An Integrated Measure of Software Maintainability

Krishan K. Aggarwal•GGS Indraprastha University•Delhi.
Yogesh Singh•GGS Indraprastha University•Delhi.
Jitender Kumar Chhabra•Regional Engineering College•Kurukshetra.

Key Words: Software maintainability, software measurement, fuzzy model, rule base representation, source-code readability, documentation quality, understandability, Fog Index.

SUMMARY AND CONCLUSIONS

For large software systems, the maintenance phase tends to have comparatively much longer duration than all the previous life-cycle phases taken together, obviously resulting in much more efforts. A good measure of software maintainability can help better manage the maintenance phase effort. Software maintainability cannot be adequately measured by only source code or by documents. The readability and understandability of both, source code and documentation should be considered to measure the maintainability. This paper proposes an integrated measure of software maintainability. The paper also proposes a new representation for rule base of fuzzy models, which will require less space for storage and will be efficient in finding the results in the simulation.

The proposed model measures the software maintainability based on three important aspects of software- Readability of Source Code (RSC), Documentation Quality (DOQ), and Understandability of Software (UOS). Keeping in view the nature of these parameters, a fuzzy approach has been used to integrate these three aspects. A new efficient representation of rule base has been proposed for fuzzy models. This integrated measurement of software maintainability, which to our knowledge is first attempt to quantify integrated maintainability, is bound to be better than any other single parameter maintainability measurement approach. Thus the output of this model can advise the software project managers in judging the maintenance efforts of the software.

1. SOFTWARE MAINTENANCE

Every software program needs to be changed to meet the changing requirements of users and customers during its lifetime. It is generally impractical and uneconomical to produce software, which does not need to be changed. Introduction of new hardware also necessitates the change in the software. The process of changing software after it has been delivered and is in use is called software maintenance (Ref. 1). This is a task that every development group has to face when the software is delivered to the customer's site, installed and is operational. Some software may be maintained for several decades. Initially, aim of the software development was to write working programs. Then the aim shifts to write good quality programs, but lastly the aim is to write good maintainable programs. It has been reported that amount of effort spent on maintenance is between 65% and 75% of total software development and support efforts (Ref. 2). Despite the fact that software maintenance is very important and challenging task, it is poorly managed. One reason for poor management is the lack of a good measure of software maintainability. The fundamental reality that "you can not control what you can not measure" highlights the importance of a good measurement of software maintainability.

2. SOFTWARE MAINTAINABILITY

Delivery or release of the software inaugurates the maintenance phase of the software. Metrics proposed like Software Maturity Index (SMI) for measurement of software maintainability take into account the modules being changed/added/removed (Ref. 3). The readability of source code or quality of various documents greatly influences software maintainability, which is not really considered today by any of the available metrics. Truly speaking software maintainability is an integrated measure of many characteristics of the software like readability of source code, documentation quality and understandability of the software.

3. FACTORS AFFECTING MAINTAINABILITY

Software is maintained through the integrated use of source code and documents. Source code readability and quality of documentation should be taken into account while measuring the software maintainability.

Following three factors are proposed for measurement:

3.1 Readability of Source Code (RSC)

Any type of maintenance will ultimately result in changing the source code. Modifying source code written by some one else is a tedious task. Understanding of just a thousand lines source code is almost impossible if source code is not well supported by meaningful comments. So readability of the source code can be estimated by finding percentage of comment lines in total code. A new factor Comments Ratio (CR) is defined as $CR = LOC/LOM$

Where LOC refers to total lines of code (including comments lines), and LOM represents total lines of comments in the code. Obviously lower the ratio, better is readability.

3.2 Documentation Quality (DOQ)

The quality of the documentation associated with a software product is as important as the quality of the code itself. Metrics to assess readability of documents have been developed. The best known of these metrics is Gunning's Fog Index, which is a measure of the readability of a passage of text (Ref. 4). The Fog Index is based on length of sentences and the number of difficult words where difficulty of a word is based on the number of syllables in the word. Lower value of Fog Index indicates better quality of the documentation while higher value is reflection of poor documentation quality.

3.3 Understandability of Software (UOS)

Software is a collection of various documents and source code. The cohesiveness among the documents and source code is very important. If the software developer cannot correlate the documentation and source code, maintenance is going to be very difficult and erroneous. Laitnen has suggested a good tool to measure the understandability of software documents (Ref. 5). The measurement is on basis of use of symbols in source code and other documents. The method defines a new term - language of the software, which now contains all symbols used in the software excluding the reserved words of the source language. If the language of the source code and language of the documents are closely related, then understandability is high (Ref. 5).

4. PROPOSED MODEL

All of the above mentioned measures assess different properties and characteristics of software maintenance. CR concentrates only on count of comments of a program. Meaningfulness of comments cannot be judged by count alone. Documentation quality focuses only on documents. If documentation is good but source code is poorly written, still software cannot be maintained. Understandability of the software tries to estimate the correlation among the style of writing of source code and corresponding documents. All these three measures evaluate entirely different characteristics of the same software. All three parameters can be measured easily using some automated tools, but an overall indicator of software maintainability is desirable, which considers all three aspects and their relative impact on the maintainability of the software. So an integrated approach for measurement of software maintainability should be used. All three measures are quite subjective in nature and thus some tool is needed, which not only integrates these three factors, but also is capable of handling their subjective nature. As, a fuzzy model is the best choice for managing ambiguous, doubtful, contradicting and diverging opinions, we propose a fuzzy model for an integrated software maintainability measure that takes into account the effect of RSC, DOQ, and UOS.

A block diagram for the fuzzy model is shown in Figure 1. It contains four modules. The fuzzification module transforms the crisp input(s) into fuzzy values. These values are then processed in fuzzy domain by inference engine based on the knowledge base (rule base or production rules) supplied by the domain expert(s). Finally the processed output is transformed from fuzzy domain to crisp domain by defuzzification module (Ref. 6-10).

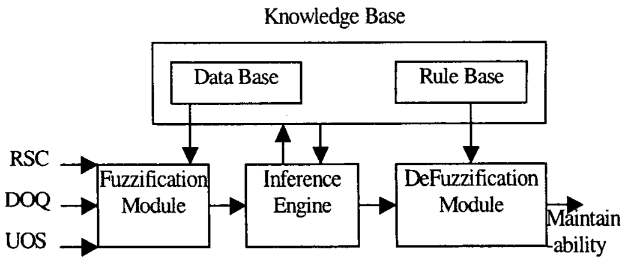

Figure 1: Working of a Fuzzy Model

4.1 Working of the Fuzzy Model

First of all, crisp values of inputs are taken. Comments Ratio (CR) is used to judge the Readability of Source Code (RSC). Quality of Documentation is judged using Fog index. Number of symbols used in the software language help in deciding the Understandability of Software (UOS). Now the degree to which the inputs belong to each of the appropriate fuzzy sets is determined. Based on these fuzzy inputs, some rules get fired. As various inputs are connected via *And* operator in the antecedents of rules, MIN fuzzy operator is applied to find the degree of membership of firing. Out of the three commonly used inference mechanisms - mamdani-style, larsen-style, and sugeno-style, *Mamdani-style* inference has been used in this model, being the most commonly used fuzzy inference methodology (Ref 6-10). MAX aggregation is used to integrate effect of all rules fired. Now centroid technique of defuzzification is applied to get a crisp value of maintainability on scale of 0 to 10 (Ref. 10-14). Lower value of maintainability reflects good maintainability, while higher value indicates poor maintainability.

In order to fuzzify the inputs, the following membership functions for the RSC, DOQ, and UOS are chosen. The base variable RSC has been divided into 3 states (linguistic variables) i.e., good, avg, and, poor as shown in Figure 2 below. RSC is measured using Comment Ratio (CR) in the source code.

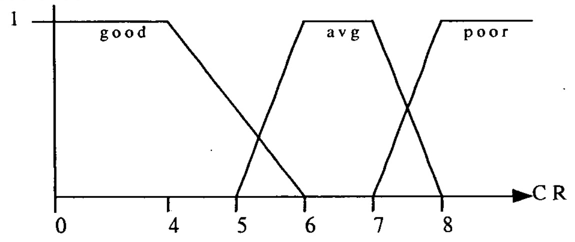

Figure 2 : Input Variable RSC

The base variable DOQ is measured using Fog Index (Ref.4)

FOG =0.4 * [(no. of words/no. of sentences) + %age of words with 3 or more syllables]

The acceptable range of Fog Index for standard, difficult, very difficult has been used to define 3 membership functions of DOQ high, med, and low as shown in Figure 3 (Ref. 4).

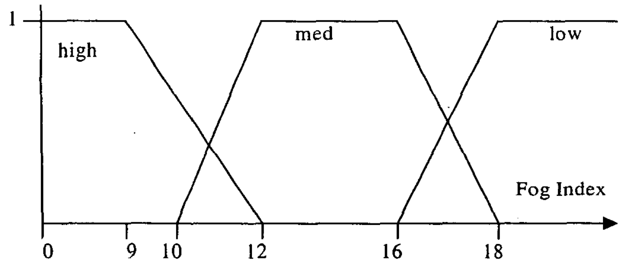

Figure 3: Input Variable DOQ

Similarly the input base variable UOS has been divided into 3 parts i.e., more, moderate, and less as shown in Figure 4 and is measured by counting number of symbols in the source code and in the documents. The understandability is estimated to be good if symbols used in the source code and documents are natural. Assuming natural symbols, count of symbols in range of 400-600 reflects good understandability and higher ranges of the count indicate decreasing understandability (Ref. 5).

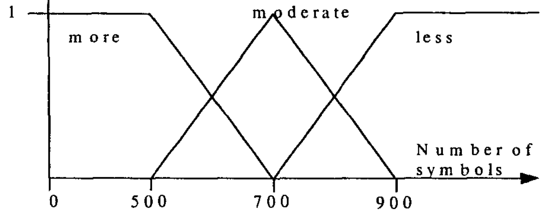

Figure 4: Input Variable UOS

The output variable is Maintainability and it has 4 membership functions very_good, good, avg, and poor as shown in Figure 5.

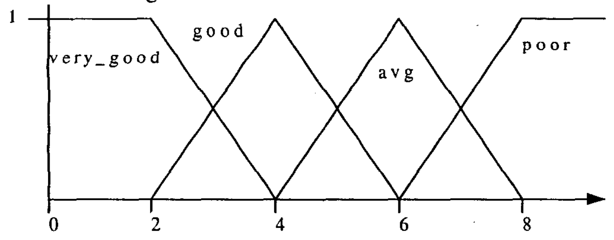

Figure 5:Output Variable Maintainability

4.2 Efficient Representation of the Rule Base for Fuzzy Models

After the fuzzification of input data, processing is carried out in fuzzy domain. 27 rules have been created for carrying out this processing. The rules are shown with help of Figure 6. In Figure 6, the output of every rule is also indicated as very_good/good/avg/poor.

Whenever a fuzzy model is to be simulated, the rule base is usually stored as

1. If (RSC is good) and (DOQ is high) and (UOS is more) then (maintainability is very_good)
2. If (RSC is avg) and (DOQ is high) and (UOS is more) then (maintainability is good)
3. If (RSC is poor) and (DOQ is high) and (UOS is more) then (maintainability is avg)

...

27. If (RSC is poor) and (DOQ is low) and (UOS is less) then (maintainability is poor).

The existing representation of the rule base is quite inefficient. It will take lot of storage space. Moreover, this type of representation will take more time during simulation in finding the rules, which get fired based on a particular set of inputs. Traditional method of finding the rules getting fired, during simulation, will sequentially check each rule and test whether that rule gets fired or not. This type of mechanism will be obviously time consuming. Before working on this specific problem, a general method of efficient representation of rule base is proposed here. In the new proposed representation of the rule base, there is no need to store membership function of any input. Only the output membership function corresponding to every rule need to be stored. A relation is established between every combination of membership functions of inputs and corresponding applicable rule number with help of weightages. There is no restriction either on number of inputs, or on number of membership functions of any input. However the method requires following conditions to be satisfied for its proper working:

1. Rules corresponding to all combinations of membership functions of various inputs must be considered. It means that if there are n inputs each having M_i number of membership functions, the method will require all $M_1 * M_2 * \dots M_N$ rules.
2. Rules must be specified in a specific order. Rule number 1 will be corresponding to the first membership function of every input. Rule 2 will be corresponding to second membership function of input 1, and first membership function of all remaining inputs. Once all membership functions of input 1 get considered, next rule will be corresponding to second membership function of second input and first membership function of all other inputs. Thus all rules get listed in this order and last rule will be corresponding to last membership function of all inputs.

This method will store only the output membership functions of all rules. The outputs must be stored in a specific order of rule numbers, as mentioned in condition 2 above. Every membership function of each input is assigned a weightage, which is stored along with other characteristics of each membership function. These weightages will be used to find the rule numbers, which will get fired for a particular set of input values. If a fuzzy model consists of N inputs and input i has got M_i number of membership functions, then total number of possible rules will be

$$Total\_Rules = \prod_{i=1}^{N} M_i \qquad \dots(1)$$

As already specified, the method requires output of all rules to be specified in a sequence such that all membership functions of first input are considered one by one with first membership function of all other inputs and so on for all membership functions of every input. The weightage for each of the membership function can be represented as W_{ij} , where i represents ith input and j represents jth membership function of ith input. The weightage can be defined mathematically as:

$W_{1,1} = 1$

$W_{i,j} = W_{i,(j-1)} + W_{i,1} \qquad \text{if } j > 1$

$W_{i,1} = W_{(i-1),M_{i-1}}$ if $i > 1$ (2)

Where $W_{(i-1),M_{i-1}}$ represents weightage of the last membership function of the (i-1)th input.

Weightage of first membership function of input 1 will be always one. For a particular input (i.e. in a particular row), weightage of second membership function onwards (i.e. from second column onwards of that row) is found by adding weightage of first membership function into weightage of previous membership function of the same input. First membership function of subsequent inputs will have weightage equal to weightage of last membership function of the previous input.

These weightages of various membership functions of all inputs are stored along with definitions of those membership functions.

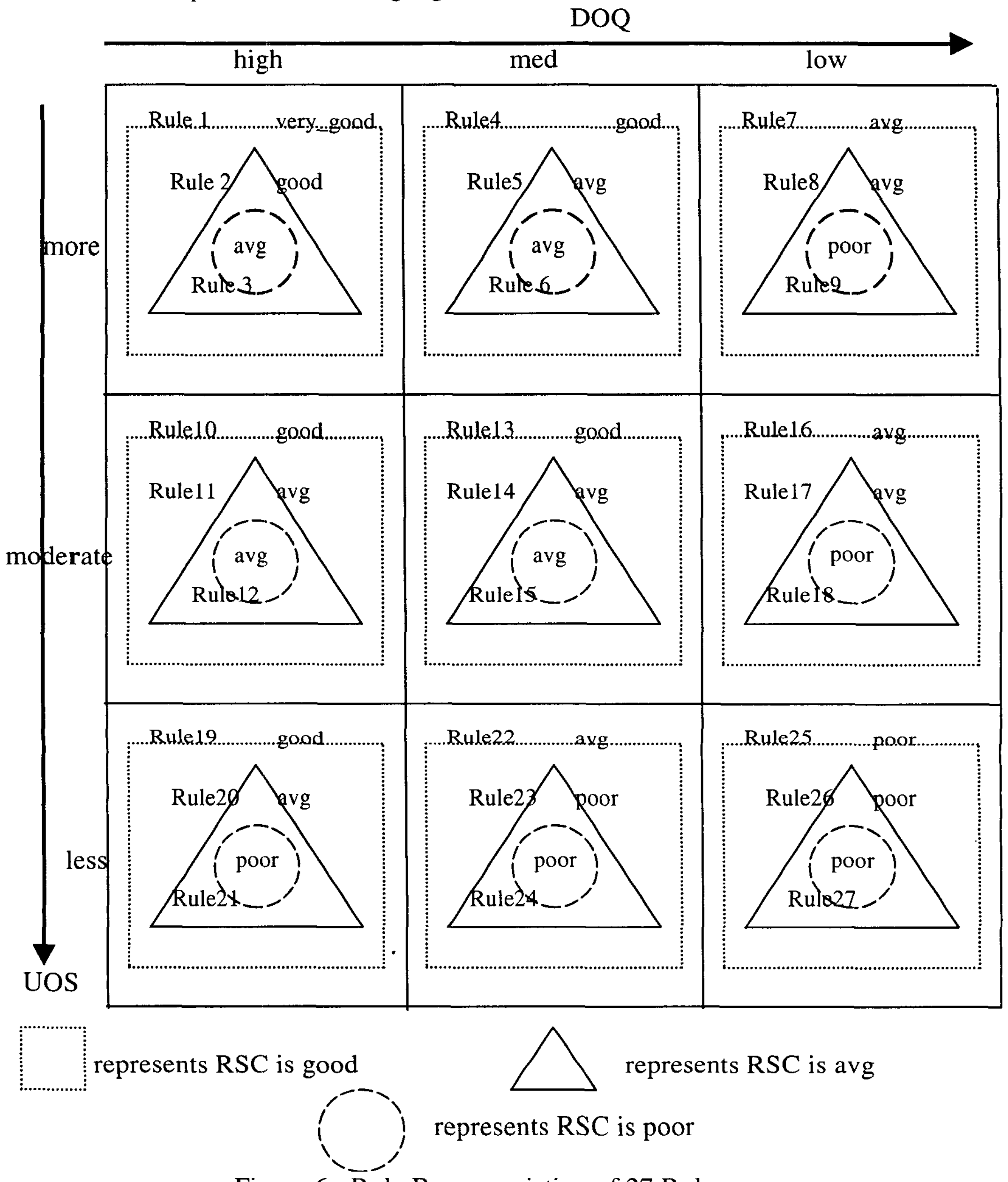

Figure 6 : Rule Base consisting of 27 Rules

When a set of inputs is applied to the fuzzy model, every input will belong to one or more membership functions. Possible rules, which should get fired, will be obviously corresponding to all combinations of applicable membership functions of the inputs. For a particular combination of membership functions of each input, the rule number, which will get fired, is found as:

$$Rule\_number = \sum_{i=1}^{I} W_{i,k} - \sum_{i=1}^{I} W_{i,1} + 1 \qquad (3)$$

Where $W_{i,k}$ denotes the weightage of the applicable membership function (say k) of ith input.

All rule numbers getting fired corresponding to various combinations of membership functions can be found by using (3) by using weightages of the membership functions considered in that combination.

This representation of the rule base and firing gives two advantages:

1. It will require less space for rule base storage. In the traditional representation, membership function of each input and membership function of the output is stored for each rule. But in this representation, the membership function of output only is stored. This representation needs some storage for storing weightage of each membership function of inputs and output. But still overall storage needed will be quite less. Let us assume that storage of one membership function in a rule takes 1 unit and let the fuzzy model have N inputs, each having M_i ($\forall$ i $\in$ 1..N) number of membership functions, then
   Units of Space Saved by new representation From Rule Base $= N * \text{Total\_Rules}$
   $= N * M_1 * M_2 * \ldots * M_N$
   Units of Space needed for storing the weightages assigned $= M_1 + M_2 + \ldots + M_N$
   Effective Units of Space saved =
   $(N * M_1 * M_2 * \ldots * M_N) - (M_1+M_2+\ldots+M_N)$
2. For any particular set of inputs during simulation, the rule base need not be searched sequentially to find out which rules really fire. Sequential search is very time consuming. Using this method, the rules applicable for firing are found in unit time by using the above-mentioned formula. Thus performance of implemented system will improve a lot because of faster firing of rules.

This efficient representation is now applied to the fuzzy model of software maintainability and its working is discussed with the help of an example.

4.3 Measurement of Software Maintainability:

The fuzzy model of software maintainability discussed above consists of 3 inputs, each having 3 membership functions. Total number of rules for this model is calculated using equation (1)

$$\text{Total\_Rules} = 3*3*3 = 27$$

The outputs corresponding to these 27 rules (written in specific order) are shown in Figure 6. In the new proposed representation, only those outputs are stored without explicitly storing membership functions of every input.

Rule No	Maintain -ability	Rule No	Maintain -ability	Rule No	Maintan -ability
1	very_good	2	Good	3	Avg
4	good	5	Avg	6	Avg
7	Avg	8	Avg	9	Poor
10	Good	11	Avg	12	Avg
13	Good	14	Avg	15	Avg
16	Avg	17	Avg	18	Poor
19	Good	20	Avg	21	Poor
22	Avg	23	Poor	24	Poor
25	Poor	26	Poor	27	Poor

The weightages for each of the membership function of three inputs are computed using equation (2), and are shown below.

	Membership Function 1	Membership Function 2	Membership Function 3
RSC	1	2	3
DOQ	3	6	9
UOS	9	18	27

Once the membership function of input 1, input 2, and input 3 is known, the rule number(s) which will get fired can easily be computed using equation (3). Let us consider the values of inputs such that RSC belongs to avg and poor, DOQ belongs to med, and UOS belongs to more and moderate. Total number of rules, which will get fired corresponding to this input set, will be 4. First rule will get fired corresponding to RSC being avg, DOQ being med, and UOS being more. Second rule getting fired corresponds to RSC being poor, DOQ being med, and UOS being more. RSC as avg, DOQ as med, and UOS as moderate cause third rule being fired, while RSC being poor, DOQ being med, and UOS being moderate cause the firing of fourth rule. The rule numbers of these four rules are found using eqn. (3).

Rule_Number 1 = 2(weightage of avg of RSC) + 6(weightage of med of DOQ) + 9(weightage of more of UOS) - 13(1+3+9) + 1 = 5

Rule_Number 2 = 3(weightage of poor of RSC) + 6(weightage of med of DOQ) + 9(weightage of more of UOS) - 13(1+3+9) + 1 = 6

Rule_Number 3 = 2(weightage of avg of RSC) + 6(weightage of med of DOQ) + 18(weightage of moderate of UOS) - 13(1+3+9) + 1 = 14

Rule_Number 4 = 3(weightage of poor of RSC) + 6(weightage of med of DOQ) + 18(weightage of moderate of UOS) - 13(1+3+9) + 1 = 15

It can be verified that the rule numbers found above are actually applicable rules in the rule base represented in Figure 6. For example, first rule getting fired corresponds to RSC as avg, DOQ as med, and UOS as more, and the rule number computed is 5. If we look at rule number 5 of the rule base, it is found that this rule will actually fire corresponding to these inputs. Similarly rule number 6, 14, and 15 are the actual rules, which will get fired for the corresponding set of inputs.

4.4 Output Computation for the Model:

Let us say we have the following inputs to the model: RSC= 5.5, DOQ=5, UOS= 350. When these inputs are fuzzified we find that RSC = 5.5 belongs to fuzzy set good with membership grade of 0.25 and to fuzzy set avg with membership grade of 0.5. DOQ = 5 belongs to fuzzy set high with membership grade of 1, and UOS = 350 belongs to fuzzy set more with membership grade of 1. With these input values we find that rule numbers 1 and 2 fire.

During composition of these rules we get the following

Min (0.25, 1, 1) = 0.25 Min (0.5,1, 1) = 0.5

When these two rules are implicated, we find that first rules gives maintainability value very_good to an extent of 0.25 and second rule gives the maintainability value good to the extent 0.5 (Ref. 5-9). This is shown below in Figure 7.

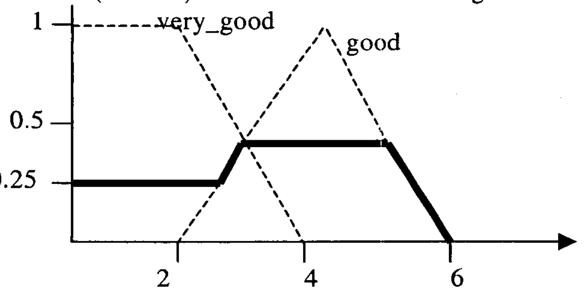

Figure 7: Output Computation of Maintainability

Defuzzification: Defuzzification of the above output can be obtained by finding the Center of Gravity of the above fuzzy output (Ref. 13-14).

$$\text{Maintainability} = \frac{\int_0^{2.5} .25x\,dx + \int_{2.5}^{3} (mx+c)x\,dx + \int_3^5 0.5x\,dx + \int_5^6 (mx+c)x\,dx}{\int_0^{2.5} .25\,dx + \int_{2.5}^{3} (mx+c)\,dx + \int_3^5 0.5\,dx + \int_5^6 (mx+c)\,dx}$$

$$= \frac{\int_0^{2.5} .25x\,dx + \int_{2.5}^{3} (0.5x-1)x\,dx + \int_3^5 0.5x\,dx + \int_5^6 (-0.5x+3)x\,dx}{\int_0^{2.5} .25\,dx + \int_{2.5}^{3} (0.5x-1)\,dx + \int_3^5 0.5\,dx + \int_5^6 (-0.5x+3)\,dx}$$

$$= \frac{\left[\frac{0.25x^2}{2}\right]_0^{2.5} + \left[\frac{0.5x^3}{3} - \frac{x^2}{2}\right]_{2.5}^{3} + \left[\frac{0.5x^2}{2}\right]_3^5 + \left[\frac{-0.5x^3}{3} + \frac{3x^2}{2}\right]_5^6}{[0.25x]_0^{2.5} + \left[\frac{0.5x^2}{2} - x\right]_{2.5}^{3} + [0.5x]_3^5 + \left[\frac{-0.5x^2}{2} + 3x\right]_5^6}$$

$= 3.2$

The effect of these rules was observed with help of MATLAB -Fuzzy Tool Box. The value of maintainability for above-mentioned inputs comes out to be 3.2 in MATLAB simulation, which is same as our value calculated above (Ref. 15). This integrated approach gives a true picture of the software maintainability. Crisp value of maintainability can be helpful to software managers in judging the maintenance efforts of various maintenance activities. Lower values of maintainability indicate need of improvement in the software, so that the maintenance costs can be reduced. Further the measured value can be used to estimate changeability and testability also.

REFERENCES

1. Ian Sommerville, "Software Engineering", 5th Ed., Addison Wesley, 1996.
2. J. R. Mckee, "Maintenance as a Function of Design", Proceedings AFIPS, National Computer Conference, Las Vegas, pp 187-93.
3. "Software Engineering Standards", 1994 Edition, IEEE, 1994.
4. James F. Peters, W. Pedrycz, "Software Engineering: An Engineering Approach", John Wiley & Sons Inc, 2000.
5. Kari Laitnen, "Estimating Understandability of Software Documents", ACM SIGSOFT, vol 21, July 1996, pp 81-92.
6. CC Lee, "Fuzzy Logic in Control System: Fuzzy Logic Controllers - Part I", IEEE on Systems, Man and Cybernetics, Vol. 20, No.2, Mar/April 1990, pp 404-417.
7. CC Lee, "Fuzzy Logic in Control System: Fuzzy Logic Controllers - Part II", IEEE on Systems, Man and Cybernetics, Vol. 20, No.2, Mar/April 1990, pp 419-435.
8. George C. Mouzouris, Jerry M. Mendel, "Non Singleton Fuzzy Logic Systems: Theory and Application", IEEE Transactions on Fuzzy Sets and Systems, Vol. 5, No. 1, Feb. 1997.
9. Shuwei Guo, Liliane Peter, "Design and Application of an Analog Fuzzy Logic Controller", IEEE Transaction on Fuzzy Systems, Vol. 4, No 4, Nov. 1996, pp 429-437.
10. Marek J. Patyra, Janos L Grantner , Kirby Koster, "Digital Fuzzy Logic Controller: Design and Implementation", IEEE Transactions on Fuzzy Systems, Vol. 4, No 4, Nov. 1996, pp 439-459.
11. J. Patyra, D. M. Mlynek (Editors), "Fuzzy Logic: Implementation and Applications", John Viley & Sons Ltd. and B. G. Teubner, 1996.
12. George J. Klir, Bo Yuan, "Fuzzy Sets and Fuzzy Logic : Theory and Applications", Prentice Hall of India, New Delhi, 1995.
13. Ronald R. Yager, D. P. Filev, "Defuzzifiaction with Constraints", Fuzzy Logic and Applications to Engineering, Information Sciences and Intelligent Systems, Edited by Z. Bien, K K Min, Kluwer Academic Publication, 1995, pp 157-166.
14. Hellendorn, C. Thomas, "On Quality Defuzzification: Theory and Application Example", Fuzzy Logic & Applications to Engineering, Information Sciences and Intelligent Systems, Edited by Z. Bien and K K Min, kluwer Academic Publication, 1995, pp 167-176.
15. Roger Jang, Ned Gulley, "Fuzzy Logic Toolbox for MATLAB, User's Guide", The Math Works Inc., USA, Jan. 1995.

BIOGRAPHIES

1. **Krishan K. Aggarwal**, Vice-Chancellor, GGS Indraprastha University, Kashmere Gate, Delhi (INDIA).
   Email: aggarwal_krishan@hotmail.com

Prof. K.K. Aggarwal, Vice-Chancellor of the GGS Indraprastha University, graduated in Electronics, & Communication Engineering from Punjab University and obtained his Masters Degree in Advanced electronics from Kurukshetra University, securing first position in both. He did his Ph.D. in Reliability, evaluation & Optimization from Kurukshetra University. He was chairman, Department of Electronics, Communication & Computer Engineering and Dean (Academics) at Regional engineering College, Kurukshetra University. He was also Director, "Center for Excellence for Manpower Development in Reliability Engineering" established by Ministry of Human Resource Development there. Before joining Indraprastha University, he was Pro Vice-Chancellor, Guru Jambheshwar University, Hisar.

Prof. K.K. Aggarwal has extensively worked in Electronics & Communication Engineering, Quality & Reliability, and has published about 200 papers, a majority of them in International journals. He has also authored books and articles; many of them have appeared in IEEE, USA publications. He has delivered lectures for several universities in India and abroad. He has been widely consulted by the industry, most notable being his contribution towards the Reliability Analysis for PSLV (Polar Satellite Launch Vehicle). Prof. K.K. Aggarwal was honored by the Reliability Society of IEEE, USA for his services as Guest editor for the special issue on "State of Reliability Effort of the Indian Sub-Continent". He was declared as Man of Decade, Man of the Century and finally Man of the Millennium by American Bibliographical Institute, USA and was conferred L.C. Verman award by the Institute of Electronics & Tele-Communication Engineers. He was also awarded the Life Time Achievement Medal by the India International Friendship Society.

2. **Yogesh Singh,** Professor & Dean, School of Information Technology, GGS Indraprastha University, Kashmere Gate, Delhi (INDIA). Email : ys66@rediffmail.com

Dr. Yogesh Singh received his M.Tech & Ph.D. (Computer Engineering) from Regional Engineering College, Kurukshetra. Prior to present assignment, he was Reader & founder Chairman, Department of Computer Science & Engineering, Guru Jambheshwar University, Hisar (1996-99);

Lecturer, Computer Science & Engineering, CR State College of Engineering, Murthal, Sonepat (1993-96) and Scientific Officer in Electronics, Communication & Computer Engineering Department, Regional Engineering College, Kurukshetra (1989-93). He was the Principal Investigator of a successfully completed MHRD-AICTE project on "Experimentation & Development of Software Reliability & Complexity Measurement Techniques". His teaching & research interests include: Software Project Planning, Software Testing, Metrics, Data Structure, Computer Architecture, Parallel Processing. He has written numerous articles, papers and reports mainly pertaining to Software Engineering, Computer Architecture & Information Technology. He is a member of various professional bodies like ISTE, IETE & CSI.

3. **Jitender Kumar Chhabra**, Senior Lecturer, Department of Electronics, Communication & Computer Engineering, Regional Engineering College, Kurukshetra-136119 (INDIA).
Email: jitenderchhabra@rediffmail.com
Jitender Kumar Chhabra has received his M.Tech in Computer Engineering & B.Tech in Computer Engineering, both from Regional Engineering College, Kurukshetra and is teaching there since last 8 years. He is pursuing his PhD work in area of software engineering in GGS Indraprastha University. His areas of interest include software engineering, data base system, data structure, programming techniques.