International Journal of Fuzzy Logic and Intelligent Systems, vol. 4, no. 2, September 2004 pp. 249-253

The Classification of the Software Quality by the Rough Tolerance Class

Wan-Kyoo Choi* and Sung-Joo Lee*

*Department of Computer Engineering, Chosun University
Seoseok-dong, Dong-gu, GwangJu 503-703, Korea

Abstract

When we decide the software quality on the basis of the software measurement, the transitive property which is a requirement for an equivalence relation is not always satisfied. Therefore, we propose a scheme for classifying the software quality that employs a tolerance relation instead of an equivalence relation. Given the experimental data set, the proposed scheme generates the tolerant classes for elements in the experiment data set, and generates the tolerant ranges for classifying the software quality by clustering the means of the tolerance classes. Through the experiment, we showed that the proposed scheme could product very useful and valid results. That is, it has no problems that we use as the criteria for classifying the software quality the tolerant ranges generated by the proposed scheme.

Key Words : Rough tolerance class, Software quality

Ⅰ. Introduction

Because the increase of the software cost of the total system cost, the interest in the software complexity and in the quality measurement has been augmented, and many measures have been proposed. The software measurement shows such important informations as maintainability, understanding and complexity of the software[4].

When we decide the software quality on the basis of the software measurement, we do it as the followings.

1. Define such linguistic variables as "it is easy to maintain" or "it is not easy to maintain".
2. Decide the ranges of the measurement value corresponding to the linguistic variables.
3. Decide that any software belongs to a specific linguistic variable according to the measurement value.

When we decide that the structure of any program is more complex than is necessary because the cyclomatic number of it is more than 20, we apply the above process to our decision.

Many researches[1,3,6,7,9,11] suggest the various ranges for classifying the software quality, but they are based on an equivalence relation. When we decide the software quality based on the software measurement, the transitive property which is a requirement for an equivalence relations is not always satisfied.

For example, we define two linguistic variables about LOC(Line Of Codes), "complex" and "non-complex", and choose the ranges corresponding to two linguistic variables as "under 20" and "20 or more", respectively.

In this case, we must decide that any program whose LOC is 19 is complex and the other program whose LOC is 20 is non-complex. However, it is more reasonable to decide that two programs have an similar degree of the complexity rather than the above decision.

This problem can happen whenever we decide the software quality by using an equivalence relation. This problem can be solved by using a tolerance relation \cite{Funa96,Slow94} instead of an equivalence relation. Therefore, in this paper, we propose a scheme which generates the tolerant ranges for classifying the software quality. We shows that we can generate the tolerant ranges for classifying the software by applying the tolerance relation to the representation of the similarity relation of the data.

Ⅱ. Tolerance relation

The rough set theory is based on the assumption that we have initially some knowledge about elements of the universe. Because with some elements the same information can be associates, hence two different elements can be indiscernible in view of the available information. Thus information indiscernibility relation on its elements. The indiscernibility relation is the starting point of rough set theory and can be employed in two ways in order to define basic concepts of this theory - by defining approximations or the rough membership function[14].

The rough set defines two basic operations on sets, the called *R-lower* and the *R-upper approximation*, and defines respectively by

$$R_* = \{x \in U | R(x) \subseteq X\}$$

$$R^* = \{x \in U | R(x) \cap X \neq \phi\}$$

The difference between the upper and the lower approximation will be called the *R-boundary* of X and will be

Manuscript received Oct. 31, 2003; revised Aug. 18, 2004.

denoted by $BN_R(X)$, i.e.

$$BN_R(X) = R^*(X) - R_*(X).$$

This is to mean that if we "see" the set X through the information , which generates the indiscernibility R, only the above approximation of X can be "observed", but not the set X. The boundary region expresses how exactly the set X can be "seen" due to the indiscernibility X. If boundary region is empty set, X can be "obseved" exactly through the indiscrnibility relation R, and the opposite case the set X can be "observed" roughly only - due to the indiscrnibility R. The former sets are crisp, whereas the later - are rough, with respect to indiscrnibility R, or formally set X is *R-exact* iff $BN_R(X) = \phi$, i.e. $R^*(X) = R_*(X)$. Otherwise the set X is *R-rough*.

If data x, y and z satisfy the equivalence relation, they must satisfy the following properties.

1. The reflexive: $_xR_x$
2. The symmetric: $_xR_y$ -> $_yR_x$
3. The transitive: $_xR_y$ *and* $_yR_z$ -> $_xR_z$

In case of classifying the data, the transitive property which is a requirement for an equivalence relations is not always satisfied. Thus it is not reasonable to represent the similarity of the data on the basis of an equivalence relation[13]. For example, let *x, y* and *z* the elements of any data set. When *x* and *y* belong to the same linguistic variable and *y* and *z* belong to the same linguistic variable, *x* and *z* does not always belong to the same linguistic variable. This case always occurs on the boundary area, on which two linguistic variable are adjacent. Therefore, in case of classifying the data, we must represent the similarity relation between the data by a tolerance relation, which satisfy only the reflexive and the symmetric[2].

When we classify the software quality on the basis of the measurement value by the software measures, the same problem happens as classifying the data. This case also does not always satisfy the transitive. Therefore, the similarity relation between the software must be represented by the tolerance relation.

Let *U* be the universal set of the data, τ be the tolerance relation about any property, and *T(x)* be a set of the elements having the tolerance relation with *x*. Then *T(x)* is defined as the following[2, 13].

$$T(x) = \{y \in U | x\ \tau\ y\}$$

In general, a tolerance relation is represented by the similarity measure, which shows an degree of the similarity between two elements[5]. The similarity measure can is variously defined according to the addressed issues, but its general property is as following: Let the similarity measure between the property values of two data objects *s(x, y)*. Then, when *s(x, y)>a* two data object *x* and *y* is said to have the tolerance relation. *a* is decided according to the addressed issues and is used to judge whether two data objects have the tolerance relation or not[13].

The tolerance class $\tau(x_i)$ of any element $x_i \in U$ is defined by using the similarity measure as followings.

$$\tau(x_i) = \{x_j | s(x_i, x_j) > a, x_i, x_j \in U, j \neq i, j = 1, \cdots, n\} \cup \{x_i\}$$

III. Scheme for generating the tolerant ranges

We propose a scheme for generating the tolerant ranges for classifying the software quality. Figure 1 shows our scheme. When the tolerant classes are given, it generates *k* tolerant ranges from them.

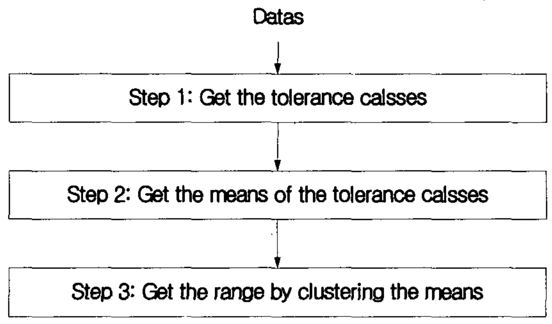

Datas
Step 1: Get the tolerance calsses
Step 2: Get the means of the tolerance calsses
Step 3: Get the range by clustering the means

Fig.1 The process for generating the tolerant ranges

In step 1, we obtains the tolerant classes by using the similarity measure. The tolerant class for any data object *x* is the set of elements whose degree of the similarity to *x* is more than *a*. The number of the tolerant classes generated from *n* data objects is *n*.

In step 2, we calculates the means of the elements which belong to each tolerant class.

In step 3, we cluster the means from step 2 as *k* groups and obtain the tolerant ranges for classifying the software quality.

We employ K-Means algorithm proposed by MacQueen. This algorithm classifies the data objects into ***k*** clusters, calculates the center value from the means of the data objects which is belonged to the cluster, calculates an distance from the center values to each data object, and includes each data objects into the cluster of the most near distance.

By the process of figure 1, we can get ***k*** ranges corresponding to ***k*** linguistic variables which satisfy the tolerance relation. When we classify programs by using the measurement value by the software measure, we can decide what tolerant range they belongs to. That is, a program is classified into $G_i(i=1, 2, \cdots, k)$, which is a linguistic variable for classifying the software.

IV. Experiment and Results

We applied our scheme to the set of the LOC measurement values about the modules written by C-language and generated the tolerant ranges for classifying the software quality by LOC. The modules were obtained from the source code of Linux kernal and Ansi-C runtime library. The number of modules were 18404, and the total lines of all modules were 533165.

In general, the similarity measure is defined by using such distance functions as Hamming distance, Euclidian distance, etc.. However they cannot be applied when classifying the program objects on the basis of the measurement values by the software measures because they cannot reflect the psychological distance.

For example, when we classify the programs by using LOC, any program of 10 lines and the other program of 20 lines have the obvious psychological difference, but any program of 100 lines and the other program of 110 lines does not have as obvious psychological difference as the former.

Therefore, we define the similarity measure for LOC by using the fuzzy membership function. Let *U* be a universal set for LOC values of the programs and x_{max} be the pre-determinated maximum value. The similarity measure between $x_i \in U$ and $x_j \in U$ is defined as followings.

$$s(x_i, x_j) = \mu(x_i, x_j) \wedge \mu(x_j, x_i)$$

$$\mu(x_i, x_j) = \frac{1}{1+(x_j-x_i)^2(x_{max}+1-x_i)/x_{max}}$$

$$(j \neq i, j=1, \cdots, n)$$

When we assume x_{max}=100, figure 2 shows an degree of the similarity between a program of 10 lines and the others and between a program of 90 lines and the others. In this figure, the range of the similar programs to a program of 10 lines and the range of the similar programs to a program of 90 lines are obviously different. Also, larger the value of LOC is, wider the range of the similarity is.

When the size of a population is more that 20000 and the confidence interval is 95% and the tolerant error is within ±1%, the optimal sample size is 8213[8].

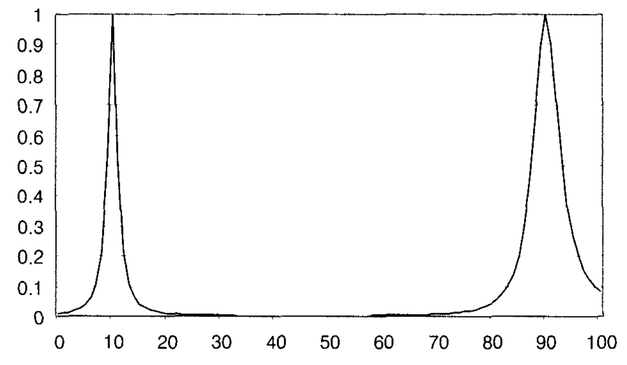


Figure 2. An degree of the similarity between a program of 10 lines and the others and between a program of 90 lines and the others

Thus, we randomly retrieved 8213 modules from the experimental set consisted of 18404 modules, and made them the experimental group T_A, and made the rest the experimental group T_B. We generated the tolerant ranges for classifying the software from T_A, applied the generated ranges to T_A and T_B, and compare the result from T_A with the one from T_B.

The maximum value of LOC of the experimental set was 100. When *a* was 0.1 and the number of the linguistic variables were "small program", "medium program" and "large program", we could obtain the tolerant ranges corresponding to each linguistic variable as like table 1 from T_A.

Table 1. Means of performance

Small program	Medium Program	Large program
$0 \leq LOC \leq 31$	$26 \leq LOC \leq 61$	$54 \leq LOC \leq 100$

When the tolerant ranges of table 1 was applied to T_A and T_B, the number of modules belonging to each linguistic variable in T_A and T_B were as like table 2.

Table 2. Means of performance

Experimental group	Small program	Medium Program	Large program
T_A	5426	2863	1089
T_B	6681	3413	1416

For testing the goodness of the tolerant ranges of table 1, we compared the characteristics of two experimental groups, T_A and T_B. That is, we compared the characteristics of the data set belonged to same linguistic variables by table 1. For comparing the characteristics, we tested that the classified data sets statistically had the significant difference by inferring the difference of the population mean of them.

When inferring the difference of the population mean, two assumptions are needed. The first is the assumption of a normal distribution, that is, the distribution of the sample is approximately normal. The second is the assumption of the homogeneity of variances, because we can't previously know the population variance in most cases.

Table 3 shows the descriptive statistics of T_A and T_B, which are generated by SPSS. We can know that the distribution of the experimental groups is approximately normal on the basis of the central limit theorem and of the fact that the skewness and the Kurtosis are close to 0 in table 3.

Table 3. Descriptive statistics of T_A and T_B

Experimental group	Mean	Std. Deviation	Skewness	Kurtosis
T_A	29.1093	20.6437	1.269	1.092
T_B	29.1127	20.8118	1.269	1.035

Let σ^2_A and σ^2_B be the population variances of T_A and T_B, respectively. We retrieved Levene's statistic in order to test the assumption of the homogeneity of variances(H_0 : $\sigma^2_A = \sigma^2_B$). Table 4 shows Levene's statistic when the confidence interval is 95%. In each case, we cannot reject H0 because Levene's statistics are enough large and p(=significance probability) > 0.05(=significance level)[12].

Table 4. Test of Homogeneity of Variances

Categories	Levene statistic	df1	df2	Significance probability(p)
Small program	2.248	1	12105	0.134
Medium program	4.507	1	6275	0.340
Large program	0.202	1	2503	0.653

Table 5. ANOVA Combined Between Groups

Categories	Sum of Squares	df	Mean Square	F	Significance probability(p)
Small program	35.607	1	35.607	0.719	0.397
Medium program	166.389	1	166.389	1.666	0.197
Large program	202.570	1	202.570	1.217	0.270

Let μ^2_A and μ^2_B be the population means of T_A and T_B, respectively. By the result of table 4, we can know that the assumption of the homogeneity of variances is satisfied. Therefore, under the assumption of the homogeneity of variances, we performed the one-way ANOVA(ANalysis Of VAriance) in order to test that the classified data sets statistically had the significant difference(H_0 : $\mu^2_A = \mu^2_B$). Table 5 shows ANOVA statistics when the confidence interval is 95%. In each case, we cannot reject H_0 because p(=significance probability) > 0.05(=significance level)[12].

Therefore, we could conclude that the characteristics of two experimental groups is not different when classifying two experimental groups by table~\ref{table: category}.

This shows that our scheme for generating the tolerant ranges can product very useful and valid results only if the given assumptions can be satisfied and that the ranges retrieved by it can be used as the criteria for classifying the software quality.

V. Conclusion

When we classify the software on the basis of the measurement values by the software measures, the transitive property which is a requirement for an equivalence relation is not always satisfied. Therefore, in this paper, we propose a scheme for classifying the software quality by the tolerance relation which satisfies the reflexive and the symmetric.

Our scheme obtains the tolerant classes by using the similarity measure, and calculates the means of the elements which belong to each tolerant class, and cluster the means as k groups, and obtains the tolerant ranges for classifying the software quality.

In experiment, we applied our scheme to LOC values of 18302 modules written by C-language. We obtained the tolerant ranges from an experimental set, applied the obtained ranges to two experimental groups, and compared their characteristics. As result, we could conclude that their characteristics is not different, that is, the obtained ranges can be used as the criteria for classifying the software.

Reference

[1] Caldiera, G. and V.R. Basili, "Identifying and Qualifying Reusable Software Components", IEEE Computer, pp.61-70, Feb. 1991,

[2] K.Funakoshi and T.B.Ho, "Information retrieval by rough tolerance relation", The 4th international Workshop on rough sets, Fuzzy sets, and Machine Discovery, Tokyo, Nov. 1996.

[3] Horst Zuse, Software Complexity-Measures and Methods, New York: Walter de Gruyter, pp.25-37, 1991.

[4] Karl J. Ottensteion, Linda M. Ottensteion, "The program dependence graph in a software development environment", ACM SIGPLAN Notices, vol.19, no.5, pp.177-184, May, 1984.

[5] M.Kretowski and J. Stepniuk, "Selection of objects and attributes a tolerance rough set approach", ICS Research Reports, 1994.

[6] Lewis John, Henry Salie, "A Methodology for Integrating Maintainability Using Software Metrics", Proceedings: Conference on Software Maintenance, Miami, Florida, IEEE, pp.32-39, Oct. 1989.

[7] Lowell J. Arthur, Measuring Programmer Productivity and Software Quality, New York:John Wiley \& Sons, pp.138-142, 1985.

[8] D.J.Luck, H.G. Wales, D.H.Taylor, Marketing Research, N.J.:Prentice-Hall, pp.611-612, 1970.

[9] T.McCabe, "A Complexity Measure", IEEE Trans.SE., SE-2, pp.308-320, 1976.

[10] Slowwinski R. and Vanderpooten D. "Similarity relations as a basic for rough approximations, ICS Research Reports, 1994.

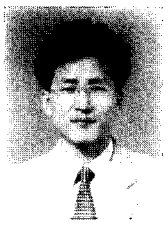
[11] Szentes J., Gras j., "Some Practical Views of Software - Complexity metrics and a Universal Measurement Tool", First Australian Software Engineering Conference, Canberra, pp.14-16, May 1986.

[12] John Neter, William Wasserman and Michael H. Kutner, Applied Linear Statistical Models, IRWIN, Boston, 1990.

[13] Daijin Kim and Chul-Hyun Kim, "Handwritten

Numerical Character Recognition Using the Tolerant Rough Set", Journal of Fuzzy logic and Intelligent Sytems, vol.9, no.1, pp.113-123, 1999.

[14] Pawlak Z., "Rough Sets-Theoretical Aspects of Reasoning about Data", Kluwer Academic Publishers, London, 1991.



Wan-Kyoo Choi

Wan-Kyoo Choi was born October 16. 1964. He received the B.A. degree in Department of Religious Studies from Seoul National University, Seoul, Korea, in 1983, and the M.S. and Ph. D. degrees in Department of Computer Science f7||m Chosun University, GwangJu, Korea, in 1997 and 200 respectively. He was in the employ of Kongsung Communication Ltd. from 1992 to1993 and Han-yang System Ltd. form 1993 to 1995 respectively. Since 2000, he has been a faculty member of the Department of Computer Science at Kwangju University, where he is currently a Full Time Lecturer. His research interests are Software Engineering, Object-oriented System, Fuzzy Sets and Rough Sets.

Phone : +82-62-972-8899
Fax : +82-62-233-6896
E-mail : wankyoo@empal.com

Sung-Joo Lee

Sung-Joo Lee was born October 31. 1943. He received the B.S. degree in Department of Physics from Hannam University, Daejeonl, Korea, in 1970, and the M.S. degree in Department of Computer Science form Kwang-Un University, Seoul, Korea, in 1992, and Ph. D. degree in Department of Computer Science from DaeGu Catholic University, DaeGu, Korea, in 1998. Since 1981, he has been a faculty member of the Department of Computer Science at ChoSun University, where he is currently a Professor. His research interests are Software Engineering, Programming Language, Object-oriented System, Rough Set.

Phone : +82-62-230-7710
Fax : +82-62-233-6896
E-mail : sjlee@chosun.ac.kr