Evaluation of Training Methods for Conditioning of Fuzzy Based Maintainability Metric

JITENDER KUMAR CHHABRA[a], SURENDER SINGH DAHIYA[b], SHAKTI KUMAR[c]
[a, b]National Institute of Technology, Kurukshetra-136119 (INDIA)
[c]Institute of Science & Technology, Klawad, Yamunanagar(INDIA)

Abstract: - The software maintainability can be ensured by carefully control of its software development process. An early measurement of maintainability starting from design phase is always desirable to produce maintainable software. Some of the researchers have tried to use soft computing techniques to measure maintainability. In spite of their reported validations, these models are not calibrated and no attention has been paid to evaluate and improve the stability of these methods.
An attempt has been made in this paper to evaluate and compare several methodologies for improving the numerical stability of a fuzzy logic based maintainability metrics system. Tuning of fuzzy system parameters is carried out using genetic algorithm with system condition number as objective function for optimization. A number of alternates are considered, in which training data sets are generated using different methods and these sets are used to evaluate objective functions in GA and accordingly fuzzy parameters are tuned. In order to show the advantage of such stability improvement, real projects' maintainability data is used and our study indicates that fuzzy model performance gets increased after conditioning.

Key-Words: - Leave software maintainability, soft computing, software metrics.

1 Introduction

For proper control of a software development process, we need to measure software attributes at every step of software development. The measured crisp value of an attribute provides right direction to software managers to take efficient and effective decisions. Some of the measures are dependent on many attributes and hence need integration, which may not be feasible using mathematical formulas, if attributes are of diverse nature. For such types of integration, fuzzy modeling is ideal [1]. In the last decade, several fuzzy based integrated measures have come up in the literature [2, 3]. Maintainability is one such hybrid attribute, whose crisp value depends on many lower order attributes such as source code size, comment ratio, software complexity, source code readability, documentation quality etc. Several researchers have stressed on the early measurement of maintainability starting from design phase itself so that timely steps could be taken for producing maintainable software [3-5].
Several models have been proposed for finding maintainability of software in different perspective. [2, 7-18]. Earlier researchers used to consider only source code and its allied comments for program comprehension [7-9], but later with increased complexity of software, understanding other software artifacts such as design documents were also assumed compulsory [2, 3, 19]. As there are several diverse attributes collected from different artifacts of software, which are distinct in measurement and performance scale, efforts are made to integrate these, in order to get a single crisp value of maintainability. For this type of integration fuzzy modeling was used by several authors [2, 3, 10].

2 Model under Consideration

For the purpose of evaluation and experimentation, we have considered a four-input-parameters maintainability metrics computed with help of a fuzzy model proposed by Aggarwal et. al. [3]. In this authors have considered average cyclomatic complexity (ACC), readability of source code (RSC), Documents quality (DOQ) and understandability of software (UOS) as important attributes for the measurement of maintainability. Average Cyclomatic Complexity (ACC) is defined as average of Cyclomatic complexities of all modules. Above model considers ACC as one of the contributing factor toward software maintainability. A source code must be supported by enough comments to make it readable for the purpose of software comprehension. In the above model RSC is measured using comment ratio in source code. DOQ quantifies the quality of documents with the help of fog's index. Fog index counts the length of sentences and difficulty of words. Higher the fog's index less is the quality of documentation for the purpose of understandability. To measure the UOS, degree of similarity of usage of symbols between the language of documentation and language of source code is considered using Laitnen's tool.

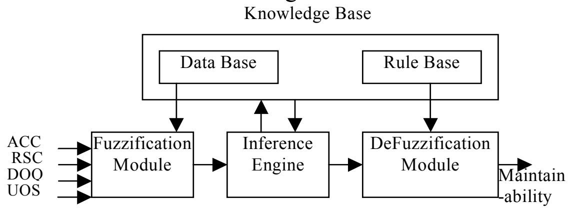


Figure 1: Maintainability Fuzzy System

The fuzzy system, as shown in Figure 1, proposed in [3], comprises of four basic elements: fuzzifier, fuzzy knowledge base, fuzzy inference engine, and defuzzifier. Fuzzification module is used to change the crisp values of inputs into fuzzy values based on the membership functions (MFs) defined in knowledge base. Fuzzy knowledge base is comprised of two parts: database in which MFs are stored and rule base which stores the decision mechanism of model. It computes the maintainability in fuzzy domain consists of three sub modules; namely rule composition, implication and aggregation module. Rule composition transforms the fuzzified antecedent part of the rule to single numerical figure that is used to implicate the output of a rule. Aggregation module then aggregates the individual outputs of rules to a single fuzzy set. Defuzzifier reverts the fuzzy output of inference module back to crisp values, which in turn is the output of the fuzzy system. Fuzzy system's performance largely depends on the accuracy of definition of membership function and correctness of rule base.

In order to fuzzify the inputs, three MFs for the ACC, RSC, DOQ and UOS were chosen and five MFs were chosen for output Maintainability. In order to measure software maintainability using the four input metrics, the rulebase for the system consisted of the eighty-one rules as follows:

1. If (ACC is LOW) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is V_GOOD).
2. If (ACC is AV) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is V_GOOD).
3. If (ACC is HIGH) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is GOOD).

…

.81. If (ACC is HIGH) and (RSC is POOR) and (DOQ is LOW) and (UOS is LESS) then (MAINTAINABILITY is V_POOR).

3. Stability Analysis of the Fuzzy System

A fuzzy model is essentially a transfer function, which maps input space to output space. A good model, be it mathematical model or heuristic model, must be stable consistent, cohesive and robust for its better performance. A model is called numerically stable, if given small perturbation in inputs, output doesn't vary marginally. This is also called structural stability [20]. Aggarwal et. al. in [21] carried out sensitivity analysis of a fuzzy model and a neural network model of a specific chosen problem. In due course, authors compared the stability of both models with the help of empirical evidence. Stability of system is derived by calculating condition number for each input parameter. Condition number of each input must be low for the whole system. Condition number is defined as the maximum value of the ratio of the relative margin in the output to the relative change in data over the problem domain and can be expressed in the form of an equation as shown below:

$$CN_x = \left| \frac{f(x+\Delta x, y, z) - f(x, y, z)}{f(x, y, z)} \times \frac{x}{\Delta x} \right| \ldots\ldots(1)$$

where CN_x is condition number corresponding to x input, $f(x,y,z)$ is the fuzzy model function to measure maintainability and x, y, z are the input parameters and Δx is the perturbation in the input parameter x. Perturbation must be very small and in our experiments we have taken this as 0.1 percent of input parameter. Functions with a condition number closer to one are "*more stable*" or "*well conditioned*" as compared to functions with a condition number greater than one.

Authors in [21] concluded that neural network based models are more stable than fuzzy models. But unavailability of sufficient training data and large training time of neural network are deterrent in adopting neural network scheme for modeling purpose. Further if the system for which, model is prepared, is a new system, then we don't have any alternative but to develop fuzzy model for such systems. Therefore, fuzzy system models must satisfy the stability criterion. When the system is continuous and parameters are monotonic, dips and steep-changes in the solution-space (surface view) indicates an ill-conditioned and unstable system. These dips and steep changes are measured using the concept of condition number as described above. So, if we need to make system stable, which has to be, the condition number needs to be minimized. In previous paper [22], an attempt has been made to improve the stability of a fuzzy model based measurement of three-inputs software maintainability metric [2] by transforming the unconditioned system to the conditioned one using genetic algorithm. The criterion of conditioning/objective function was chosen as the overall stability of system, which is aggregated by taking mean squared value of condition number of each input. The minimization of the defined objective function results into defining revised boundaries of the membership functions of each of the inputs under consideration. In our previous methodology, we have only considered the tuning of MFs by training fuzzy model by equispaced points training data set. A training data set is made by taking equispaced points from each input and then merging these in all permutations, which in turn is used in objective function. The results in previous paper were quite encouraging and prompted us to experiment with new alternatives of different training data sets and try to find out the best method for tuning MFs of fuzzy system with minimum condition number.

4. Fuzzy System Conditioning using Genetic Algorithms

As illustrated in Figure 2, initially, GA generates a random population of individuals called chromosomes and then based on an objective function it ranks and selects individuals to build a mating pool in order to generate next generation offsprings using genetic operators such as crossover or mutation, which have the higher possibility of being fitter than the present individuals. In fuzzy system conditioning, boundaries of membership functions of already unconditioned system are varied and each individual in population defines new revised boundaries, which is also a solution to the problem, however with a varying fitness. Now each solution is checked for its fitness based on condition criterion described in section III.

Fuzzy system knowledge base has two main components: MFs of inputs & outputs and the rule base. These components are created using expert knowledge. Sometimes, when the system is not a legacy system, there are enough chances that definitions of membership functions and rule base are not optimal [23]. This is the main source of instability in the system and same is true for the model under consideration. Hence our problem is reduced to optimize the MFs and the rule base such that the condition numbers of all inputs of the system decreases. In the present study, we have considered optimization of the MFs only using different alternative of training data sets with a view to adjudge the best method of

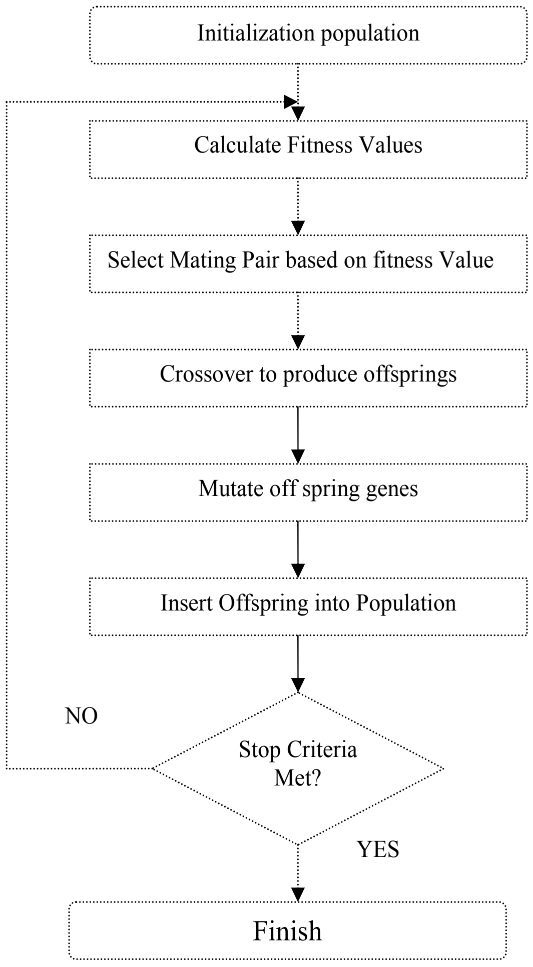

Initialization population
Calculate Fitness Values
Select Mating Pair based on fitness Value
Crossover to produce offsprings
Mutate off spring genes
Insert Offspring into Population
Stop Criteria Met?
NO
YES
Finish

Figure 2: Major Steps in GA Algorithm

training in order to get a stable fuzzy model. We have carried out experiments in Matlab framework, in which MFs of fuzzy model described above are tuned using GA such that overall condition of the system is reduced. As there is no guarantee of settling time of a condition number to a minimum specified value, therefore GA is iterated for a fix number of times. In our experimentation setup there are 4 runs of 100 generation each for one GA optimization.

5. Genetic Encoding of Fuzzy System

In the four-inputs and single-output system discussed above, each MF is trapezoidal (triangular MF is a special case of trapezoidal MF), thus each MF has four parameters, which must be adjusted for optimization. So a chromosome (probable solution) will contain individual genes in the form of the parameters of all the MFs of inputs and outputs. In the model under consideration, there are total 17 membership functions. So there will be 17*4=68 genes in a chromosome. In order to tune a MF, we allowed 10% variation in the existing range of each parameter. Thus if a parameter value of a MF of an input is 5 and input range is 1 to 13, then this parameter value is tuned within the range of 5-0.1*(13-1) and 5+0.1(13-1) i.e within the range of 3.8 to 6.2. A pictorial encoding scheme is shown below in figure 3.

6. Objective function definition

This is a multi-objective problem, as we need to decrease the condition number of each input. The condition number of each input can be calculated using equation 1 and is shown below in form of equation 2-5:

$$CN_{ACC} = \left| \frac{f(ACC + \Delta ACC, RSC, DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{ACC}{\Delta ACC} \right| \text{..(2)}$$

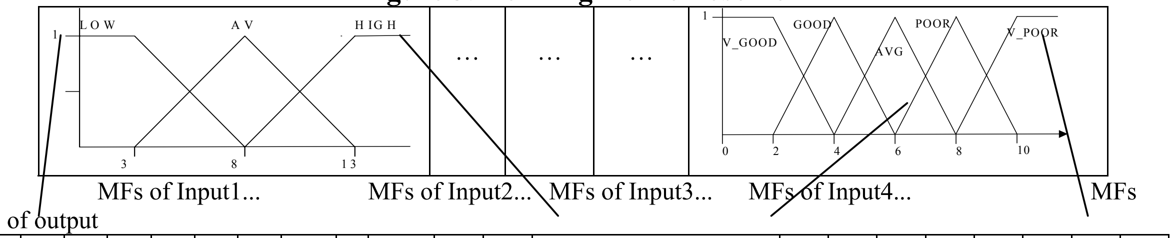
$$CN_{RSC} = \left| \frac{f(ACC, RSC + \Delta RSC, DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{RSC}{\Delta RSC} \right| \text{..(3)}$$

$$CN_{DOQ} = \left| \frac{f(ACC, RSC, DOQ + \Delta DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{DOQ}{\Delta DOQ} \right| \text{..(4)}$$

$$CN_{UOS} = \left| \frac{f(ACC, RSC, DOQ, UOS + \Delta UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{UOS}{\Delta UOS} \right| \text{..(5)}$$

An integrated objective function is calculated by taking the mean square value of all condition numbers, corresponding to each of the inputs. This number is called condition number of the system CN_{SYS} and is used as objective function for GA. $CN_{SYS} = MSE(CN_{ACC}, CN_{RSC}, CN_{DOQ}, CN_{UOS})$……(6)

Figure 3. Defining a chromosome



1	2	3	4	5	6	7	8	9	10	11	12	::::::::::::::::::::::	61	62	63	64	65	66	67	68
0	1	3	8	3	8	8	13	8	13	13	15	::::::::::::::::::::::	6	8	8	10	8	10	10	12

7. Generation of Training Data for MFs Tuning

In our experimentations, we have developed two modules. First is the main genetic module and is based on the algorithm described above. Second module defines an objective function which is used to calculate system condition for a given data set of inputs (here called training data set and is a matrix of 2401×4 size) based on equation 2, 3, 4, 5 and 6. Main GA module generates various acceptable solutions. For each chromosome in each population, GA module changes only membership functions' parameters from the unconditioned fuzzy model keeping rule base unchanged and then in objective function, fuzzy inference system evaluates output of system with respect to inputs provided in form of training data. Subsequently, an input is perturbed throughout in the training data set, and again same fuzzy system is used to evaluate outputs with respect to this perturbed data set. These two output sets are used then to calculate condition number of the system w.r.t. the particular input by following equation 2 to 5. Same procedure is repeated for other inputs and then following equation 6, we get overall system conditioning for particular chromosome. We have formulated six methods for constructing training data for optimization as described below:

a) Conditioning with different random data set for each solution in population

In this method a new random numbers data set of 2401 points are generated for each chromosome in population of one generation of GA algorithm for evaluating condition number of inputs and whole system.

b) Conditioning with same random data set for each solution in population

This method generates random numbers data set of 2401 points once in Main GA module and same is used for finding fitness for each chromosome in each population of GA algorithm.

c) Conditioning with one input equispaced and same random numbers data set for each of rest inputs for each solution in population

Here main function generates two data sets of 2401 points each from all inputs' ranges. One data set contains equispaced points from all inputs and another set contains random numbers from all inputs. Then both of these sets are used to find the condition numbers of all inputs.

d) Conditioning with one input equispaced and different random numbers data set for each of rest inputs for each solution in population

Here main GA function generates one data set which contains equispaced numbers from all inputs and another set which contains random numbers from all inputs is created by the objective function each time for each chromosome. Then these both sets are used to create four training data sets for all inputs by following the same procedure as described in method three.

e) Conditioning with all permutation generated by taking equispaced points from each input for each solution in population

In this method seven equispaced points are taken from each input space and then these are combined in all permutations to form a training data set of 2401(7^4) points, which is used to calculate condition numbers for each input. We have taken only seven equispaced points from each input. However it can be less or more.

f) Conditioning with each input equispaced points summation for all solutions in population

Here from each input space 2401 equispaced points are generated and then these are clubbed together to form a training data set of 2401 points.

In order to validate the methodologies, these need to be tested with a new data set. So once the training is completed, these methods are tested with a data set of 10000 points. This data set is randomly generated for all inputs and is same for all above defined alternatives. To fully randomize the output, we repeated this process ten times and then average system condition number of all ten outputs is taken for comparison. Table 1 lists condition number of unconditioned and conditioned system for all six alternatives.

TABLE 1: Comparison of condition numbers of unconditioned and unconditioned systems

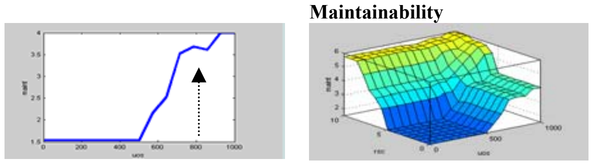
Methods	CN_{ACC}	CN_{RSC}	CN_{DOQ}	CN_{UOS}	CN_{SYS}
Unconditio-ned System	5.0622	11.945	13.314	4.4294	96.087
Conditioned System with 1st Method	2.9006	11.943	11.973	5.629	82.631
Conditioned System with 2nd Method	2.1427	4.7642	5.5817	2.2429	16.064
Conditioned System with 3rd Method	2.7784	5.8229	5.9942	3.0474	22.023
Conditioned System with 4th Method	3.6703	4.5191	6.5657	3.657	23.258
Conditioned System with 5st Method	1.895	7.6208	4.4021	8.7757	40.451
Conditioned System with 6st Method	2.384	13.577	5.8428	4.6509	62.599

Figures 4(a) to 10(a) show the graph of input UOS versus maintainability while taking constant values of other three inputs. Figure 4(a) shows Maintainability curve with respect to UOS input parameter in unconditioned System. There are steep changes and dip (marked by arrow) in the maintainability curve. Figures 4(a) to 10(a) show surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes. Figures 4(b) shows surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes in unconditioned system. This is also comprised of steep changes in maintainability with a gradual increase in input-parameters.

8. Results & Discussions

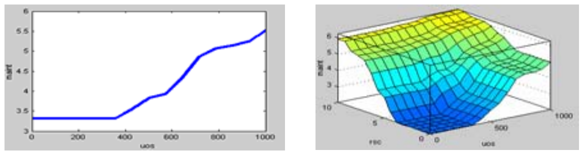
A per Table 1, Condition number for each input parameter is reduced in all six methods. From table, it can be deduced that Second Method is best, in which overall system condition has been reduced from 96.087 to 16.064. This is a six fold decrease in condition number of the system. It is a significant improvement in the stability and system can be further stabilized with increased number of GA iterations. Others better methods are third & fourth methods, for which condition numbers are 22.023 and 23.258 respectively.

UOS Vs Maintainability **(b) UOS & RSC Vs Maintainability**



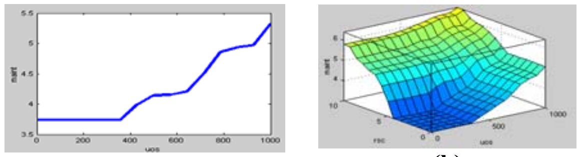
(a) (b)

Figure 4: unconditioned System



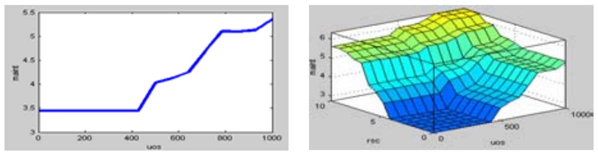
(a) (b)

Figure 5: System is conditioned with first method



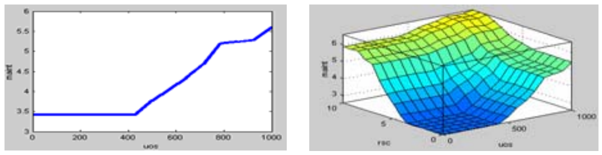
(a) (b)

Figure 6: System is conditioned with second method



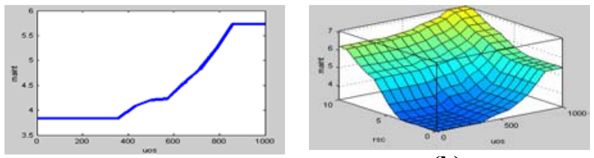
(a) (b)

Figure 7: System is conditioned with third method



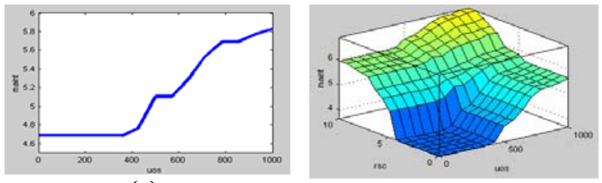
(a) (b)

Figure 8: System is conditioned with fourth method



(a) (b)

Figure 9: System is conditioned with fifth method



(a) (b)

Figure 10: System is conditioned with sixth method

In these three methods, which yield good conditioning, condition numbers of all the inputs as well as whole system have been less as compared to unconditioned system while on the other side first, fifth and sixth methods one or two inputs condition number is larger than unconditioned system. Although in these methods, whole system condition number is reduced but these also destabilize system at other inputs. Figure 9(a) is of unconditioned system, in this as pointed by an arrow, there is a dip at axis (3.65, 870) which violates model consistency and continuity. In all other figures this dip has been removed completely. If we analyze the surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes of figure 9(b), there are steep changes in maintainability given small variations in inputs for unconditioned system.
These steep changes in maintainability have been replaced by smoother curves in figures 10(b) to 15(b) for conditioned system for the same surface view. If we compare the surface view of maintainability versus RSC & UOS of unconditioned system (figure 9(b)) with conditioned system of second method (Figure 11(b)) which has the least condition number, surface view is smoothest of all other methods. So we can deduce that system can be made stable or well conditioned by following method number 2 or 3 or 4 in that order.
Authors in [3] have validated their model by collecting empirical data of maintenance time of eight software projects and same has been used to evaluate the performance of the model. In this paper, we have used the same data to check the effect of conditioning on fuzzy system performance. Table 2 shows the computed maintainability values from unconditioned fuzzy model and other fuzzy model conditioned with all the six alternatives, against input data from eight projects. In the last row of table 2 computed maintainability values are correlated with average maintenance time of eight projects. Here, we find that conditioning does not deteriorate the correlation in all the methodologies, rather it has been increased in each method. Although these methodologies are not targeted to increase correlation between maintainability and average maintenance time but this favorable change further proves the merit of our proposed methodologies for conditioning the system.

9. Conclusion

This paper has presented six different alternatives to generate training data in order to find out the best possible method of conditioning. Each of the methods was evaluated using a new data set of 10000 points. Our initial study indicates that if training data set is created randomly from each input space and system is conditioned using this training data then this method outperforms other methods by making system more stable on average basis. Methods with one input equispaced and same random numbers data set for each of rest inputs for each solution in population or with one input equispaced and different random numbers data set for each of rest inputs for each solution in population also give better results as compared to the rest three methods. These alternatives are also validated against eight real projects average maintenance time by computing maintainability from the conditioned systems and increased correlation proves the worthiness of these methodologies.

Table 2: Correlation between observed maintenance time and computed maintainability values

Project Number	ACC	RSC	DOQ	UOS	Corrective maint-time	Maintainability of system when conditioned with						
						Unconditioned system	First Method	Second Method	Third Method	Fourth Method	Fifth Method	Sixth Method
1	8.5	3.8	11	355	11.300	3.610	3.615	3.888	3.640	3.212	3.983	4.460
2	12	7.7	15	528	21.700	7.370	6.726	6.332	6.674	6.958	7.124	6.700
3	13	5.7	11	492	18.300	5.110	5.538	5.400	5.277	5.542	5.543	5.707
4	5.4	8.3	12	567	18.000	6.810	6.027	5.311	5.546	6.116	5.966	5.692
5	15	8.9	12	363	21.100	8.000	7.492	6.750	7.331	7.275	7.409	7.051
6	7.5	7.4	8.9	390	16.100	4.560	4.826	4.948	5.127	5.290	5.152	5.763
7	11	9.2	12	451	17.900	7.070	6.758	6.473	6.438	6.375	6.942	6.543
8	9.1	6.9	13	479	17.200	6.000	6.240	6.046	6.024	5.986	6.456	6.087
Correlation between computed maintainability values and observed maintenance time →						0.873	0.902	0.875	0.913	0.961	0.899	0.912

References

1. W. Pedrycz, J. F. Peters, "*Computational Intelligence and Software Engineering*", World Scientific*,* Singapore, 1998.

2. K. K. Aggarwal; Y. Singh; J. K. Chhabra, "An Integrated Measure of Software Maintainability", *IEEE Proceedings of Annual Reliability and Maintainability Symposium, RAMS-2002,* Seatle Westin, USA, Jan 28-31, 2002, pp. 235-241.

3. K. K. Aggarwal; Y. Singh; J. K. Chhabra, "A Fuzzy Model for Measurement of Software Maintainability & Its Performance", *Int. Journal of Electronics & Computer Science*, USA Vol 6, No2, 2004 pp. 31-43.

4. W.M. Osborne, E.J. Chikofsky, "Fitting Pieces to the Maintenance Puzzle", *IEEE Software*, Vol 7, No 1, pp. 11-12, 1990.

5. N. F Schneidewind, "The State of Software Maintenance", *IEEE Transactions on Software Engineering*, Vol SE-13, No 3, pp. 303-310, 1987.

6. T. M. Pigoski, "*Practical Software Maintenance – Best Practices for Managing Your Software Investment*", John Wiley & Sons, New York, NY, 1997.

7. P. Oman; "HP-MAS: A Tool for Software Maintainability Assessment", U.I. Software Engineering Test Lab Report #92-07-ST, August 1992.

8. F. Zhuo, B. Lowther, P. Oman, J. Hagemeister, "Constructing and Testing Software Maintainability Assehssment Models," *First International Soflware Metrics Symposium*, IEEE Computer Society Press, Baltimore, Maryland, May 21-22, 1992, pp. 423-433.

9. J. Munson, T. Khoshgoftaar, "The Detection of Fault-Prone Programs," *IEEE Transactions on Software Engineering*, Vol. 18, No 5, May 1992. pp. 423-433.

10. J. K. Chhabra, K. K. Aggarwal, Y. Singh, *"*A Fuzzy Model for Measurement of Software Understandability*", International Symposium on Performance Evaluation of Computer & Telecommunication Systems SPECTS'03,* Montreal, Canada, July 20-24, 2003.

11. J. K. Chhabra, K.K. Aggarwal, Y. Singh, "Code and Data Spatial Complexity: Two Important Software Understandability Measures", *Information and Software Technology*, Vol. 45, 2003, pp. 539–546.

12. J. K. Chhabra, K.K. Aggarwal, Y. Singh, "Measurement of Object-Oriented Software Spatial Complexity", *Information and Software Technology*, Vol. 46, No 10, August 2004, pp. 689-699.

13. B. S. Mitchell, S. Mancoridis, "On the Automatic Modularization of Software Systems using the Bunch Tool", *IEEE Transactions on Software Engineering*, Vol. 32, No. 3, March 2006, pp.193 – 208.

14. F. Xia, "A Change Impact Dependency Measure for Predicting the maintainability of Source Code", *Proceedings of the 28th Annual International Computer Software and Applications Conference*, COMPSAC-2004. Vol. 2, 2004, pp. 22 – 23.

15. J. K. Chhabra, K.K. Aggarwal, Y. Singh, "Measuring Maintainability of Object-Oriented Software", *International Journal of Management And Systems IJOMAS*, Vol 20, No 2, 2004, pp. 139-156.

16. M. Humphrey, S. Henry, J. Lewis, "Evaluation of the Maintainability of Object-Oriented Software", *IEEE Region 10 Conference on Computer and Communication Systems*, September 1990.

17. M. Kiewkanya, N. Jindasawat, P. Muenchaisri, "A Methodology for Constructing Maintainability Model of Object-oriented Design" *Proceedings of Fourth International Conference on Quality Software*, QSIC, 2004 pp. 206 – 213.

18. V.S. Alagar, L. Qiaoyun, O.S. Ormandjieva, "Assessment of Maintainability in object-oriented Software" *39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems*, TOOLS 39, 29 July-3 Aug. 2001, pp. 194 – 205.

19. L. Briand, C. Bunse, J. Daly, "An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents", *Proceedings of International Conference on Software Maintenance*, 1-3 Oct. 1997, pp. 130 - 138.

20. http://en.wikipedia.org/wiki/Structural_stability

21. K.K. Aggarwal; Y. Singh; P. Chandra; M. Puri, "Sensitivity Analysis of Fuzzy and Neural Network Models", ACM SIGSOFT Software Engineering Notes, Vol. 30, No. 4, July 2005, pp. 1-4.

22. S. S. Dahiya, J. K. Chhabra, S. Kumar, "Use of Genetic Algorithm for Software maintainability Metrics' Conditioning" *15th International Conference on Advanced Computing and Communications, ADCOM- 2007* (accepted for publication)

23. I. Rojas, J. Gonzalez, H. Pomares, F. J. Rojas, F. J. Fernandez, A. Prieto, "Multidimensional and Multideme Genetic Algorithms for the Construction of Fuzzy System", *International Journal of Approximate Reasoning*, Vol. 26, 2001, pp. 179-210.