Effectiveness of Rank-Order Weighting Methods in AspectJ Software Evolvability Measuring

Khine Zar Ne Win, Khin Mar Myo
University of Computer Studies, Mandalay
khinezarniewinn@gmail.com, kmmyo.ag@gmail.com

Abstract

The rapid growth of aspect-oriented applications increases the need to evaluate these applications. In the past, the software attributes have been tried to subjectively and objectively evaluate for the object-oriented applications. However, weighting the aspect-oriented software attributes is only subjective, depending mostly on experts' judgments. Among the weighting techniques, the Rank-Order Weighting Method is one of the less subjective techniques to evaluate the attributes weighting approaches. In this study, this approach is applied to assigning the weights of AspectJ's software attributes. A simulation was conducted to compare the target values and based values by using different rank-order weighting methods with the matching results by using Self Organizing Map (SOM). The evolvability results getting from experimenting with the two AspectJ software projects are discussed in this paper. According to the results, it is obvious that the rank ordered weighting method can be effectively applied in the area of software quality measuring area.

1. Introduction

Aspect-oriented programming (AOP) defines a new program construct, called an aspect, which is used to capture cross-cutting aspects of a software application in separate program entities. It comes into picture because of some limitations of Object-Oriented Programming (OOP). In OOP, there are parts of a system that cannot be viewed as being the responsibility of only one class, they cross-cut the complete system and affect parts of many classes. This problem can be solved by using Aspect-Oriented Approach of Software Development. In absence of aspect-oriented programming techniques, crosscutting concerns lead to redundant code fragments throughout the software system. Among the aspect-oriented programming languages, AspectJ is the most well-known and is a superset of Java (i.e., all AspectJ programs compile to Java byte code).

To keep on developing software and acceptance of AOP, software quality measuring is the important factor for accessing the competitive position of any software company. In this view, software metrics are needed to do such assessments. In order to propose the measurement of software quality, we need to consider the base attributes that are the most important for stimulating developer awareness. Moreover, the weights of attributes cannot be the same. To assign the weights of the attributes, we need to consider which technique is used; subjectively or objectively. However, there is no explicit description about how the attributes are weighted, only it is implied that they have assigned weights to the attributes according to their experiences from the past projects, as well as using questionnaires to collect students' view point, but, finally the experts decided for the weighting. It seems that the evaluator should be experienced or there should be a domain expert in evaluation team. So, these are the problems one faces when using and depending just on expert's judgments for weights. Thus, the main objective of this research is to propose a two-step weighting approach as a less subjective weighting approach to overcome this problem.

The rest of the article is structured as follows: AspectJ programming language is discussed in section 2. And then, measuring the evolution of software and its sub characteristics are depicted in section 3 and the attribute selection are in section 4. The proposed two-step weighting approach is described in section 5. This is followed by simulation study in section 6 and then results and discussion in section 7. Finally, conclusion is offered.

2. Aspect-oriented Programming with AspectJ

AspectJ is an aspect-oriented programming (AOP) language extension created at PARC (Palo Alto Research Center Incorporated), formerly XeroxPARC,

for the Java programming language. It is available in Eclipse Foundation open-source projects, both stand-alone and integrated into Eclipse. AspectJ has become a widely used de facto standard for AOP by emphasizing simplicity and usability for end users. It uses Java-like syntax, and included IDE integrations for displaying crosscutting structure since its initial public release [1].

In AOP, points in the code where we would want to run our code (instead of writing it in the class itself) are called join points. A join point can be almost any line of code - a method or constructor call, a field get or set, object initialization, or exception handler execution. AspectJ intercepts program execution flow at these join points and runs the code we want to run instead, called advice, which can happen before, after, or instead of the join point. An aspect describes these join points as pointcuts, which can describe one or more join points. A pointcut is a set of related joint points. When program execution reaches one of the join points in the pointcut, then the aspect code will run [2], [16].

3. Measuring Evolvability and Its Sub Characteristics

In this research, we focus on measuring the evolvable quality of aspect-oriented software implemented by AspectJ. Software evolvability is a many-sided quality attribute. So, it can be accessed by using many sub characteristics as follows. After the evolution of software components, we must find out that how much new evolved software is extensible, how much design patterns is stable compared old version to new version. Moreover, we must also find that which amount of change has been made in new version and how much new evolved system is sustainable to achieve the long-living software systems as a cost-effective way.

Therefore, we considered that the sub characteristics of evolvable software quality are integrated with extensibility, changeability, design stability and sustainability qualities measurements illustrated in Figure 1. These measurements: Extensibility Quality Measurement (EQM), Changeability Quality Measurement (CQM), Design Stability Measurement (DSQM) and Sustainability Quality Measurement (SQM) have been proposed and validated in [13], [14] and [15]. In this system, developers can analyze and measure the corresponding amount of the internal metrics as extensible, changeable, design stable and sustainable qualities of the software by using the proposed four metrics and then can measure the external factor, evolvability. These metrics can assess the corresponding amount of quality for the given software within a short time.

In this paper, we propose the effectiveness of weight assignments by using the rank-ordered weighting methods into the based attributes to measure the quality of software evolvability and its characteristics: extensibility, changeability, design stability and sustainability.

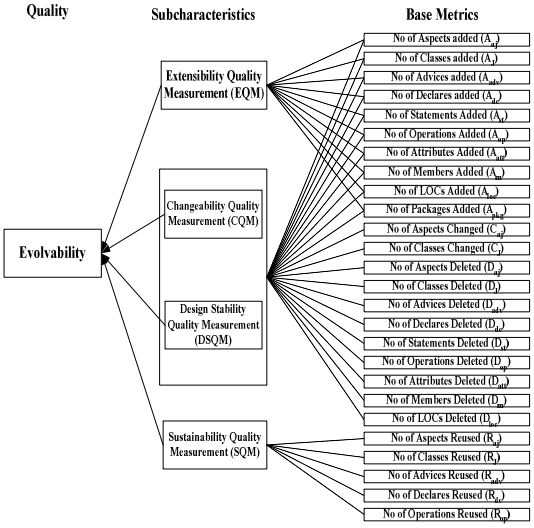

Figure 1. The Quality Model

4. Attribute Selection

To evaluate the qualities of the aspect-oriented software more accurately, the related attributes for each of qualities are needed to select. The following attributes are selected according to many guidelines and aspect-oriented features. According to guidelines [7], [8], [9], [10], [16] and [17] the selected attributes are identified as having more positively impact on the sub characteristics to measure the evolvable AspectJ projects. In this study, the following 26 attributes have been selected for proposed metrics:

1) No. of Aspect added (A_{aj})
2) No. of Class added (A_J)
3) No. of Advice added (A_{adv})
4) No. of Declare added (A_{dc})
5) No. of Statement Added (A_{st})
6) No. of Operation Added (A_{op})
7) No. of Attribute Added (A_{att})
8) No. of Member Added (A_m)
9) No. of Line of Code Added (A_{loc})
10) No. of Package Added (A_{pkg})
11) No. of Aspect Changed (C_{aj})

12) No. of Class Changed (C_J)
13) No. of Aspect Deleted (D_{aj})
14) No. of Class Deleted (D_J)
15) No. of Advice Deleted (D_{adv})
16) No. of Declare Deleted (D_{dc})
17) No. of Statement Deleted (D_{st})
18) No. of Operation Deleted (D_{op})
19) No. of Attribute Deleted (D_{att})
20) No. of Method Deleted (D_m)
21) No. of Line of Code Deleted (D_{loc})
22) No. of Aspects Reused (R_{aj})
23) No. of Classes Reused (R_j)
24) No. of Advices Reused (R_{adv})
25) No. of Declares Reused (R_{dc})
26) No. of Operations Reused (R_{op})

The above attributes are selected not only based on the guidelines but also based on the features of aspect-oriented software.

5. Proposed Two-Steps Weighting Approach

After selecting the internal (based) attributes to measure the quality metrics, another two important tasks, (i) Ranking and (ii) Weighting to internal attributes, are needed to perform and each of these techniques uses less subjective methods. To elicit weights for the attributes from their ranks, the proposed two-step weighting method is used. The following steps are involved:

i. Ranking the attributes in order from most important to least important
ii. Using one of the formulas among rank-order weighting techniques for assigning weights

(i) Ranking the Internal Attributes

When we rank to internal attributes, the most prominent attributes of aspect-oriented features (such as aspect, class, advice and intertype declarations) should assign the highest weights values.

The remaining internal attributes are similar with Object Oriented Feature (such as LOCs, statements, operations, attributes, packages and members). Therefore lowest weight values should be assigned to these internal attributes.

In the literatures, the evolvability measurement is defined in terms of atomic changes to the modules in a program. The main concept is to measure the quality when transforming source code edits into a set of atomic changes, which captures the semantic differences between two releases of a program. Zhang et al. [17] presented a catalog of atomic changes for AspectJ programs. Although the atomic changes to the module are adding, deleting and changing to the modules including in the source code, these changes are not ranking in this paper.

In section 6, it is detail explanation that how to assign the rank and weight values to the based attributes.

(ii) Weighting the Internal Attributes

The weights of based metrics are needed to assign because the important of individual attributes are not the same. To get the weight assignments, we can use alternative weighting methods such as direct weighting techniques, point allocation techniques, rank-ordered weighting techniques, etc. Among these techniques, rank-ordered weighting techniques(SMARTER) are used in order to overcome the subjectivity in assigning weights to the attributes of aspect-oriented applications.

This step is to calculate weights for the attributes using their ranks as determined from the earlier step. Calculation is done by using rank-order weighting formula and the weight of each attribute according to its rank can be expressed, [5], [11] and [12]. There are famous rank orders weighting formulas, which are Reciprocal of the Ranks (RR), Rank-sum (RS), Rank-Order Centroid (ROC) and Equal Weight (EW). In general, weight for the i^{th} most important attribute in each formula is following.

Rank Sum:

$$wt_i - \frac{K - r_i + 1}{\sum_{j=1}^{K} K - r_j + 1} \qquad (1)$$

where r_i is the rank of the i^{th} attribute and K is the total number of attributes

Rank Reciprocal:

$$wt_i - \frac{1/r_i}{\sum_{j=1}^{K}(\frac{1}{r_j})} \qquad (2)$$

where r_i is the rank of the i^{th} attribute and K is the total number of attributes

Rank Order Centroid:

$$wt_i = \left(\frac{1}{K}\right)\sum_{j=1}^{K}\left(\frac{1}{r_j}\right) \quad (3)$$

where r_i is the rank of the i^{th} attribute and K is the total number of attributes

Equal Weight

$$wt_i = \left(\frac{1}{K}\right) \quad (4)$$

where K is the total number of attributes

6. Simulation Study

The objective of simulation is to evaluate the effectiveness of Rank Order Weighting Method. The following steps are applied to approve the Rank Order Weighting Method is applicable for software quality measurement.

1. Accept the AspectJ project.
2. Extract the internal features from AspectJ project.
3. Assign the rank and weight value to each internal attribute.
4. Calculate the Evolvability and its characteristics.
5. Compare the SOM clustering results between the target values and the based attributes of Evolvability and its characteristics.

Firstly, the system accepts the AspectJ projects MobileMedia (MM) [3] and HealthWatcher (HW) [4] and then the values of the selected attributes are extracted from these projects as described in section 4.

In step 3, it is needed to assign the rank and weight value of each internal attribute in each proposed metric. Attributes' weights and ranks are calculated according to their total number of attributes of a metric by using alternative ranking methods shown in Table 1,2, and 3.

Table 1. Ranking attributes and weight assignments for SQM

Rank	Attributes	ROC	RR	RS	EW
1	R_{ai}	0.4567	0.4379	0.3333	0.2
2	R_J	0.2567	0.2189	0.2667	0.2
3	R_{adv}	0.1567	0.1459	0.2	0.2
4	R_d	0.09	0.1095	0.1333	0.2
5	R_{op}	0.04	0.0876	0.0667	0.2

Table 2. Ranking attributes and weight assignments for CQM and DSQM

Rank	Attributes	ROC	RR	RS	EW
1	A_{ai}	0.1641	0.2743	0.0909	0.0476
2	C_{ai}	0.126	0.1372	0.0866	0.0476
3	D_{aj}	0.1022	0.0914	0.0823	0.0476
4	A_J	0.0863	0.0686	0.0779	0.0476
5	C_J	0.0744	0.0549	0.0736	0.0476
6	D_J	0.0649	0.0457	0.0693	0.0476
7	A_{adv}	0.0569	0.0392	0.0649	0.0476
8	D_{adv}	0.0448	0.0343	0.0606	0.0476
9	A_d	0.0442	0.0305	0.0563	0.0476
10	D_d	0.0389	0.0274	0.0519	0.0476
11	A_s	0.0341	0.0249	0.0476	0.0476
12	D_s	0.0298	0.0229	0.0433	0.0476
13	A_{op}	0.0258	0.0211	0.039	0.0476
14	D_{op}	0.0222	0.0196	0.0346	0.0476
15	A_{att}	0.0188	0.0183	0.0303	0.0476
16	D_{att}	0.0156	0.0171	0.026	0.0476
17	A_m	0.0126	0.0161	0.0216	0.0476
18	D_m	0.0098	0.0152	0.0173	0.0476
19	A_{loc}	0.0072	0.0144	0.013	0.0476
20	D_{loc}	0.0047	0.0137	0.0087	0.0476
21	A_{pkg}	0.0023	0.0131	0.0043	0.0476

Table 3. Ranking attributes and weight assignments for EQM

Rank	Attributes	ROC	RR	RS	EW
1	A_{aj}	0.2929	0.3414	0.1818	0.1
2	A_J	0.1929	0.1707	0.1636	0.1
3	A_{adv}	0.1429	0.1024	0.1455	0.1
4	A_d	0.1096	0.0854	0.1273	0.1
5	A_s	0.0846	0.0683	0.1091	0.1
6	A_{op}	0.0646	0.0569	0.0909	0.1
7	A_{att}	0.0479	0.0488	0.0727	0.1
8	A_m	0.0336	0.0427	0.0545	0.1
9	A_{loc}	0.0211	0.0379	0.0364	0.1
10	A_{pkg}	0.01	0.0341	0.0182	0.1

Step 4 is to measure the evolvability. Evolvability is a multifaceted quality attribute, so we have been measured the sub characteristics: EQM, CQM, DSQM and CQM that are primary importance for an evolvable software system and each metric has been validated by using SOM (published in [13], [14] and [15]). The evolvability from old version to new version has been calculated by using equation 5.

Evolvability = (EQM * 0.25) + (CQM * 0.25) + (DSQM *0.25) + (SQM * 0.25) (5)

After calculating the evolvability and its sub characteristics, it is needed to validate how

much the attribute values of each quality factor have true weights. So, we use the SOM to cluster the results between the target values and the based attributes in step 5. If the matching result between the two clusters is the highest result, the rank and weight assignment to the based attributes will be true and the proposed metric will be robustness. Consequently, the evaluation and validation results of evolvability are discussed in section 7.

7. Results and Discussions

In this section, we use the MobileMedia (MM) and HealthWatcher (HW) as case study projects. Figure 2 to figure 10 show the analysis of target values of extensibility quality, changeable quality, design stability, sustainability and evolvability of MobileMedia projects' versions combination from version 1 to 7 (${}^{n}C_{r} = {}^{7}C_{2} = 21$ pairs) and HealthWatcher projects' versions combination from version 1 to 10 (${}^{n}C_{r} = {}^{10}C_{2} = 45$ pairs). These figures show that the results of how much new evolved system are extensible and sustainable, and how much design pattern is stable and then how many changes are there. These characteristics can be known by comparing the results between the two versions such as V [1, 2] and V [1, 3], etc.

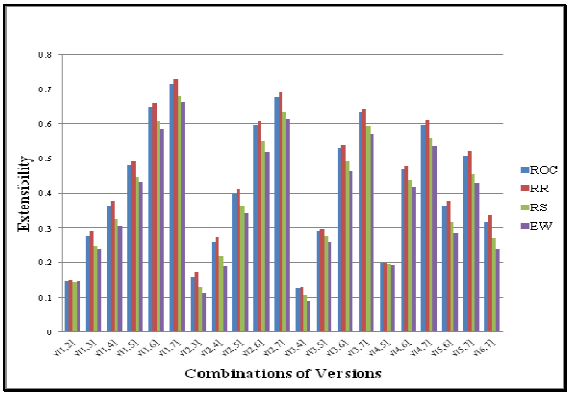

Figure 2. Analyzing the Extensibility by using Rank-Order Weighting Methods(MM)

In figure 2 and 4, the distinct part is that the target values measured by using ROC and RR methods are higher than the values by using two other methods. Like MobileMedia, the values by using these two methods (ROC and RR) are also the highest values in the experiment of HealthWatcher Projects' Extensibility and Changeability illustrated in figure 3 and 5.

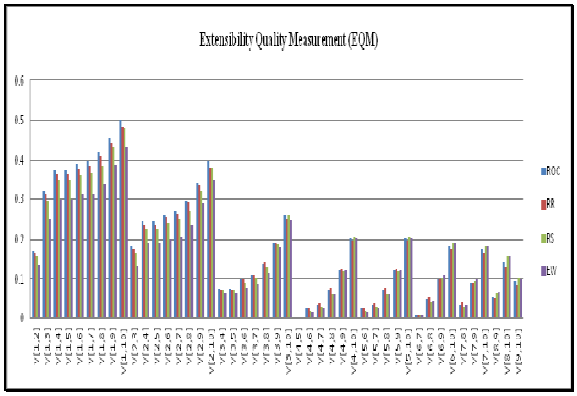

Figure 3. Analyzing the Extensibility by using Rank-Order Weighting Methods(HW)

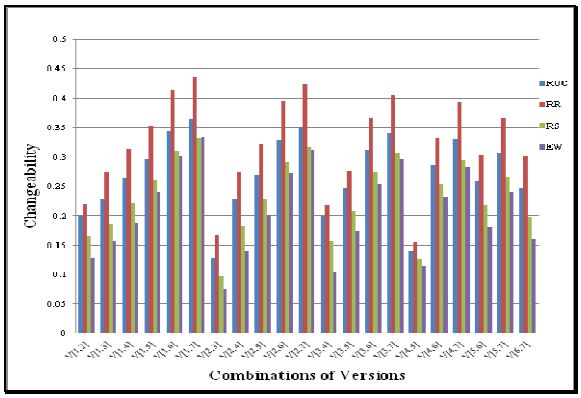

Figure 4. Analyzing the Changeability by using Rank-Order Weighting Methods(MM)

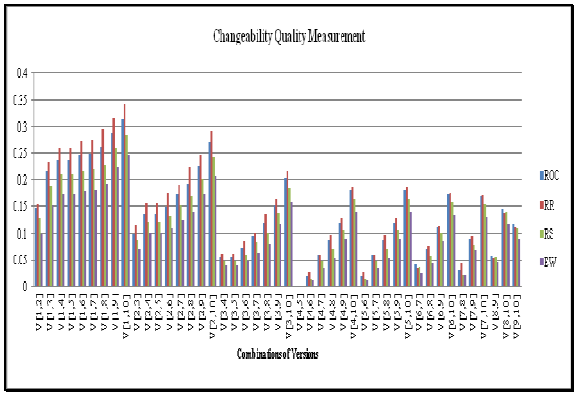

Figure 5. Analyzing the Changeability by using Rank-Order Weighting Methods(HW)

In figure 6 and 8, the target values of SQM and DSQM that are measured by using EW methods are higher than the values by using other methods. In figure 7 and 9, the highest values of HealthWatcher Projects' qualities are also the same as MobileMedia.

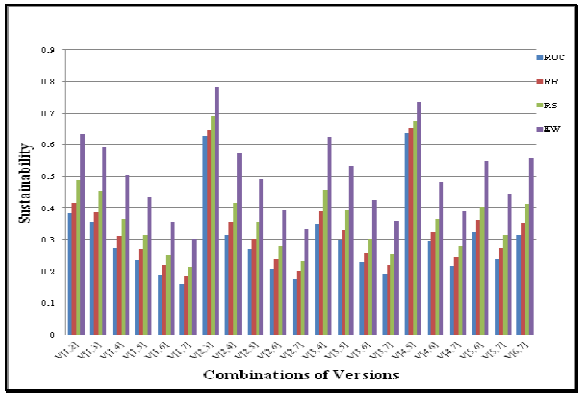

Figure 6. Analyzing the Sustainability by using Rank-Order Weighting Methods(MM)

In figure 10, the Evolvability values using the EW method are the highest when comparing these values with other values. Like MobileMedia, HealthWatcher's Evolvability has the highest value that is measured by using EW technique illustrated in figure 11.

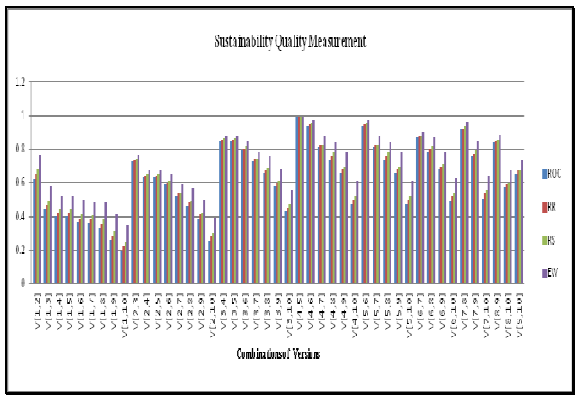

Figure 7. Analyzing the Sustainability by using Rank-Order Weighting Methods(HW)

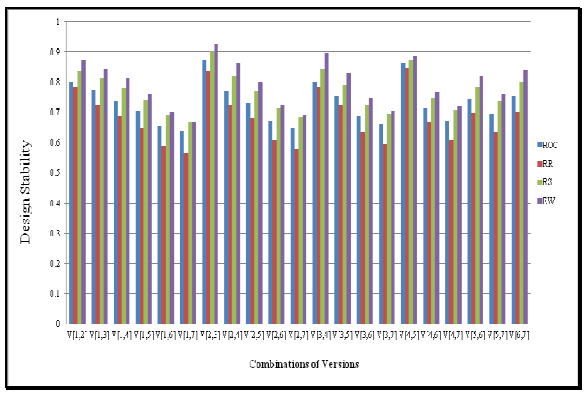

Figure 8. Analyzing the Design Stability by using Rank-Order Weighting Methods(MM)

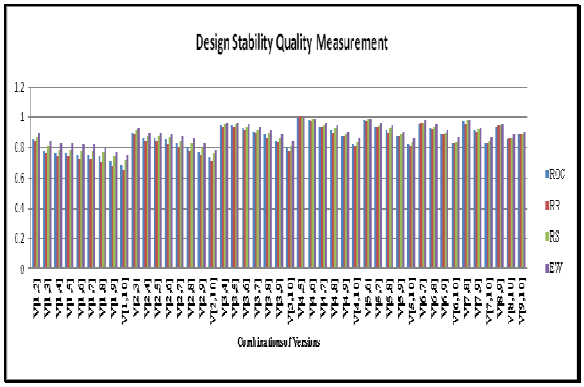

Figure 9. Analyzing the Design Stability by using Rank-Order Weighting Methods(HW)

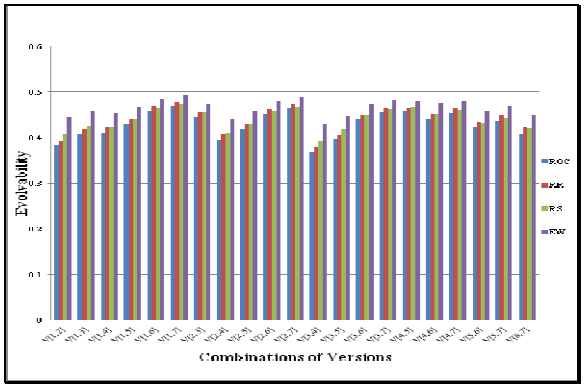

Figure 10. Analyzing the Evolvability Measurement by using Rank-Order Weighting Methods(MM)

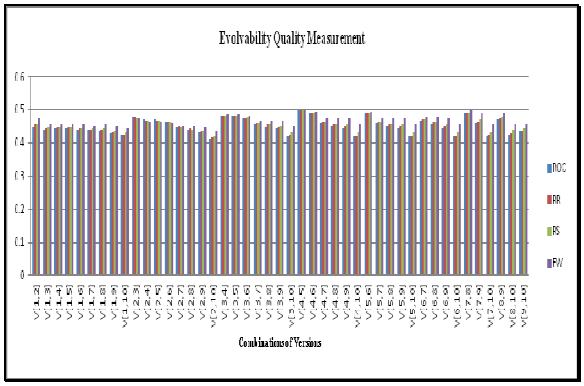

Figure 11. Analyzing the Evolvability Measurement by using Rank-Order Weighting Methods(HW)

After evaluating the proposed metrics, we need to check the validity to achieve the reliability and robustness of each metric. Proposed metrics have been evaluated and validated both theoretically and empirically using extension of Briand's OO Metrics Framework and Self-Organizing Map (SOM) in [11], [12] and [13].

The empirical results (by using SOM) of evolvability and its characteristics in MobileMedia and HealthWatcher projects are shown in Table 4 and Table 5.

Table 4. Analytical results of Evolvability and its characteristics for MobileMedia

Alternative Weights	EQM	CQM	DSQM	SQM	Evolvability
ROC	95%	91%	91%	91%	82%
RR	91%	81%	81%	91%	76%
RS	95%	91%	91%	91%	80%
EW	91%	91%	95%	91%	71%

Table 5. Analytical results of Evolvability and its characteristics for HealthWatcher

Alternative Weights	EQM	CQM	DSQM	SQM	Evolvability
ROC	87%	76%	76%	93%	81%
RR	82%	71%	73%	93%	91%
RS	82%	76%	78%	93%	86%
EW	71%	89%	89%	93%	71%

Before validating the metrics, the extensibility and changeability of AspectJ projects have the highest values by using the ROC and RR techniques. Moreover, the design stability and sustainability have the highest values measured by using EW methods.

After validating by using SOM, the target quality measurements and the based metrics by using these rank-order weighting techniques also have the highest matching results: the extensibility and changeability by using the ROC and RR techniques and , the design stability and sustainability by using EW methods.

According to experimental results by using the Self-Organizing Map Algorithm, we can see that the proposed metrics are validated and useful for software development processes and we can say that the base attributes have true weights.

8. Conclusion

For improving the quality of aspect-oriented software applications, Extensibility, Changeability, Design Stability and Sustainability metrics have been integrated as the sub characteristics of evolvability. And then, the proposed metrics have been formulated with the selected attributes according to the guidelines and their weights have been calculated by using Rank-Order Weighting Methods. Moreover, it has been validated by using the Self-Organizing Map (SOM) whether the rankings and weighting assignments of the base attributes are consistent with the true weights of each attribute. To ensure repeatability of the experiments, research results have been validated by means of a series of case studies empirically and theoretically. These quality measuring metrics can be used as a self-assessment tool for software developers. So, we can say that rank-order weighting techniques have robustness and effectiveness on measuring the AspectJ project evolution.

References

[1] http://en.wikipedia.org/wiki/AspectJ

[2] http://java.dzone.com/articles/aop-and-aspectj-terminology, © 1997-2014, DZone, Inc.

[3] MobileMedia,http://mobilemedia.cvs.sourceforge.net/viewvc/mobilemedia/

[4] HealthWatcher,http://www.comp.lancs.ac.uk/~greenwop/tao/

[5] B. S. Ahn, K. S. Park, “Comparing methods for multiattribute decision making with ordinal weights”, Computers & Operations Research 35 (2008), www.sciencedirect.com

[6] H.P. Breivold, I. Crnkovic and P.J. Eriksson, “Analyzing Software Evolvability”, Computer Software and Applications, (2008). COMPSAC '08. 32nd Annual IEEE International Conference

[7] Z. Durdik, B. Klatt, H.Koziolek, K. Krogmann, J. Stammel, R. Weiss, “Sustainability Guidelines for Long-Living Software Systems”

[8] H. Koziolek, “Sustainability Evaluation of Software Architectures: A Systematic Review”, from ACM Digital library

[9] K. Mguni and Y.Ayalew, “An Assessment of Maintainability of an Aspect-Oriented System”, ISRN Software Engineering Volume 2013, http://dx.doi.org/10.1155/2013/121692

[10] A. Przybylek, “Analysis of the impact of aspect-oriented programming on source code quality”, GDANSK UNIVERSITY OF TECHNOLOGY, Faculty of Electronics, Telecommunications and Informatics,(2011)

[11] M. Saeid, A. Ghani, and H. Selamat, “Rank-Order Weighting of Web Attributes for Website Evaluation”, the International Arab Journal of Information Technology, Vol. 8, No. 1, January(2011)

[12] T.Z. Thaw, “Measuring and Evaluation of Reusable Quality and Extensible Quality for XML Schema Documents”, IEEE, (2011), Cyberjaya, Selangor, Malaysia.

[13] K.Z.N. Win, “Quantifying and Validation of Changeability and Extensibility for Aspect-Oriented Software”, ICAET'2014, Singapore, March 29-30, (2014)

[14] K.Z.N. Win, “Measuring and Evaluating Sustainability and Design Stability Software Qualities for Long-Living Aspect-Oriented Applications”, 10th International Conference on "ASEAN Community Knowledge Networks for the Economy, Society, Culture, and Environmental Stability", Mandalay Hill Resort Hotel, Republic of the Union of Myanmar,8-12 May (2014)

[15] K.Z.N. Win, “Quantifying and Evaluating of Evolvability Software Quality for Aspect-Oriented Software”, ICEEHE, 2013, published in Transactions on GIGAKU, Volume2, No 2, and October (2014)

[16] J. Zhao. “Change impact analysis for aspect-oriented software evolution”, In Proc. 5th International Workshop on Principles of Software Evolution, pages 108–112, May (2002).

[17] Zhang et al., “Change impact analysis for AspectJ programs”, 24th IEEE International Conference on Software Maintenance, Beijing, China (2008)