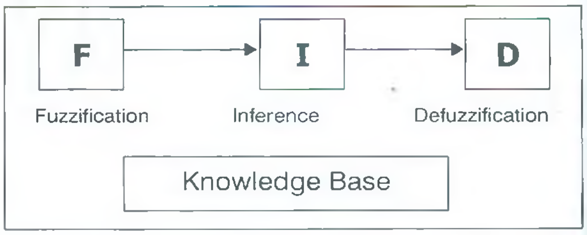


Figure : 1 A Fuzzy Model

It consists of 4 modules. The fuzzification module is the firs step in working of any fuzzy model, which transforms th crisp input(s) into fuzzy values. In the next stage, thes values are processed in fuzzy domain by inference engin based, on the knowledge base (rule base or productio rules) supplied by domain expert(s). During this stage, th fuzzy operators are applied, the implication process i followed, and then all outputs are aggregated. Finally th processed output is transformed from fuzzy domain t crisp domain by defuzzification module.

Membership Functions of Variables

First of all the inputs WMC, RFC, CBO and DIT are fuzzifie into the membership functions as shown in figures 2,3, and 5. The level values are just the indicative figure provided by project managers and programmers of NASA The threshold values need to be derived by collecting an analyzing data over a period of time.

Input parameters for the Proposed Model

* WMC is a weighted sum of the methods in a clas [Chidamber S.R. & Kemerer, C.F. (1991)]. The Cyclomati Complexity [McCabe, Thomas J. (1976) and B. Beizer (1990 is used to evaluate the minimum number of test case needed for each method. The number of methods and th complexity of those methods are a predictor of how muc time and effort is required to develop and maintain th class.

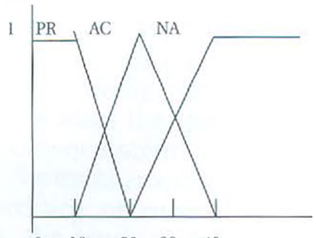


Figure 2 : Membership function for WMC

<=20	Preferred (PR)
20=40	Acceptable (AC)
>40	Not Acceptable (NA)

Table1 : Term set for WMC

* RFC is the cardinality of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class [Chidamber S.R. & Kemerer, C.F. (1991)]. The larger the number of methods that can be invoked from a class through messages, the greater the complexity of the class, complicating testing and debugging due to ripple effect.

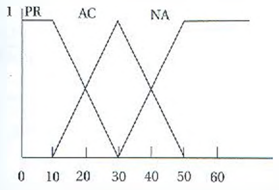


Figure 3 : Membership function for RFC

<=30	Preferred
30-50	Acceptable
>50	Not Acceptable

Table 2 : Term set for RFC

* CBO is a count of the number of other classes to which a class is coupled. It is measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends [Chidamber S.R. & Kemerer, C.F. (1991)]. Coupled classes must be bundled or modified if they are to be reused. A high CBO indicates classes that may be difficult to understand, reuse or maintain. The larger the CBO, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult. Low coupling makes the class easier to understand, less prone to errors spawning, promotes encapsulation and improves modularity.

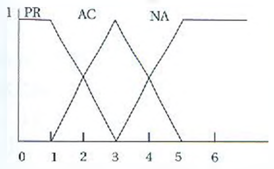


Figure 4 : Membership function for CBO

<=3	Preferred
3-5	Acceptable
>5	Not Acceptable

Table 3 : Term set for CBO

* DIT- The depth of a class within the inheritance hierarchy is the number of jumps from the class to the root of the class hierarchy and is measured by the number of ancestor classes. When there are multiple inheritances, use the maximum DIT. DIT of 0 indicates a "root".

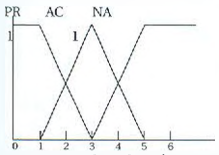


Figure 5 : Membership function for DIT

<=3	Preferred
3-5	Acceptable
>5	Not Acceptable

Table 4 : Term set for DIT

Output Variable/Parameter

Having defined the metrics and threshold values we need interpretation guidelines to assist in identifying those areas of code deemed to be more complex. The output variable complexity is defined to have 3 membership 3-membership functions- LOW, MEDIUM and HIGH.

Table 5: Term Set for Output Variable Complexity

# Of OO Metrics deviating from thres values hold	**Complexity**
<= 1	Simple class, low
1 – 2	More complex class, medium
> 2	High Complex class, high

Rule Base For the proposed Model

In the proposed fuzzy model we are considering four inputs each consisting of three terms therefore our rule base consists of 81 rules after considering all the possible combinations of inputs.

Suppose if a fuzzy model is having n inputs each consisting of m terms then the possible number of rules, say R, for this model can be calculated by considering the Cartesian product of all the input states.

R = M * M * M..upto n times

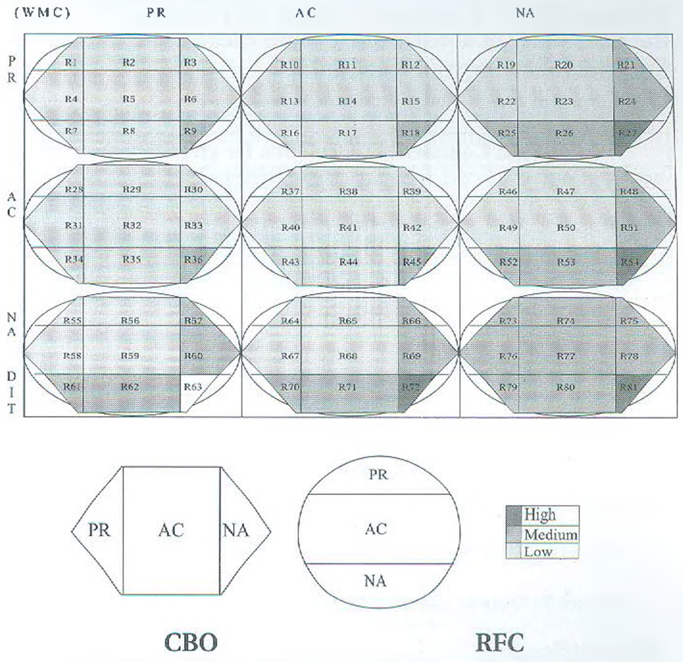
Or $R = M^n$

Our rule base can be represented in a tabular form consisting of 81 rows as below: -

Table 6: Rule Base of the Fuzzy Model

S. No	CBO	WMC	RFC	DIT	Complexity
1	PR	PR	PR	PR	LOW
2	AC	PR	PR	PR	LOW
3	NAC	PR	PR	PR	LOW
.					
.					
80					
81	NAC	NAC	NAC	NAC	HIGH

A pictorial representation of Rule Base consisting of 81 rules is shown in Figure 6: -


(WMC)
PR
AC
NA
PR
AC
NA DIT
R1 R2 R3 R4 R5 R6 R7 R8 R9
R10 R11 R12 R13 R14 R15 R16 R17 R18
R19 R20 R21 R22 R23 R24 R25 R26 R27
R28 R29 R30 R31 R32 R33 R34 R35 R36
R37 R38 R39 R40 R41 R42 R43 R44 R45
R46 R47 R48 R49 R50 R51 R52 R53 R54
R55 R56 R57 R58 R59 R60 R61 R62 R63
R64 R65 R66 R67 R68 R69 R70 R71 R72
R73 R74 R75 R76 R77 R78 R79 R80 R81
PR AC NA
PR AC NA
High
Medium
Low
CBO
RFC
Figure 6: Pictorial Representation of Rule Base

Working of Fuzzy Model

First of all the input data i.e. crisp values for inputs are fed to the fuzzification module. After that the degree of participation of each input corresponding to their fuzzy set is determined [K.K. Aggarwal, Yogesh Singh, Jitender Kumar Chhabra (2002)]. Then inference engine starts processing of these inputs on the basis of which some rules get fired which gives the fuzzified output. For selection of rules all the input states are considered simultaneously i.e. these inputs are connected by AND operator therefore we are using min fuzzy operator to find the degree of membership of firing [Marek J. Patyra, Janos L. Grantner, Kirby koster (1996) and J. Patyra, D. M. Mlynek (Editors) (1996) and George J. Klir, Bo Yuan (1995)]. If more than 1 rule get fired then to aggregate the effects of all the fired rules max aggregation operator is used and after that defuzzification of the fuzzified output is done .We are using the center of gravity method for this purpose.

Sample Output Computation For The Model

Suppose we have the following crisp inputs to the model:

WMC = 10, DIT = 5, RFC = 20, CBO = 1

These inputs are fed to the fuzzification module and after the fuzzification of the given values we find that WMC = 10 belongs to fuzzy set preferred (PR) with membership grade 1, DIT = 5 belongs to the fuzzy set not acceptable (NA) with membership grade 1, RFC = 45 belongs to fuzzy set acceptable (AC) with membership grade of 0.25 and to fuzzy set not acceptable (NA) with membership grade of 0.75 and CBO = 1 belongs to fuzzy set preferred (PR) with the possibility of a degree of 1. With these input values we find that following rules get fired:

Table 7: Complexity Calculation for given input

WMC (10)	RFC (45)	CBO (1)	DIT (5)	COMPLEXITY	Membership grade of complexity
PR	AC	PR	NA	LOW	Min (1,0.25,1,1) = 0.25
PR	NA	PR	NA	MEDIUM	Min (1,0.75,1,1) = 0.75

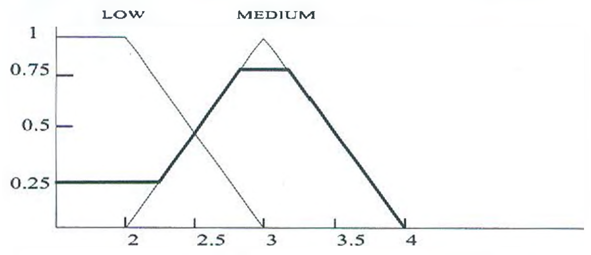


Figure 7: Output variable Complexity Aggregation

Defuzzification

After getting the fuzzified outputs as shown above, we defuzzify them to get the crisp value of the output variable complexity [Ronald R. Yager, D.P. Filev (1995) and Hellendorn, C. Thomas (1995)]. For this we are finding the Centre Of Gravity (COG) of the above fuzzy output as follows:

$$COG = \int \mu \, x \, dx / \int \mu dx$$

$$COG = \frac{\left(\int_0^{2.25} 0.25x dx \quad \int_{2.25}^{2.75} yx dx + \int_{2.75}^{3.25} 0.75x dx + \int_{3.25}^{4} yx dx \right)}{\left(\int_0^{2.25} 0.25 dx + \int_{2.25}^{2.75} y dx + \int_{2.75}^{3.25} 0.75 dx + \int_{3.25}^{4} y dx \right)}$$

$$COG = \frac{\left(\int_0^{2.25} 0.25x dx + \int_{2.25}^{2.75} (mx + c) x dx + \int_{2.75}^{3.25} 0.75x dx + \int_{3.25}^{4} (mx + c) x dx \right)}{\left(\int_0^{2.25} 0.25 dx + \int_{2.25}^{2.75} (mx + c) dx + \int_{2.75}^{3.25} 0.75 dx + \int_{3.25}^{4} (mx + c) dx \right)}$$

$$COG = \frac{\left(0.25[x^2/2]_0^{2.25} + \int_{2.25}^{2.75} (x - 2) x dx + 0.75[x^2/2]_{2.75}^{3.25} + \int_{3.25}^{4} (4 - x) x dx \right)}{\left(0.25[x]_0^{2.25} + \int_{2.25}^{2.75} (x - 2) dx + 0.75[x]_{2.75}^{3.25} + \int_{3.25}^{4} (4 - x) dx \right)}$$

$$COG = 2.2996456$$

We observed the effect of these rules by simulating the proposed fuzzy system with the help of MATLAB- Fuzzy Tool Box [Roger Jang, Ned Gulley (1995)]. For the above given inputs the value of complexity comes out to be 2.3035 which is very near to our calculated value. One sample surface view of the simulated fuzzy model is shown in the figure 8.

The integrated approach gives a true picture of the software complexity. This can prove to be a handy tool for manager to rank the Object Oriented classes based on their complexities. Also the complexity measurement will aid in decisions related to testing, for example, more complicated class demands for thorough testing and should be given priority over the less complicated ones. Since we are considering an integrated approach for indicating the complexity therefore the effect of all the parameters can be adjudged simultaneously.

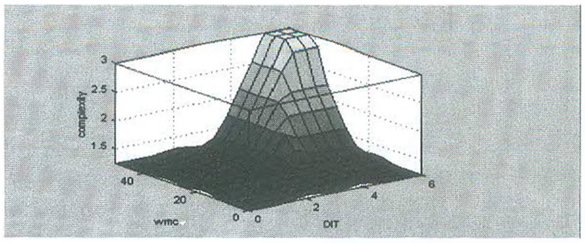


Figure 8: Surface View of the Simulated Fuzzy Model

CONCLUSION

This paper has presented a single unified measure of software complexity of Object Oriented software. The proposed model measures the complexity of object-oriented systems based on the four important object oriented metrics WMC, RFC, CBO and DIT. This model integrates the effects of different parameters of object-oriented systems, which can affect the complexity and is not biased towards one particular aspect of the system. This model can be useful in deciding about the order of complexities of different classes which in further can help in determining the risk associated with the classes of different complexity level and we can prioritize the classes on the basis of risk associated with them for regression testing.

REFERENCES

[1] B. Beizer (1990), Software Testing Techniques. Van Nostrand Reinhold, New York.

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli (1991), Fundamentals of Software Engineering. Prentice Hall, Upper Saddle River, NJ 07458, 1st edition.

[3] Chidamber and Kemerer, (1994), *"A Metrics Suite for Object-Oriented Design"*, IEEE Transactions on Software Engineering, Vol. 20, No. 6, June, pp. 476 - 493.

[4] Chidamber S.R. & Kemerer, C.F. (1991), *"Towards a Metrics Suite for Object Oriented Design"* Proc. OOPSLA.

[5] D.D. Banker, S.M. Datar, D. Zweig (1989), *"Software Complexity and Maintainability"*, Proceedings of Tenth International Conference on Information Systems, Boston, Dec 4-6, p247-255.

[6] Edward V. Berard. Metrics for Object-Oriented Software Engineering, The Object Agency, Inc.

[7] George J. Klir, Bo Yuan (1995), Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall of India, New Delhi.

[8] Hellendorn, C. Thomas (1995), *"On Quality Defuzzification: Theory and Application Example"*, Fuzzy Logic and Applications to Engineering, Information Sciences and Intelligent Systems Edited by Z. Bien, K.K. Min, Kluwer Academic Publication, pp 167-176.

[9] J A McCall, P K Richards, G.F Walters (1977), *"Factors in Software Quality"*, vol 1-III, US Rome Air Development Center Reports NTIS AD/A-049014, 015, 055, National Technical Information Service, US Department of Commerce.

[10] J. Patyra, D. M. Mlynek (Editors) (1996), Fuzzy Logic: Implementation and Applications, John Viley & Sons Ltd. And B.G. Teubner.

[11] K.K. Aggarwal, Yogesh Singh, Jitender Kumar Chhabra (2002), *"An Integrated Measure ofOf Software Maintainability"*, Proceedings of Annual Reliability and Maintainability Symposium- International Symposium on Product Quality and Integrity RAMS- 2002,Seatle Westin U.S.A, Jan 28-31, p.235-241.

[12] K.K. Aggarwal, Yogesh Singh, Jitender Kumar Chhabra (2002), *"Towards Complexity Measurement of Object Oriented Software"*, All India Seminar on Challenges Ahead with Information Technology CAIT-2002, SLIET, Longowal, Jan 19-20, p-90-92.

[13] K.K. Aggarwal, Yogesh Singh, Jitender Kumar Chhabra (2003), *"A Unified Measure of Object-Oriented Software Complexity"*, Journal of the CSI, Communicated, May.

[14] Linda H. Rosenberg, Ruth Stapko, Albert Gallo, *"Risk-based Object Oriented Testing"*, NASA GSFC SATC NASA, Unisys SATC NASA, Unisys Code 302 Code 300.1

[15] Marek J. Patyra, Janos L. Grantner, Kirby koster (1996), Digital Fuzzy Logic Controller: Design and Implementation, IEEE Transactions on Fuzzy Systems, Vol. 4 ,No4, No. 4, Nov. , pp 439-459.

[16] McCabe, Thomas J., "A Complexity Measure (1976)" IEEE Transactions on Software Engineering SE-2, pp 308-320.

[17] N. Wilde and R. Huitt (1992), *"Maintenance support for object-oriented programs,"* IEEE Trans. on Soft-ware Eng., Vol. 18, No. 12, pp. 1038 - 1044, December.

[18] Pfleeger, S.L. and Palmer, J.D. (1990) *"Software Estimation for Object Oriented Systems,"* Int'l. Function Point Users Group Fall conference, San Antonio TX.

[19] Roger Jang, Ned Gulley (1995), *"Fuzzy Logic Toolbox for MATLAB, User's Guide"*,", The Math Works Inc., USA, Jan.

[20] Ronald R. Yager, D.P. Filev (1995), *"Defuzzification with Constraints"*, Fuzzy Logic and Applications to Engineering, Information Sciences and Intelligent Systems, Edited by Z. Bien, K.K. Min, Kluwer Academic Publication, pp 157-166.

[21] Rosenberg, Linda, and Gallo, Albert (1999), *"Implementing Metrics for Object Oriented testing"*, Practical Software Measurement Symposium.

[22] Stale Amland (2000), *"Risk Based Testing and Metrics: Risk analysis fundamentals and metrics for software testing including a financial application case study"*, The Journal of Systems and Software, Vol. 53, pp. 287-295.