Hindawi
International Journal of Digital Multimedia Broadcasting
Volume 2018, Article ID 7624768, 9 pages
https://doi.org/10.1155/2018/7624768

Research Article

A Novel Method of Complexity Metric for Object-Oriented Software

Tong Yi and Chun Fang

School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330013, China

Correspondence should be addressed to Tong Yi; 1337742168@qq.com

Received 8 September 2018; Accepted 16 October 2018; Published 1 November 2018

Guest Editor: Yuanlong Cao

With the rapid development and wide application of multimedia technology, the demand for the actual development of multimedia software in many industries is increasing. How to measure and improve the quality of multimedia software is an important problem to be solved urgently. In order to calculate the complicated situation and fuzziness of software quality, this paper introduced a software quality evaluation model based on the fuzzy matter element by using a method known as the fuzzy matter element analysis, combined with the TOPSIS method and the close degree. Compared with the existing typical software measurement methods, the results are basically consistent with the typical software measurement results. Then, Pearson simple correlation coefficient was used to analyse the correlation between the existing four measurement methods and the metric of practical experience, whose results show that the results of software quality measures based on fuzzy matter element are more in accordance with practical experience. Meanwhile, the results of this method are much more precise than the results of the other measurement methods.

1. Introduction

At present, with the rise and application of multimedia technology, it is a great challenge to provide more reliable technical support and strong technical support for the development of multimedia software. At the same time, object-oriented technology has become the mainstream of current software development, which is suitable for developing multimedia software, for example, using the image processing software Adobe Photoshop developed by C++, using Action Script to develop animation processing software Flash, and using C++ for the Jedi survival and heroic alliance games.

We must point out that multimedia software is a typical complex system; therefore, how to scientifically measure the complexity of multimedia software plays a vital role in developing high-quality multimedia software. Software metrics has become the important and long-term focused research field of software engineering and also became an important and effective method in assessing and predicting software development activities. The purpose of software metrics research is to provide guidance for developing high-quality software [1].

Since the concept of software measurement was first proposed by Rubey R. J. and Hartwick R. D. in 1968[2], the researches, development, and applications have been carried out for more than fifty years. Through literature review, this paper found that previous researches mainly from internal attributes, external attributes, and other aspects of the research of software quality metric. Over these years, many scholars have made a broad and deep research on the software quality metric and prefer to find the key or the important software quality measurement factors from the inner elements of software itself. The factors were measured or counted directly or indirectly to construct the corresponding metric model. Early metrics on structured programs were primarily focused on Lines of Code (LOC) [3], McCabe coloring graph method [4], Function Point Analysis (FPA) [5], etc.

In 1994, Chidamber S. and Kemerer C. proposed a CK metrics set for object-oriented software quality metrics research. The Weighted Methods per Class (WMC), Number of Children (NOC), Depth of Inheritance (DIT), Coupling Between Objects (CBO), Lack of Cohesion (LCOM), and Response for a Class (RFC) are included in set, which are the fundamental of object-oriented software quality metrics.

Padhy N. et al. proposed the three metrics based on CK metrics set and combined WMC, RFC, CBO, DIT. and NOC together [6]. In addition, Misra S. and Adewumi A. et al. proposed a cognitive complexity metrics set for evaluating object-oriented software projects [7], including method complexity, message complexity, attribute complexity, weighted class complexity, and code complexity. According to software measurement experience, Gupta D. L. et al. proposed some possible exist the hypothetical situation in measurement validation and design 14 measurement elements, including WMC, CBO, and RFC. Furthermore, Gupta D. L. et al. took open source software code as the data source and used SPSS software make logistic regression analysis. The results of the study showed that these methods can predict design flaw of class in software quality metric, and software defects prediction methods based on object-oriented metrics are developed [8]. Wang J. and Wang Q. found that dependency relationship is an important reason of software complexity. The dependency relationship can reflect cohesion and coupling between software elements. Meanwhile, cohesion and coupling are recognized as a measure of software quality of the important indicators. Besides, the dependency relationship of software is proved to be an important factor of software defects prediction through the experimental study. It can predict software integration errors and provide help for software quality metric in early stage [9]. The above methods of object-oriented software metrics are all belong to research of software quality metric based on software internal attributes.

However, developers and researchers paid attention to broad software quality characteristics in the process of software quality metric research based on external attributes of software quality. These characteristics include software quality characteristics of ISO/IEC 25010 software quality model in narrow sense and other software quality characteristics associated with software development and application. Gosain A. and Sharma G. defined the dynamic software quality characteristics, including robust, unambiguous, dynamic, discriminating, and machine independent. Then they evaluated cases with Java software and found that the dynamic software quality characteristic has significant positive correlation with maintainability by Pearson correlation analysis and principal component analysis [10]. Similarly, Hu X and Zuo J. et al. choose 6 software quality characteristics from GB/T16260 series of standards. The 6 software quality characteristics include capability, reliability, usability, efficiency, maintainability, and portability. Then the hierarchical model of evaluation is established for research and analysis external attributes of software quality [11].

Class diagram, a very important software model diagram, describes the classes and their relationships among the systems. They can be scientifically constructed whether or not it has a significant impact on the complexity of software. At present, the class complexity measure method is still rare. Marchesi M. [12] uses 7 indicators to measure the complexity of the class diagram from different angles. However, the method only considers the relationship between classes and inheritance, without considering other relationships, such as the association relationship and aggregation relationship. On the basis of Marchesi M. research, Genero M. [13] uses 14 indicators to further distinguish the relationship between classes and classes, that is, the combination of relative complexity measure and absolute complexity measure. The theories of Dr. Zhang Y. [14], In P. [15], Gosain A. [10], Gupta D. L. [8], and Padhy N. [6] are similar with Genero M's, which use a set of indicators to evaluate the complexity of class diagrams. The advantage of it could analyse the complexity of a class diagram from different perspectives, but its disadvantage is that it is difficult to compare two or two class diagrams. Dr. Zhou Y. transforms UML class diagrams into weighted dependencies. And then he uses the information entropy to define the complexity of UML class diagram [16], which has achieved good measure results. Dr. Yi T. has made improvements on the basis of Dr. Zhou Y. making a comprehensive consideration of interclass relationships, class attributes, and class complexity of the method. He proposed a UML class diagram complexity measurement method based on dependency analysis [17, 18].

In this paper, the research work mentioned above is a part of existing domestic and international research work, but there is no doubt that the results of researches in the UML class diagram model are not enough. One of the important reasons is that UML standard issued by the object management group (OMG) only gives the description of the semantic conceptual level in various modelling elements, which leads to the fact that the researchers often use different weighting indicators for the class diagram model. It means that researchers do not have a uniform standard, resulting in different metrics for the same class diagram. Meanwhile, because of the comprehensiveness, fuzziness, and complexity of the software quality measurement system, the software quality measurement is a process of multiple indicator decision making; the fuzzy matter element theory is introduced in this paper. In order to overcome the limitation of weight precision of the class relationship between two classes in the literature [16–18], this paper proceeds from fuzzy matter element theory, introducing the concept of close degree, and used entropy method to calculate the weight of every indicator; software quality measurement model was established in fuzzy matter element that based on entropy weight and TOPSIS method applied to UML class diagram metric. Firstly, element indicators of UML class diagram constitute the compound fuzzy matrix of matter elements and then fuzzy matrix of matter elements of the optimal subordinate degree obtained with the dimensionless, calculating the weight of each element indicators by entropy method, finally, through TOPSIS method and the concept of Euclid approach degree got comprehensive attribute values of each UML class diagrams. This paper hopes to only use a comprehensive complexity value to evaluate the complexity of UML class diagram and enough really predicts the complexity of software quality.

2. A Novel Method of Complexity Metric for Software Quality

2.1. Building Evaluate Compound Fuzzy Matter Element of Software Quality. Matter element analysis [19] is a new

discipline that studies laws and methods for solving incompatible problems. It is an intersecting edge discipline of thinking science, systems science, and mathematics. Matter element analysis itself is not a branch of mathematics. It is a new discipline that develops on the basis of classical mathematics and fuzzy mathematics and is different from them. The new subject, Matter Element Analysis, which was created in 1994 by Chinese scholar Cai Wen, was specifically designed to solve incompatible problems. The fuzzy matter element combines fuzzy set theory and matter element analysis theory, which can not only solve the ambiguity of measurement indicators, but also solve the incompatibility of measurement results. Because of its simple calculation method, reliable evaluation results, and strong practicality, this theory is widely used in logistics science and technology [20], electromechanical [21], architecture [22], and other fields.

The matter element $R = (T, C, X)$ for evaluating the software quality was constructed in this paper, where T denotes the software class diagram to be evaluated, C denotes the evaluation indicator, and X denotes the corresponding magnitude of the evaluation indicator. If X has ambiguity, R is called a fuzzy matter element. If T has n evaluation indicators $C_1, C_2, \ldots, C_n$ whose corresponding magnitudes are $X_1, X_2, \ldots, X_n$, R is said to be n-dimensional fuzzy matter elements[23]. The n-dimensional matter elements of m the software diagrams to be evaluated are combined to form the n-dimensional compound fuzzy matter elements of m the software diagram to be evaluated. R_{mn} is defined as follows:

$$R_{mn} = \begin{pmatrix} & T_1 & T_2 & \cdots & T_m \\ C_1 & X_{11} & X_{21} & \cdots & X_{m1} \\ C_2 & X_{12} & X_{22} & \cdots & X_{m2} \\ \vdots & \vdots & \vdots & & \vdots \\ C_n & X_{1n} & X_{2n} & \cdots & X_{mn} \end{pmatrix} \quad (1)$$

In formula (1), T_i represents the i (i=1,2,... m) software class diagram, C_j is the j (j=1,2,...,n) evaluation indicator of the software class diagram, and X_{ij} represents the corresponding magnitude of the j evaluation indicator of the i software class diagram.

2.2. Dimensionless of Evaluation Indicators. In the evaluation of software class diagrams, there are many evaluation indicators involved. If there are no uniform metrics among the indicators, the evaluation process will be difficult to carry out. In order to compare the different dimension indicators together for comparison, the magnitude of these evaluation indicators must be dimensionless [23]. The dimensionless process is to remove the dimension's influence on the physical value through mathematical methods. There are generally two types of indicators for quantification processing results, some of which are larger and better indicators, that is, positive indicators; others are smaller, better indicators, that is, negative indicators. According to the actual situation, this paper selects the smaller and better indicators in the software quality evaluation.

$$\max_j = \max\left(X_{1j}, X_{2j}, \ldots, X_{mj}\right) \quad (2)$$

$$\min_j = \min\left(X_{1j}, X_{2j}, \ldots, X_{mj}\right) \quad (3)$$

$$u_{ij} = \frac{\max_j - X_{ij}}{\max_j - \min_j} \quad (4)$$

In formula (4), u_{ij} is the dimensionless result of the j-th evaluation indicator of the i-th software class diagram. $\max_j$ is the maximum value of the j-th evaluation indicator of the software class diagram, and $\min_j$ is the minimum value of the j-th evaluation indicator of the software class diagram.

After the dimensionless treatment of formula (1) through formula (4), formula (5) is obtained, that is, the fuzzy matter element weight matrix of optimal membership degree R'_{mn}.

$$R'_{mn} = \begin{pmatrix} & T_1 & T_2 & \cdots & T_m \\ C_1 & u_{11} & u_{21} & \cdots & u_{m1} \\ C_2 & u_{12} & u_{22} & \cdots & u_{m2} \\ \vdots & \vdots & \vdots & & \vdots \\ C_n & u_{1n} & u_{2n} & \cdots & u_{mn} \end{pmatrix} \quad (5)$$

2.3. Evaluation Indicator Weight Determining Based on Entropy Method. In the process of software quality evaluation, the weight of an indicator reflects the relative importance of the indicator in the overall evaluation process. Therefore, the determination of weight is very important. Common weight determination methods include entropy method, expert scoring method, and analytic hierarchy process. This paper uses entropy method to calculate weights to achieve the subjective and objective unity of weights. The entropy method is based on the difference in the degree of information contained in each indicator, that is, the utility value of the information to determine the weight of the indicator. It is an objective weighting method.

The formula for calculating the information entropy and weight function in the comprehensive evaluation is as follows: For the software quality evaluation model in question, if there are initial data matrix R_{mn} of the n evaluation indicators of the m software class diagram to be evaluated, each indicator is significantly different in the dimension, order of magnitude, and merits of indicators. Therefore, the initial data must be standardized:

$$y_{ij} = \frac{X_{ij}}{\sum_{i=1}^{m} X_{ij}}, \quad i = 1, 2, \ldots, m; \; j = 1, 2, \ldots, n \quad (6)$$

Get information entropy of the j-th evaluation indicator according to formula (7):

$$e_j = -k \sum_{i=1}^{m} y_{ij} \ln y_{ij}, \quad j = 1, 2, \ldots, n \quad (7)$$

The constant k in formula (7) is related to the number of samples, m, and $k = 1/\ln m$ is often taken. Because of information entropy e_j can be used to measure the information utility value of the j-th evaluation indicator. When the sample is completely disordered, e_j =1; meanwhile, the information value of e_j is zero for the utility value of the comprehensive evaluation. Therefore, the information utility value of an evaluation indicator is determined by the difference between 1 and the information entropy e_j of the evaluation indicator; that is,

$$h_j = 1 - e_j, \quad j = 1, 2, \ldots, n \tag{8}$$

The entropy method is used to estimate the weight of the evaluation indicator. Its essence is to use the information utility value of the evaluation indicator to measure. When the difference h_j is higher, the importance of the evaluation is bigger, so the weight of the j-th evaluation indicator is

$$w_j = \frac{h_j}{\sum_{j=1}^{n} h_j}, \quad j = 1, 2, \ldots, n \tag{9}$$

Fuzzy matter element weight matrix of optimal membership degree is

$$R_w = \begin{pmatrix} & C_1 & C_2 & \cdots & C_n \\ w_j & w_1 & w_2 & \cdots & w_n \end{pmatrix} \tag{10}$$

2.4. Fuzzy Compound Matter Element for Evaluating the Quality Characteristics. R_s is a weighted fuzzy compound matter element for evaluating the quality characteristics, and then there are

$$R_s = \begin{pmatrix} & T_1 & T_2 & \cdots & T_m \\ C_1 & C_{11} = w_1 u_{11} & C_{21} = w_1 u_{21} & & C_{m1} = w_1 u_{m1} \\ C_2 & C_{12} = w_2 u_{12} & C_{22} = w_2 u_{22} & & C_{m1} = w_2 u_{m2} \\ \vdots & & & & \\ C_n & C_{1n} = w_n u_{1n} & C_{2n} = w_n u_{2n} & & C_{mn} = w_n u_{mn} \end{pmatrix} \tag{11}$$

In formula (11), $C_{ij} (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ the calculation value of the j-th evaluation indicator of the i-th software class diagram is represented.

2.5. Calculating Comprehensive Evaluation of Software Quality. TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [24] is a multiobjective decision-making method. The basic idea is to define the ideal solution and negative ideal solution of the decision problem. It is assumed that the ideal solution is the optimal program and the negative ideal solution is the worst program. If there is an evaluation plan in the feasible evaluation plan, the evaluation plan is the closest to the ideal, while far away from the negative ideal solution, we call this program the optimal program.

Further determine the ideal solution vector C^+ and negative ideal solution vector C^- of matrix R_s:

$$\begin{aligned} C^+ &= \left(C_1^+, C_2^+, \ldots, C_n^+\right), \\ C_j^+ &= \max\left\{C_{1j}, C_{2j}, \ldots, C_{mj}\right\}, \quad j = 1, 2, \ldots, n \end{aligned} \tag{12}$$

$$\begin{aligned} C^- &= \left(C_1^-, C_2^-, \ldots, C_n^-\right), \\ C_j^- &= \min\left\{C_{1j}, C_{2j}, \ldots, C_{mj}\right\}, \quad j = 1, 2, \ldots, n \end{aligned} \tag{13}$$

There are several ways to calculate the distance between ideal solutions and negative ideal solutions, such as Euclidean distance, Manhattan distance, Chebyshev distance, and so on. Among them, Euclidean distance is an easy-to-understand distance calculation method, which is derived from the distance formula between two points in Euclidean geometry. In this paper, the Euclidean distance is used, and its calculation formula is as follows [23]:

$$S_i^+ = \left\|C^+ - C_i\right\| = \sqrt{\sum_{j=1}^{n} \left(C^+ - C_{ij}\right)^2}, \quad i = 1, 2, \ldots, m \tag{14}$$

$$S_i^- = \left\|C_i - C^-\right\| = \sqrt{\sum_{j=1}^{n} \left(C_{ij} - C^-\right)^2}, \quad i = 1, 2, \ldots, m \tag{15}$$

In formula (14) and formula (15), $C_i = (C_{i1}, C_{i2}, \cdots, C_{in})^T$; C_i is the i-th column vector of matrix R_s.

In this paper, the comprehensive evaluation of software quality adopts entropy method for consideration. The source has S_i^+ and S_i^-. The binary entropy function can be used to calculate the weights of the Euclidean distance between each class diagram to be evaluated and the ideal solution.

$$H^+ = H\left(S_i^+\right) = -S_i^+ \log_2 S_i^+ - \left(1 - S_i^+\right) \log_2 \left(1 - S_i^+\right), \quad i = 1, 2, \ldots, m \tag{16}$$

Similarly, the binary entropy function is used to calculate the weights of the Euclidean distances of each class diagram to be evaluated and the negative ideal solution; namely,

$$H^- = H\left(S_i^-\right) = -S_i^- \log_2 S_i^- - \left(1 - S_i^-\right) \log_2 \left(1 - S_i^-\right), \quad i = 1, 2, \ldots, m \tag{17}$$

According to the concept of close degree [23], combined with the uncertainty of the ideal solution and the negative ideal solution, the fuzzy matter element software quality metric measures the software quality by the following uncertainty-weighted fusion method. The calculation formula is as follows:

$$Z_i = \frac{H^+ S_i^+}{H^+ S_i^+ + H^- S_i^-}, \quad i = 1, 2, \ldots, m \tag{18}$$

In formula (18), the value is between 0 and 1. The closer the value is to 0, the evaluation object complexity is smaller and the closer to the optimal ideal level.

3. Case Analysis

3.1. Data Sources. In order to validate the measurement method proposed in this paper, we will do an experiment to estimate the metric values. With the permission of Genero M., we selected twenty-six UML class diagrams [13] related to the bank information systems as the object of the experiment. For better representation, NDep represents dependency, NAssoc represents normal association, NAgg represents aggregation, NGen represents generalization, NM represents class method, NA represents class attribute, and NC represents the number of classes. For specific indicators and data for details, see Table 1.

3.2. Model Establishment. According to the above theory and evaluation indicator system, the steps for establishing a fuzzy matter element evaluation model are as follows.

Step 1. Construct the composite fuzzy matrix of matter elements according to Table 1.

Step 2. Calculate the degree of optimal membership. According to the compound fuzzy matter element matrix determined in the first step, the degree of optimal membership is calculated using formula (4), and the fuzzy matter element matrix of optimal membership degree is obtained.

$$R'_{26\times 7} = \begin{pmatrix} & T_1 & T_2 & T_3 & T_4 & \cdots & T_{25} & T_{26} \\ NDep & 1 & 1 & 1 & 1 & \cdots & 0 & 1 \\ NAssoc & 0.928571 & 0.928571 & 0.928571 & 0.785714 & \cdots & 0 & 0.642857 \\ NAgg & 1 & 0.888889 & 0.777778 & 1 & \cdots & 0.555556 & 0 \\ NGen & 1 & 1 & 1 & 1 & \cdots & 0.333333 & 0.708333 \\ NM & 1 & 0.955556 & 0.922222 & 0.955556 & \cdots & 0.155556 & 0.233333 \\ NA & 1 & 0.961538 & 0.903846 & 0.942308 & \cdots & 0.269231 & 0.423077 \\ NC & 1 & 0.967742 & 0.935484 & 0.967742 & \cdots & 0.354839 & 0.612903 \end{pmatrix} \tag{19}$$

Step 3. Based on the fuzzy matter element matrix of optimal membership degree $R'_{27\times 7}$, according to formula (6), formula (7), formula (8), formula (9), formula (10), and formula (11) by entropy method to obtain each indicator weights that composes the fuzzy matter element weight matrix of optimal membership degree R_w,

$$R_w = \begin{pmatrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\ w_j & 0.17124 & 0.143127 & 0.150199 & 0.124097 & 0.189334 & 0.191128 & 0.184991 \end{pmatrix} \tag{20}$$

Step 4. Get R_s by formula (11).

$$R_s = \begin{pmatrix} & T_1 & T_2 & T_3 & T_4 & \cdots & T_{25} & T_{26} \\ NDep & 0.017124 & 0.017124 & 0.017124 & 0.017124 & \cdots & 0 & 0.017124 \\ NAssoc & 0.132904 & 0.132904 & 0.132904 & 0.112457 & \cdots & 0 & 0.09201 \\ NAgg & 0.150199 & 0.13351 & 0.116821 & 0.150199 & \cdots & 0.083444 & 0 \\ NGen & 0.124097 & 0.124097 & 0.124097 & 0.124097 & \cdots & 0.041366 & 0.087902 \\ NM & 0.189334 & 0.18092 & 0.174608 & 0.18092 & \cdots & 0.029452 & 0.044178 \\ NA & 0.191128 & 0.183777 & 0.17275 & 0.180101 & \cdots & 0.051458 & 0.080862 \\ NC & 0.184991 & 0.179024 & 0.173056 & 0.179024 & \cdots & 0.065642 & 0.113382 \end{pmatrix} \tag{21}$$

Table 1: Twenty-six UML class diagrams evaluation indicators.

Class diagrams	NDep	NAssoc	NAgg	NGen	NM	NA	NC
1	0	1	0	0	8	4	2
2	0	1	1	0	12	6	3
3	0	1	2	0	15	9	4
4	0	3	0	0	12	7	3
5	0	1	3	0	21	14	5
6	0	2	0	0	12	6	3
7	1	3	0	0	13	8	4
8	0	2	2	2	14	10	6
9	1	1	0	0	12	9	3
10	0	2	3	2	22	14	7
11	0	2	3	4	30	18	9
12	0	3	3	2	39	19	7
13	1	3	2	2	35	22	8
14	0	0	0	4	30	11	5
15	0	0	0	10	30	12	8
16	0	0	0	18	38	17	11
17	2	11	6	10	76	42	20
18	1	11	6	16	88	41	23
19	1	7	6	20	94	45	21
20	3	13	7	24	98	56	33
21	0	1	5	2	47	28	9
22	0	3	5	20	65	31	18
23	0	11	6	21	79	44	26
24	0	1	5	19	69	32	17
25	4	14	4	16	84	42	22
26	0	5	9	7	77	34	14

Note: Genero metric values from Genero M.'s experiment in Table 1.

Step 5. Software quality measurement values of fuzzy matter element in this paper are calculated by formula (12), formula (13), formula (14), formula (15), formula (16), formula (17), and formula (18).

$$\begin{aligned} Z_i = (&0.02089, 0.101997, 0.339518, 0.205419,\\ &0.856791, 0.106658, 0.251853, 0.510079,\\ &0.105591, 1.075057, 1.733703, 2.00285,\\ &1.906329, 0.606743, 1.143489, 2.588069,\\ &8.394743, 9.021374, 8.872664, 9.91465,\\ &3.522168, 6.674977, 9.271023, 6.446144,\\ &8.760155, 7.146074) \end{aligned} \tag{22}$$

3.3. Data Analysis

3.3.1. Comparing the Experiment Results of Four Metrics. To verify the effectiveness and practicability of the proposed measurement method, this paper plans to compare with the method proposed by Dr. Zhou Y. [16] and the method proposed by Dr. Yi T. [17, 18] in the three aspects, understandability, analysability, and maintainability. For convenient discussion, the method of Dr. Zhou Y. and the method of Dr. Yi T. are called Zhou metric [16], Yi15 metric [17], and Yi18 metric [18]; the measurement method proposed in this paper is called Z_i metric, as shown in Table 2.

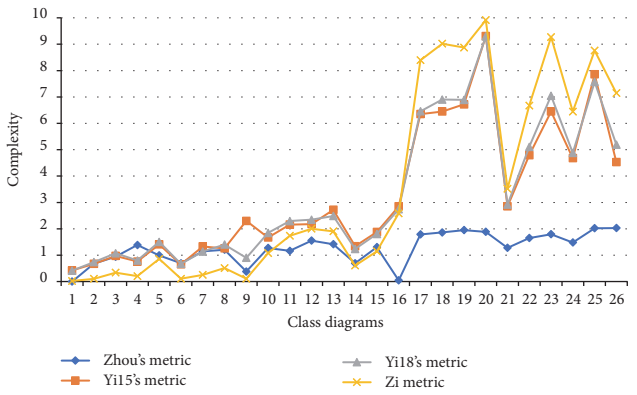

Figure 1: Comparing the experiment results.

Comparing the experimental results of the above four software quality measurement models, as shown in Figure 1,

Table 2: Comparing the experiment results of measurement methods.

Class diagrams	Zhou metric	Yi15 metric	Yi18 metric	Z_i metric	Understandability	Analysability	Maintainability
1	0	0.42	0.41176	0.02089	1	1	1
2	0.673012	0.67	0.73657	0.101997	2	2	2
3	0.940493	0.97	1.07097	0.339518	2	2	2
4	1.386294	0.76	0.7987	0.205419	2	2	2
5	0.989909	1.41	1.50423	0.856791	2	2	2
6	0.693147	0.65	0.66949	0.106658	2	2	2
7	1.14688	1.33	1.13783	0.251853	2	3	3
8	1.206376	1.26	1.40369	0.510079	3	3	3
9	0.381909	2.3	0.9008	0.105591	2	2	2
10	1.271002	1.67	1.8433	1.075057	3	3	3
11	1.16503	2.16	2.29233	1.733703	3	3	3
12	1.553338	2.18	2.34555	2.00285	3	3	3
13	1.414547	2.72	2.48526	1.906329	3	3	3
14	0.693147	1.34	1.237	0.606743	2	2	2
15	1.303487	1.88	1.80312	1.143489	2	3	3
16	0.04308	2.85	2.7398	2.588069	4	4	4
17	1.787461	6.35	6.45495	8.394743	6	6	6
18	1.8612	6.45	6.90285	9.021374	6	6	6
19	1.949444	6.72	6.88925	8.872664	6	5	6
20	1.883662	9.31	9.29632	9.91465	6	6	7
21	1.277816	2.85	2.91541	3.522168	3	3	3
22	1.649751	4.79	5.10804	6.674977	5	5	5
23	1.794866	6.45	7.04766	9.271023	6	6	6
24	1.480208	4.68	4.88021	6.446144	5	5	5
25	2.020782	7.86	7.5702	8.760155	6	5	6
26	2.030221	4.53	5.19316	7.146074	4	5	5

and comparing them with the understandability, analysability, and maintainability of the class diagrams obtained by the practical experience, it is found that the four metric results are similar. But also some interesting results are found.

(1) For the class diagram 4, the Zhou metric has higher values for the computational class diagram 4 complexity, the Yi15 metric and the Yi18 metric have lower values for the complexity of the class diagram 4, and the complexity of the class diagram 4 that the practical experience has obtained has lower values. The complexity of class diagram 4 calculated using the fuzzy matter element model in this paper is low, which is consistent with the actual experience.

(2) For the class diagram 9, the Zhou metric and the Yi18 metric have lower values for the computational class diagram 9 complexity, the Yi15 metric has higher values for the complexity of the class diagram 9, and the complexity of the class diagram 9 that the practical experience has obtained has lower values. The complexity of class diagram 9 calculated using the fuzzy matter element model in this paper is low, which is consistent with the actual experience.

(3) For the class diagram 16, the Zhou metric has lower values for the computational class diagram 16 complexity, the Yi15 metric and the Yi18 metric have higher values for the complexity of the class diagram 16, and the complexity of the class diagram 16 that the practical experience has obtained has higher values. The complexity of class diagram 16 calculated using the fuzzy matter element model in this paper is high, which is consistent with the actual experience.

(4) For the class diagram 19, the Yi18 metric for the complexity of the class diagram 19 has lower values than the class diagram 18, the Zhou metric and the Yi15 metric have higher values for the computational class diagram 19 complexity, and the complexity of the class diagram 19 that the practical experience obtained has lower values than the class diagram 18. The complexity of class diagram 19 calculated using the fuzzy matter element model in this paper is consistent with actual experience.

(5) For the class diagram 25 and the class diagram 26, the Zhou metric shows that the 26th class diagram complexity is higher than the 25th class diagram complexity and the Yi15 metric and Yi18 metric methods show the 26th class diagram complexity lower than it. The complexity of the class diagrams 25 and 26 obtained by the practical experience is opposite with the Zhou metric, which is in consistent with

Table 3: Correlation coefficient and correlation intensity.

Correlation coefficient absolute value	Correlation intensity
$\|r\| = 0$	Zero correlation
$0 < \|r\| \le 0.3$	Weak correlation
$0.3 < \|r\| \le 0.5$	Low correlation
$0.5 < \|r\| \le 0.8$	Significant correlation
$0.8 < \|r\| \le 1$	High correlation
$\|r\| = 1$	Completely correlation

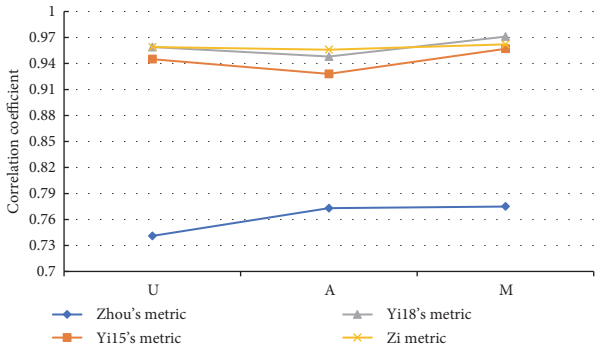

Figure 2: Pearson simple correlation analysis.

the complexity of the class diagram calculated by using the fuzzy matter element model in this paper.

3.3.2. Pearson Simple Correlation Coefficient Test. In order to further discuss the existing correlation between the results of complexity metric and the value of understandability, the value of analysability, and the value of maintainability, we propose the Pearson simple correlation coefficient to test whether or not the complexity measure method is consistent with the practical experience. Pearson simple correlation coefficient is calculated as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sqrt{\sum \left(X_i - \overline{X}\right)^2} \sqrt{\sum \left(Y_i - \overline{Y}\right)^2}} \tag{23}$$

The correlation intensity between the two variables refers to Table 3.

Using the well-known statistical software SPSS for correlation analysis and the results of the correlation analysis are shown in Table 4.

Through the comparison and data analysis in Table 4, we can find that this paper's UML class diagram metric is consistent with practical experience in the understandability, analysability, and maintainability. The value of the UML class diagrams complexity measure Z_i is calculated by fuzzy matter element model, which is compared with practical experience. The Pearson simple correlation coefficient between Z_i and the value of the understandability of practical experience is 0.959. The Pearson simple correlation coefficient between Z_i and the value of the analysability of practical experience is 0.956. The Pearson simple correlation coefficient between Z_i and the value of the maintainability of practical experience is 0.962. Zhou metric is significantly correlated with the class diagram of practical experience. But the fuzzy matter element model metrics, Yi15 metric, and Yi18 metric are highly correlated with the class diagram of practical experience. Therefore, the complexity measure method of this paper is consistent with the practical experience from the view of average.

3.3.3. The Analysis of the Visualization of Measurement Results. In order to compare the abovementioned four metrics methods more intuitive, the classification results of this paper are shown in Figure 2. For better representation, U represents the understandability, A represents the analysability, and M represents the maintainability.

From Figure 2, we can find that this paper's results are closer to 1. It suggests that the class diagram complexity calculated by the fuzzy matter element model is consistent with the value of practical experience by comparing with other metrics. This method can quickly calculate the comprehensive attribute value of the software class diagrams. Meanwhile, the results of this study can be more accurately to reflect the software complexity. So the measurement model proposed in this paper is relatively better.

4. Conclusions

This paper uses the basic theory and method of matter element analysis, combined with fuzzy set theory and TOPSIS method to establish a fuzzy matter element model based on entropy weight and TOPSIS method. It is applied to the evaluation of software class diagram, and at the same time the difference between the entropy values as a weight, making full use of the information in the original data, to a certain extent reduces the subjectivity of weight determination; the evaluation results are in good agreement with the actual situation, indicating that the method is reasonable and feasible.

Data Availability

The data used to support the findings of this study were supplied by M. Genero under license. M. Genero at the Department of Computer Science at the University of Castilla-La Mancha, Ciduad Real, Spain, has allowed the author to quote the twenty-seven UML class diagrams related to bank information systems and the corresponding metric values. Reference: M. Genero. Defining and validating metrics for conceptual models [D], University of Castilla-La Mancha, 2002.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Table 4: The correlation analysis of the complexity measurement results.

Metric methods	Understandability	Analysability	Maintainability	Average
Zhou metric	0.741∗∗	0.773∗∗	0.775∗∗	0.763
Yi15 metric	0.945∗∗	0.928∗∗	0.957∗∗	0.949667
Yi18 metric	0.959∗∗	0.948∗∗	0.971∗∗	0.959333
Z_i metric	0.959∗∗	0.956∗∗	0.962∗∗	0.959

Note: ∗∗ indicates significant correlations at the 0.01 level in bilateral test.

Acknowledgments

This research has been supported by the Science and Technology Foundation of Jiangxi Provincial Department of Education (Project Name: Research on Software Complexity Measurement Based on Multiple Attribute Decision Making).

References

[1] X. Zhou, X.-K. Chen, J.-S. Sun, and F.-Q. Yang, "Software measurement based reusable component extraction in object-oriented system," *Acta Electronica Sinica*, vol. 31, no. 5, pp. 649–653, 2003.

[2] R. J. Rubey and R. D. Hartwick, "Quantitative measurement of program quality," *ACM National Computer Conference*, vol. 23, pp. 671–677, 1968.

[3] B. Hardekopf and C. Lin, "The ant and the grasshopper: Fast and accurate pointer analysis for millions of lines of code," *ACM SIGPLAN Notices*, vol. 42, no. 6, pp. 290–299, 2007.

[4] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976.

[5] N. Choursiya and R. Yadav, "An enhanced function point analysis (FPA) method for software size estimation," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 2797–2799, 2015.

[6] N. Padhy, S. Satapathy, and R. P. Singh, "Utility of an object oriented reusability metrics and estimation complexity," *Indian Journal of Science and Technology*, vol. 10, no. 3, pp. 1–9, 2017.

[7] S. Misra, A. Adewumi, L. Fernandez-Sanz, and R. Damasevicius, "A suite of object oriented cognitive complexity metrics," *IEEE Access*, vol. 6, pp. 8782–8796, 2018.

[8] D. L. Gupta and K. Saxena, "Software bug prediction using object-oriented metrics," *Sādhanā*, vol. 42, no. 5, pp. 655–669, 2017.

[9] J. Wang and Q. Wang, "Analyzing and predicting software integration bugs using network analysis on requirements dependency network," *Requirements Engineering*, vol. 21, no. 2, pp. 161–184, 2016.

[10] A. Gosain and G. Sharma, "A dynamic size measure for object oriented software," *International Journal of Systems Assurance Engineering and Management*, vol. 8, pp. 1209–1221, 2017.

[11] X. Hu, J. Zuo, and K. Wang, "Study on AHP-based quantification of software quality," *Computer Application and Software*, vol. 30, no. 11, pp. 138–141, 2013.

[12] M. Marchesi, "OOA metrics for the Unified Modeling Language," in *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering, CSMR 1998*, pp. 67–73, Italy, March 1998.

[13] M. Genero, M. Piattini, and M. Chaudron, "Quality of UML models," *Information and Software Technology*, vol. 51, no. 12, pp. 1629-1630, 2009.

[14] Y. Zhang, J. Tao, and L. Qian, "A metrics suite for class complexity based-on UML," *Computer Science*, vol. 29, no. 10, pp. 128–132, 2002.

[15] P. In, S. Kim, and M. Barry, "UML-based object-oriented metrics for architecture complexity analysis," in *Proceedings of the Ground System Architectures Workshop the Aerospace Corporation*, March 2003.

[16] H. Lu, Y. Zhou, B. Xu, H. Leung, and L. Chen, "The ability of object-oriented metrics to predict change-proneness: A meta-analysis," *Empirical Software Engineering*, vol. 17, no. 3, pp. 200–242, 2012.

[17] T. Yi, "On the application of information entropy-based multi-attribute decision in UML class diagram metrics," *International JoUrnal of u- and e-Service, Science and Technology*, vol. 8, no. 6, pp. 105–116, 2015.

[18] T. Yi and C. Fang, "A complexity metric for object-oriented software," *International Journal of Computers and Applications*, pp. 1–6, 2018.

[19] F. Geng and X. Ruan, "Campus network information security risk assessment based on FAHP and matter element model," in *Intelligent Computing Methodologies*, vol. 10363, pp. 298–306, 2017.

[20] Y. Hu, "Comprehensive evaluation of multi index panel data based on fuzzy matter element analysis," *Statistics & Decision*, no. 14, pp. 32–35, 2016.

[21] H. Jiang, Q. Zhang, and J. Peng, "An improved cloud matter element model based wind farm power quality evaluation," *Power System Technology*, vol. 38, no. 1, pp. 205–210, 2014.

[22] W. J. You, Z. S. Xu, and D. L. Liu, "On the fire risk assessment for the ancient buildings based on the matter element analysis," *Journal of Safety & Environment*, vol. 17, no. 3, pp. 873–878, 2017.

[23] Q. Pang, H. Wang, and Z. Xu, "Probabilistic linguistic term sets in multi-attribute group decision making," *Information Sciences*, vol. 369, pp. 128–143, 2016.

[24] R. N. Sun, B. Zhang, and T. T. Liu, "Service ranking method based on improved entropy TOPSIS," *Journal of Chinese Computer Systems*, vol. 38, no. 6, pp. 1221–1226, 2017.