Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 933 – 940

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Construction of Membership Function for Software Metrics

Harikesh Bahadur Yadav[a,]*, Dilip Kumar Yadav[b]

[a,b]*Department of Computer Applications,National Institute of Technology, Jamshedpur,831014,INDIA*

Abstract

A software developer has to deal with a lot of challenging requirements such as cost prediction, defect prediction, reliability prediction, testing effort prediction, safety prediction, and many more while developing quality software. However, it has been found that the most of the software development activity is performed by human beings. This may introduce various faults across the development, causing failures in near future. Therefore, prediction of software defect has been one of the major areas of concern. A number of the software defect prediction model using software metrics has been proposed in last two decades. However, predicting software defect by taking all the software metrics (traditional, object oriented and process) is computationally complex. Therefore, an intelligent selection of metrics plays a vital role in improving the software quality. In the early phases of the software development life cycle, software metrics are associated with uncertainty and can be assessed in linguistic terms. Construction of membership function is very important because the success of a method depends on the membership functions used. Therefore, in this paper, a methodology has been proposed to construct the membership functions of software metrics.

Keywords: software metrics; fuzzy logic; membership function; k-means clustering; software defect

1. Introduction

Today, Computer is used in diverse areas such as air traffic control, nuclear reactors, real-time sensor networks, industrial process control, hospital health care, home appliances, shopping, auditing, web teaching, personal

* Corresponding author. Tel.: +91-8986760730; fax: +0-657-2373246.
E-mail address: yadavaharikesh@gmail.com

doi:10.1016/j.procs.2015.01.002

entertainment, and so on. It is very essential to ensure that the underlying software will perform its intended functions correctly. Various historical events illustrate the effect of software failures encountered in and around the world[1,2]. The impact of these software failures can have a broad range of consequences starting from minor inconvenience to the loss of human life. Therefore, software defect prediction is unavoidable and it has become the most popular area of research nowadays. Many research industries started new projects in this field.

A number of the software defect prediction model using software metrics has been proposed in last two decades [3,8]. The prediction of defect from these models may be useful for quality software development. Almost all existing defect prediction models have considered a considerable number of software metrics such as traditional software metrics, object oriented software metrics, process metrics[9,10]. Catal[9] provided a systematic review of various software fault prediction models which state that most of the models uses method level metrics. In another review, it is explained that how the context of models, independent variables used and the modelling technique affect the various prediction accuracy[11]. Recently, Danijel Radjenovic *et al.*[10] reported that object oriented metrics and process metrics are more successful in finding the faults compared to traditional size and complexity metrics.

Drawbacks of software defect prediction by taking all the software metrics are as follows:

- Computationaly complex
- More expensive processing cost
- There are many less important software metrics
- Correlations among the software metrics
- Increase time complexity

It has been found that early phases software metrics are associated with uncertainty and can be assessed in linguistic terms. However, most of the research articles in journals dealing with fuzzy logic appear without using membership function. Developing membership functions of numerical data are one fundamental step in the design of a problem which is to be solved by fuzzy set theory. Therefore, membership functions play a very vital role in fuzzy inference system building. There are many problems that make membership function generation a non-trivial task.

- Lack of knowledge and interpretation of membership function.
- How to choose the appropriate fuzzy profile development technique?

The rest of this paper is organized as follows: In section 2, related work is discussed. In section 3, a method for construction of a fuzzy profile of software metrics is presented. In section 4, an example is explained to demonstrate the proposed method. In section 5, result of the proposed methodology is discussed. In section 6, finally the conclusion and future scope are presented.

2. Related work

In the literature, the importance of static code metrics can be observed[4,6-8,19]. Software developers can select a subset of software metrics amongst the existing large set of software metrics. Prior studies show that the right selection of metrics plays a vital role in improving the defect prediction[12-15]. Right selection of software metric could improve the prediction accuracy of the proposed model. For this study, static code metrics[16,17] and line count metrics have been considered as the simplified metrics set because the defect prediction using static code metrics shows higher performance[18].

The fuzzy set theory provides a way to capture the uncertainty, vagueness and imprecision present in the software metrics. Zadeh's had provided a new way of the thinking about uncertainty[19,20]. Membership functions play a very important role in fuzzy expert system building. Triangular and trapezoidal shapes provide a convenient representation of domain expert knowledge and it also simplifies the process of computation[22-24,34]. Therefore, it is needful to clarify that how the membership function is derived. Many research articles have been proposed for generating fuzzy if- then rule from numeric data[25-29]. Hong and Lee[28] proposed a method for membership function construction which needs predefine membership functions of the input variable. Ping and Chen proposed a new method for fuzzy profile generation based on α- cuts of equivalence relations. As the number of input variable becomes larger and the variable has a huge amount of values, the complexity of proposed algorithm will increase[29]. Dombi[33] had pointed out some common features among these different approaches in every research article:

- All membership functions are continuous.
- All memberships functions map an interval [a, b] to [0, 1].

- Membership functions are either monotonically increasing or monotonically decreasing or both increasing and decreasing.

On the basis of above literature surveys and review, it has been found that, most of the previously developed method uses predefined membership functions. Fuzzy profile development of software metrics is very important because the success of a method depends on the membership functions used. Therefore, in this paper, we propose a general learning method for construction of membership function of software metrics. The key contribution of this research paper is as follows:

- Selection of software metrics
- Fuzzy profile development of selected software metrics

3. Proposed methodology

In this section, steps taken in the selection of software metrics and their fuzzy profile development are explained.

3.1. Selection of Software metrics

This paper makes use of KC2 data set available from the Promise data repository[30]. This data set consists of 21 software metrics. However, from the studies[31], it is clear that out of 21 only 13 metrics play an important role in defect prediction. These 13 metrics are shown in Table 1.

Table 1. Selected software metrics

Metrics Category	Metrics Name	
Halstead Metrics	Total no. operator	(N_1)
	Total no. operands	(N_2)
	No. of unique operator	(n_1)
	No. of unique operands	(n_2)
McCabe	Cyclomatic complexity	
	Essential complexity	
	Design complexity	
Line of Code	LOC blank	
	LOC code and comments	
	LOC comments	
	Executable LOC	
	Total Line of Code	
Branch Count Metric	Branch count	

3.2. Construction of membership function

In this section, we proposed a technique for construction of a fuzzy profile of static code software metrics. Let S denote a set of N training patterns (F_1, F_2, …., F_j, …., F_n) and F_j denote a feature. The feature F_j have n values v_{1j}, v_{2j}, …., v_{nj}. $F_{j\ min}$ and $F_{j\ max}$ denote the minimum and maximum value of feature F_j. When the input feature is quantitative value, then followings steps are performed for membership function generation.

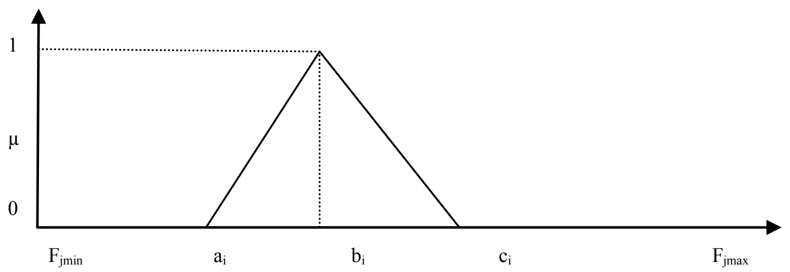

Fig. 1. A triangular membership functions

Step 1. Sort the values of feature F_j in ascending order.

Step 2. Perform K –means clustering algorithm for clustering the quantitative values of the feature F_j into k clusters ($y_1, y_2, \ldots, y_i, \ldots, y_k$) where, $y_{i\,min}$ and $y_{i\,max}$ denote the minimum and maximum value of i^th cluster (y_i).

Step 3. Find out the cluster centers ($b_1, b_2, \ldots, b_i, \ldots, b_k$) of k clusters ($y_1, y_2, \ldots, y_i, \ldots, y_k$).

Step 4. Determine the membership value of two boundary points every cluster.

Substep 4.1. Find the difference between adjacent data. For each pair v_i and v_{i+1} (i=1, 2, ..., n-1) the difference is $diff_i = v_{i+1} - v_i$.

Substep 4.2. Find the similarity value between adjacent data of quantitative values of feature F_j. The similarity between adjacent data is obtained according to the following formula[32].

$$s_m = \begin{cases} 1 - \dfrac{diff_i}{C * \sigma_s} & \text{for } diff_i \le C * \sigma_s \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where,

s_m	Represents the similarity between adjacent data
σ_s	Standard derivation of $deff_i$
C	Control Parameter deciding the shape of membership functions.

Substep 4.3. The minimum value of s_m of i^th cluster is chosen as the membership value of two boundary point's $y_{i\,min}$ and $y_{i\,max}$ of i^th cluster.

Step 5. Determine the left vertex point (a_i, 0) by interpolation.

$$a'_i = b_i - \frac{b_i - y_{i\,min}}{1 - \mu(y_{i\,min})} \tag{2}$$

$$a_i = \begin{cases} 0 & \text{for } a'_i \le 0 \\ b_{i-1} & \text{for } 0 < a'_i \le b_{i-1} \\ a'_i & \text{for } a'_i > b_{i-1} \end{cases} \tag{3}$$

Step 6. Determine the Right vertex point (c_i, 0) by interpolation.

$$c'_i = b_i + \frac{y_{i\,max} - b_i}{1 - \mu(y_{i\,max})} \tag{4}$$

$$c_i = \begin{cases} c'_i & \text{for } c'_i \le b_{i+1} \\ b_{i+1} & \text{for } c'_i > b_{i+1} \end{cases} \tag{5}$$

Step 7. Find the membership value of each quantitative values of feature F_j

$$\mu(v) = \begin{cases} \frac{b_i - v}{b_i - a_i} & \text{for } v < b_i \\ \frac{c_i - v}{c_i - b_i} & \text{for } b_i \le v < c_i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

4. An illustrative example

In this section we demonstrate the proposed method with an example. The numeric value of Cyclomatic complexity software metrics has been considered for explaining the proposed algorithm which is shown in Table 2.

Step 1. The sorted values of Cyclomatic complexity metrics are shown in Table 2.

Step 2. Let k=3, After applying k- means clustering algorithm we get three clusters y_1, y_2, and y_3. Cluster y_1 contain 95 values (1, 1,…., 2.4), Cluster y_2 contains 32 values (2.5, 2.5806,….., 4.5429), and cluster y_3 contain 18 values (5.1053, 5.25,……, 11.08) which is shown in Table 2.

Step 3. Cluster center of y_1, y_2, and y_3 cluster is b_1=1.443, b_2=3.419, and b_3= 6.090 respectively.

Step 4. In this step, we determine the membership value of two boundary points of every cluster by applying the Substep 4.1 to Substep 4.3. In order to get a similarity value between adjacent values of Cyclomatic complexity metrics, first of all the difference between adjacent data is calculated e.g. (v_2-v_1=1-1=0). The calculated values of diff_i's are shown in Table 2.

Let constant C=4. The standard deviation is calculated as 0.324. The value of similarity (s_m) is calculated using the formula 1 as follows:

$$s_1 = 1 - \frac{0}{4 * .324} = 1$$

$$s_2 = 1 - \frac{0}{4 * .324} = 1$$

……………………………………

$$s_{143} = 1 - \frac{0.2632}{4 * .324} = .7966$$

$$s_{144} = 1 - \frac{3.8168}{4 * .324} = 0$$

Table 2. Selected software metrics

Sr. No	Cyclomatic Complexity Value (v_i)	diff$_i$'s (v_{i+1}-v_i)	Standard Deviation (σ_s)	Constant (C)	Value of similarity (s_m)	Clusters(y_i)	Cluster Centre (b_1)	Cluster Centre (b_2)	Cluster Centre (b_3)
1	1	0	0.324	4	1	y_1	1.443	3.419	6.09
2	1	0			1	y_1			
……………	…………..	…………..			…………..	………..			
94	2.381	0.019			0.985319	y_1			
95	2.4	0.1			0.922733	y_1			
96	2.5	0.0806			0.937723	y_2			
97	2.5806	0.0861			0.933473	y_2			
…………	…………	………..			……….	………….			
126	4.5	0.0429			0.966852	y_2			
127	4.5429	0.5624			0.565449	y_2			
128	5.1053	0.1447			0.888194	y_3			
129	5.25	0			1	y_3			
………….	………….	…………			…………	………..			
143	7	0.2632			0.796632	y_3			
144	7.2632	3.8168			0	y_3			
145	11.08					y_3			

It is clear from table 2 that the minimum similarity value of cluster y_1, y_2, and y_3 are 0.922, 0.565, and 0 respectively. Therefore the membership value of the two boundary points $y_{i\ min}$ and $y_{i\ max}$ of y_i (i=1, 2, 3) is 0.922, 0.565, and 0 respectively.

Step 5. Determine the left vertex point (a_i, 0) by interpolation.
For cluster y_1, b_1=1.443, $y_{1\ min}$=1, $\mu(y_{1\ min})$=0.922, For cluster y_2, b_2=3.419, $y_{2\ min}$=2.5, $\mu(y_{2\ min})$=0.565, and for cluster y_3, b_3=6.09, $y_{3\ min}$=5.105, $\mu(y_{3\ min})$=0. The value of left vertex point is calculated using formula 2 and 3 as follows:

$$a'_1 = 1.443 - \frac{1.443 - 1}{1 - .92} = (-\text{ve})$$

a_1=0

$$a'_2 = 3.419 - \frac{3.419 - 2.5}{1 - .565} = (1.31 < 1.443)$$

a_2=1.443

$$a'_3 = 6.090 - \frac{6.090 - 5.11}{1 - 0} = 5.11$$

a_3=5.11

Step 6. Determine the Right vertex point (c_i, 0) by interpolation.
For cluster y_1, b_1=1.443, $y_{1\ max}$=2.4, $\mu(y_{1\ max})$=0.922, For cluster y_2, b_2=3.419, $y_{2\ max}$=4.54, $\mu(y_{2\ max})$=0.565, and for cluster y_3, b_3=6.09, $y_{3\ max}$=11.08, $\mu(y_{3\ max})$=0.The value of left vertex point is calculated using formula 4 and 5 as follows:

$$c'_1 = 1.443 + \frac{2.4 - 1.443}{1 - .92} = 13.40 > 3.419$$

c_1=3.419

$$c'_2 = 3.419 + \frac{4.543 - 3.419}{1 - .565} = 5.99$$

c_2=5.99

$$c'_3 = 6.090 + \frac{11.08 - 6.090}{1 - 0} = 11.08$$

c_3=11.08

Step 7. In this step, membership value of each quantitative value of feature F_j is calculated using formula 6.

5. Result of Proposed Methodology

After the operation of step 1 to 7, the fuzzy profile of Cyclomatic complexity software metrics of KC2 data sets is derived as shown in Fig. 2. It is clear that, the range of linguistic variable "Cyclomatic complexity" is 1 to 11.08. The metrics are grouped into three category y_1, y_2, and y_3. The right and left vertex of y_1, y_2, and y_3 is (0, 3.419), (1.44, 5.99), and (5.11, 11.08) respectively.

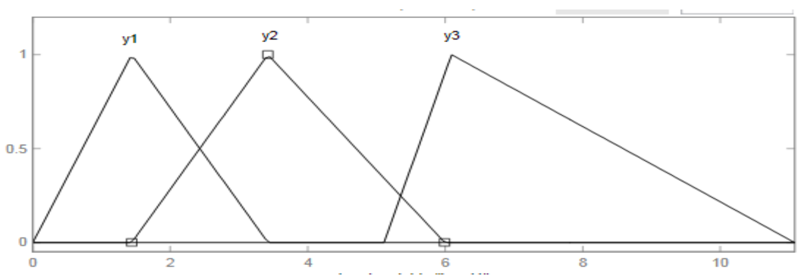

Fig. 2. Cyclomatic complexity membership functions

6. Conclusions

In this paper, a new approach is proposed to construct the fuzzy profile of software metrics for numerical data. The proposed method is better than Hall et al. algorithm[28], because no need to predefine fuzzy profile of input and output variables. The proposed algorithm helps researchers in fuzzy profile development of software metrics. Therefore, the proposed algorithm reduces the time and effort needed for it. This algorithm significantly helps researchers and software practitioners to develop a fuzzy rule based system for early software defect prediction. This provides a guideline to the software developer for early identification of cost overruns, schedules mismatch, software development process issues, software resource allocation and release decision making.

References

1. Lyu MR. *Handbook of software Reliability Engineering*. McGraw-Hill, IEEE Computer Society Press, Los Alamitos, CA, USA,1996. 1–851
2. Pham H. *System Software Reliability*. Reliability Engineering Series, Springer-Verlag Publisher, London, 2006.
3. Singh Y, Malhotra R. *Object Oriented Software Engineering*. PHI Learning, 2012.
4. Munson JC, and Khoshgoftaar TM. The detection of fault-prone programs. *IEEE Trans. on software engineering*, 1992. 18:423–433.
5. Basili VR, Briand LC, Melo WL. A validation of object oriented design metrics as quality indicators. *IEEE Trans. on software engineering,* 1996. 22:751–761.

6. Guo L, Cukic B, Singh H. Predicting fault prone modules by the dempster-shafer belif networks. *In 18th IEEE int. conf. on automated software engineering, Montreal, Canada*, 2003. 249–252.
7. Khoshgoftaar TM, Seliya N. Fault prediction modeling for software quality estimation: comparing commonly used technique. *Empirical software engineering*, 2003. 8:255–283.
8. Khoshgoftaar TM, and Allen EB. A Comparative Study of Ordering and Classification of Fault-Prone Software Modules. *Empirical Software Engineering*, 1999. 4:159–186
9. Catal C. Software fault Prediction: A literature review and current trends. *Expert System with Applications*, 2011. 38:4626–4636.
10. Hall GA, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. on software engineering,* 2012. 38:1–31.
11. Radjenovic D, Herico M, Torkar R *et al.* Software fault prediction metrics: A systematic literature review. *Information and Software Technology*, 2013. 55:1397–1418.
12. Wang HJ, Khoshgoftaar TM, Liang QA. A study of software metric selection techniques: stability analysis and defect prediction model performance. *International journal on artificial intelligence tools*, 2013. 22
13. Wang HJ, Khoshgoftaar TM, Wald R *et al*. A study on first order statistics- based feature selection technique on software metric data. *In: proceedings of the 25th Int'l conf. on software engineering & knowledge engineering (SEKE'13)*, Boston, USA, 2013. 467–472.
14. Liu H and Yu L. Toward integrating feature selection algorithm for classification and clustering. *IEEE Trans. on knowledge and data engineering*, 2005. 27:491–502.
15. Peng H, Long F, Ding C. Feature selection based on mutual information criteria of max- dependency, max-relevance, and min redundancy. *IEEE Trans. on pattern analysis and machine intelligence*, 2005. 27:1226–1238.
16. Halstead M. Element of software science. *Elsevier*, 1977.
17. McCabe TA. Complexity measure. *IEEE Trans. on software engineering,* 1976. 2:308–320.
18. Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering,* 2007. 33:2–13.
19. Malhotra R. Comparative analysis of statistical and machine learning methods for predicting faulty modules. *Applied Soft Computing,* 2014. 21:286–297.
20. Zadeh LA. Fuzzy Sets, *Information and control*, 1965. 8:338–353.
21. Zadeh LA. The concept of linguistic variable and its application to approximate reasoning-1. *Information Sciences*, 1975. 8:199–245.
22. Yadav DK, Charurvedi SK and Mishra RB. Early software defects prediction using fuzzy logic. *International Journal of Performability Engineering*, 2012. 8:399–408.
23. Yadav HB, Yadav DK. A multistage model for defect prediction of software development life cycle using fuzzy logic. *In Proceedings of the Third International Conference on Soft Computing for Problem Solving (SOCPROS-2013),* 2013, IIT Roorkee, India, *Springer India Publication, Advances in Intelligent Systems and Computing*, 2014. 259:661-671.
24. Yadav HB, Yadav DK. Defects prediction of early phases of software development life cycle using fuzzy logic. *In Proc. of the 4th International Conference (CONFLUENCE 2013) The Next Generation Information Technology Summit*, Amity University ,Uttar Pradesh, 2013*, IET Publications India.* 2-6.
25. Lee CC. Fuzzy logic in control systems: fuzzy logic controller. II. *IEEE Transactions on Systems, Man and Cybernetics,* 1990. 20:419–435.
26. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics,* 1985. 15:116–132.
27. Wang LX, Mendel JM. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics,* 1992, vol.22, pp.1414–1427.
28. Hong TP, Lee CY, Induction of fuzzy rules and membership functions from training examples, *Fuzzy Sets and Systems*, 1996. 84:33–47.
29. Wu TP, Chen SM. A new method for constructing membership functions and fuzzy rules from training examples. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* , 1999. 29:25–40.
30. Promise, http://promisedata.org/repository/
31. Seliya N, Khoshgoftaar TM, Shi Zhong. Analyzing software quality with limited fault-proneness defect data. *High-Assurance Systems Engineering, 2005. HASE 2005. Ninth IEEE International Symposium on* , 2005. 89–98.
32. Hattori K and Tor Y. Effective algorithm for the nearest neighbour method in the clustering problem. *Pattern Recognitions*, 1993. 26:741–746.
33. Dombi J. Membership function as an evaluation. *Fuzzy sets and systems*, 1990. 35:1–21.
34. Kaya M and Alhajj R. A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. In *fuzzy systems, 2003, FUZZ'03, The 12th IEEE international conference on*, 2003. 2:881–886