Quantifying and Validation of Changeability and Extensibility for Aspect-Oriented Software

Khine Zar Ne Winn

***Abstract*—**Today, real-world software needs evolution after a certain period of time to meet the current trends, technology, changes in user requirement and operational environment. During system evolution, software change is an essential operation. After the evolution of system, we must find out that how much new evolved system is extensible when the software functionalities are added or removed during maintenance, or when the existing programs are modified to reuse. Therefore, we need to have measurement mechanisms for evaluating software quality. In this study, we aim to measure the changeability and extensibility of aspect-oriented (AO) software implemented in AspectJ. The analysis is based on MobileMedia, AspectJ projects, by using Self-Organizing Map (SOM).

***Keywords*—**Changeability, extensibility, aspect-oriented (AO), AspectJ, MobileMedia, Self-Organizing Map (SOM).

I. 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. In Object-Oriented Programming, 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 [1].

Separation of concern is vital to quantitatively assess the quality of software produced using Aspect-Oriented Software Development (AOSD). In this view, software metrics are needed to do such assessment. The external quality attributes of aspect-oriented software are usually measured, using modeling techniques, as a function of metrics that quantify the internal structural properties of the aspect-oriented software.

However, there are few empirical studies [2]-[3]-[4]-[5] that have been conducted to relate software metrics with external quality attributes of aspect-oriented software in general, and metrics have not been evaluated as indicators of aspect extensible and changeable using SOM in particular.

Motivated by afore mentioned issues, we attempt to prove the presented metrics empirically with MobileMedia[17], AspectJ projects. MobileMedia is a Software Product Line (SPL) that manipulates photo, music, and video on mobile devices. This is a joint project coordinated by Lancaster University and developed on top of MobilePhoto. The availability of adequate metrics for these assessments provides software managers early insight into trends in software evolution, and thus assists them in managing and controlling long-lived software systems.

The rest of this paper is organized as follows. Section II defines the Aspect-Oriented Programming. Section III describes for supporting tools. Section IV presents proposed system. Section V discusses the case study and Section VI reviews related works. Section VII concludes the paper.

Khine Zar Ne Winn is with the University of Computer Studies, Mandalay, Myanmar.

(Corresponding author's e-mail: khinezarnewinn1@gmail.com).

II. Aspect-Oriented Programming

The goal of aspect-oriented programming is to provide an advanced modularization scheme to separate the core functionality of a software system from system-wide concerns that cut across the implementation of this core functionality. To this extent, AOP introduces a new abstraction mechanism, called an aspect. An aspect is a special kind of module that represents a crosscutting concern. Aspects are defined independently from the core functionality of the system and integrated with that base program by means of a dedicated aspect weaver, a dedicated tool similar to a compiler that merges aspect code and base code in the appropriate way [6]. At runtime or compile time, depending on AOP framework, cross-cutting code is injected at the specified pointcuts. Beyond modularizing crosscutting concerns with the design and implementation perspective, AOP enables specific applications that improve the quality of software. Implementing software application using AOP improves software quality in many ways, like in term of better understanding, higher productivity and cost savings. Software developer can apply AOP approach for improving quality in design and implementation perspective as discussed by Ramnivas Laddad [7]. With AOP, users can create reusable aspects that implement a variety of contracts and provide guidance for the most positive qualities.

III. Tools Support

A. AJATO: an AspectJ Assessment Tool

AJATO [18] is to support the quantitative assessment of Aspect-oriented (AO) software artifacts. The goals of the tool are: (i) to compute existing AO metrics, and (ii) to support the

application of a heuristics suite. Most of the AO metrics available in this tool are based on traditional metrics, such as lines of code, and on extensions of those widely used with object-oriented design, such as the Chidamber and Kemerer metrics. Besides, the metrics suite also encompasses new metrics for measuring separation of concerns. Some of the metrics available in AJATO are Concern Diffusion over Components (CDC), Number of Attributes per Concern (NOA concern), Number of Operations per Concern (NOO concern), Vocabulary Size (VS), Number of Attributes (NOA), and Number of Operations (NOO). In addition to the metrics, this assessment tool also implements some heuristics rules in order to automate some modularity analysis about the numbers. It also supports the automatic generation of warnings when certain design principles, such as narrow interfaces and high cohesion, are violated. AJATO also supports the association of application-specific thresholds with metrics and heuristics [8].

B. WinMerge Tool

WinMerge[9] is an Open Source differencing and merging tool for Windows. Use it to detect changes between files and folders, and then to merge changes selectively. Users can use WinMerge by itself, or launch it from a version control system or another application. When the user compares two files or folders, the result is displayed in the WinMerge window. The WinMerge window's visual display and functions are designed to help the users see differences between their targets, and to merge selected differences if they wants to.

In a file comparison: WinMerge shows differences between lines. The built-in text editor provides syntax highlighting for several programming languages and other file formats.

In a folder comparison: WinMerge indicates various kinds of differences between the files that the folders contain [9].

IV. Proposed System

A. Proposed Quality Measurement Process

To demonstrate the proposed metrics, aspect-oriented software are analyzed and measured by using the above tools. The quality measuring process is presented in Fig.1.

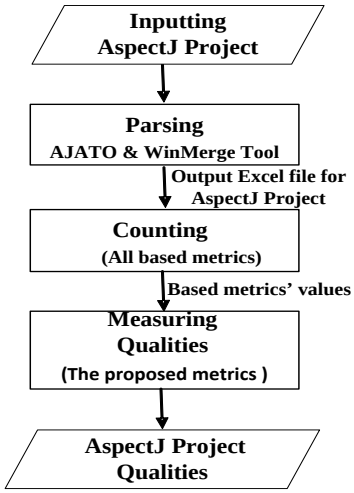

Fig. 1 Architecture of Proposed Quality Measuring System

B. Attribute Selections

To measure the qualities of the aspect-oriented software more accurately, the related attributes for each of qualities are needed to select. These attributes are selected according to many guide lines and aspect-oriented features. The proposed metrics are formulated based on the rank weighting method to measure the quality of extensibility and changeability. According to guidelines [11] and [19], the selected attributes are identified as having more positively impact on extensible and changeable qualities of AspectJ projects. In this study, the following 21 attributes are 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 Method 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})

C. Weight Assignments

The based attributes in MobileMedia projects (version 1 to 7) are counted and used by the proposed two metrics for measuring the quality of this project as the input based metrics. To quantify the proposed metrics, the weights of based metrics are needed to assign. Because the important of individual attributes are not the same, their weights are not equal. There are a variety of situations where it is reasonable to use ranked weights, and there have been various techniques developed to deal with ranked weights and arrive at a choice or rank alternatives under consideration. Therefore, rank-order weighting methods are used to assign weights of attributes. Among these methods, Rank-Order Centroid (ROC) method is suggested the best method in many literatures [12] and is defined as:

$$\mathrm{ROC_i}(\mu) = \frac{1}{t}\sum_{l=i}^{t}\frac{1}{p_l}, \quad i = 1,2,\ldots t \qquad (1)$$

where μ is a metric, p_l is the position of i^{th} attribute for μ metric and t is the total number of attributes.

Table I and Table II show the weight assignments of the internal software attributes for Changeability and Extensibility by using equation 1. To calculate the proposed metrics, these corresponding weights are multiplied with each of the base attributes as a preprocessing step.

TABLE I
RANKING ATTRIBUTES AND ASSIGN WEIGHTS FOR EQM

Rank	Attributes	Weight(ROC)
1	A_{aj}	0.2929
2	A_J	0.1929
3	A_{adv}	0.1429
4	A_d	0.1096
5	A_s	0.0846
6	A_{op}	0.0646
7	A_{att}	0.0479
8	A_m	0.0336
9	A_{loc}	0.0211
10	A_{pkg}	0.01

TABLE II
RANKING ATTRIBUTES AND ASSIGN WEIGHTS FOR CQM

Rank	Attributes	Weight(ROC)
1	A_{aj}	0.1641
2	C_{aj}	0.126
3	D_{aj}	0.1022
4	A_J	0.0863
5	C_J	0.0744
6	D_J	0.0649
7	A_{adv}	0.0569
8	D_{adv}	0.0448
9	A_d	0.0442
10	D_d	0.0389
11	A_s	0.0341
12	D_s	0.0298
13	A_{op}	0.0258
14	D_{op}	0.0222
15	A_{att}	0.0188
16	D_{att}	0.0156
17	A_m	0.0126
18	D_m	0.0098
19	A_{loc}	0.0072
20	D_{loc}	0.0047
21	A_{pkg}	0.0023

D.Changeability Quality Measurement (CQM)

As changeability grows in importance as a consideration in the engineering of systems, there is a critical need to have a more rigorous and quantified definition. Change can be defined as the transition over time of a system to an altered state. A change applies to a class, to a variable or to a method. In this study, the formula for CQM metric is:

$$\begin{aligned} CQM = {} & \frac{1}{T_{aj}}(A_{aj} + C_{aj} + D_{aj}) + \frac{1}{T_J}(A_J + C_J + D_J) \\ & + \frac{1}{T_{adv}}(A_{adv} + D_{adv}) + \frac{1}{T_{dc}}(A_{dc} + D_{dc}) \\ & + \frac{1}{T_{st}}(A_{st} + D_{st}) + \frac{1}{T_{op}}(A_{op} + D_{op}) \\ & + \frac{1}{T_{att}}(A_{att} + D_{att}) + \frac{1}{T_m}(A_m + D_m) \\ & + \frac{1}{T_{loc}}(A_{loc} + D_{loc}) + \frac{1}{T_{pkg}}(A_{pkg}) \end{aligned} \quad (2)$$

E. Extensibility Quality Measurement (EQM)

Aspect-oriented software development (AOSD) is likely to have positive effect on performance, modularity and evolution. So, it becomes important to reuse components. We focus on a specific kind of reusing of component called extensibility, i.e. the extension of software without accessing existing code to edit or copy it. Extensibility is a systemic measure of the ability to extend a software design principle where the implementation takes into consideration future growth [10]. So, the formula for EQM is:

$$\begin{aligned} EQM = {} & \frac{1}{T_{aj}}(A_{aj}) + \frac{1}{T_J}(A_J) + \frac{1}{T_{adv}}(A_{adv}) + \frac{1}{T_{dc}}(A_{dc}) \\ & + \frac{1}{T_{st}}(A_{st}) + \frac{1}{T_{op}}(A_{op}) + \frac{1}{T_{att}}(A_{att}) \\ & + \frac{1}{T_m}(A_m) + \frac{1}{T_{loc}}(A_{loc}) \\ & + \frac{1}{T_{pkg}}(A_{pkg}) \end{aligned} \quad (3)$$

V. CASE STUDY

In this paper, we use the MobileMedia as a case study. In the comparison of two versions, all of the attributes used in quantifying the result are normalizing by total number of each attribute of old version (T_{aj},T_J,T_{adv},T_{dc},T_{st},T_{op},T_{att},T_m,T_{loc},T_{pkg}). Fig.2 and Fig.3 show the progress of extensibility quality and changeable quality of the projects, combinations of version 1 to 7.These figures show the results of how much new evolved system is extensible and how many changes are there.

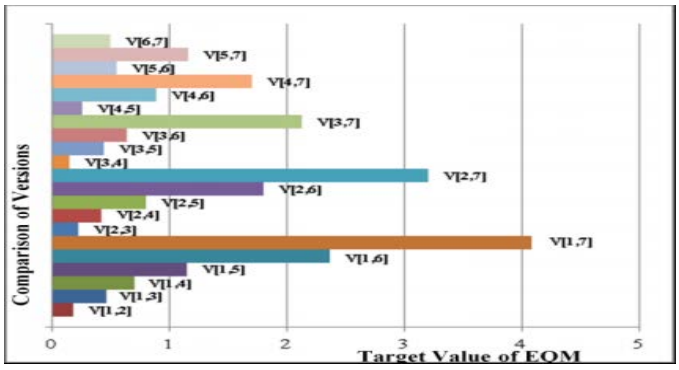

Fig. 2 Target Values for Extensible Quality Measurement (EQM)

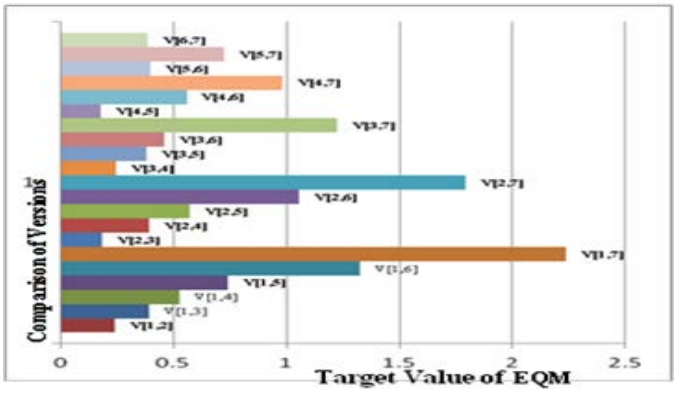

Fig. 3 Target Values for Changeability Quality Measurement (CQM)

A. Empirical Validation Using Self-Organizing Map (SOM)

The unsupervised learning method, SOM is suitable for measuring the validity and usefulness of the proposed metrics that they are difficult to prove their validity and usefulness because they don't have historical data and other comparison metric. One advantage is that they do not require more understanding with input data and they can cluster and estimate incomplete historical data. The SOM learns similarity of input feature vectors and responds groups of similar input vectors [13].

Before clustering MobileMedia projects, two data files are created for each of proposed metrics, one contains the particular proposed metric values of the combination of number of versions ${}^{n}C_{r} = {}^{7}C_{2} = 21$ pairs and another contains its based attributes of these versions' comparisons. In preprocessing step, file attributes are converted into binary data according to equation 4:

$$f(x) = \begin{cases} 1, & x < \frac{f(y)}{2} \\ 0, & otherwise \end{cases} \quad (4)$$

where f(x) is the binary conversion function that converts the x based metric value into binary data and f(y) is the total number of each attributes of the latest version of MobileMedia, AspectJ project.

. After converting binary data, the contingency tables are created for each of versions' pairs. To create the contingency table, dissimilarities are calculated by Jaccard coefficient [18]:

$$d(x,y) = \frac{p}{p+q+r} \quad (5)$$

p=number of times x_i=1 and y_i=1
q=number of times x_i=1 and y_i=0
r=number of times x_i=0 and y_i=1

These dissimilarity matrixes of versions are trained with the SOM and each version are clustered in terms of dissimilarity matrixes. For instance, the two versions of EQM and CQM metrics are clustered by SOM and the cluster matching result is analyzed.

To prove the validity and usefulness of the proposed metrics, the clustering method, SOM and well-known AspectJ projects are used. MobileMedia project's versions are clustered by SOM depending on two types of input: only particular proposed metric and its based attributes. The cluster results getting from two inputs are compared to match or not. According to our experiment, most of the clustering results are matched. Therefore, it is obvious that the proposed metrics are validated and useful. The clustering results are shown in the following Fig. 4, 5, 6 and 7.

In Fig. 4, SOM clusters MobileMedia project' versions into two groups depending on only EQM values. The first group contains version: V[1,7] and the second group contains other versions. The version contained in first group are the highest value of EQM.So,the other group contains the versions that have the other values of extensibility.Moreover, in Fig. 5, SOM also clusters two groups depending on its based attributes. The first group contains version: V[1,7], V[2,7] and V[3,7] and the other group contains other versions' pairs. The versions contained in first group are the highest values of attributes of extensibility Hence,the other group contains the versions that have the other values of internal attributes.The first groups' members of two types are then compared and analyzed. It is clear that 1 are matched. Moreover, the second groups' members of two types are also compared and analyzed. All of the result shows that 19 are matched. Therefore the total cluster matching result is 91% for proposed metric EQM.

Fig. 6 and Fig. 7 show the clustering results of Changeability Metric, CQM. The first group contains version: V[1,7] which is the highest value of CQM and the other group contains other pairs of versions. Furthermore, the second groups' members of two types are also compared and analyzed. The result shows that 19 are matched. So, the total cluster matching result for CQM is also almost 91%.

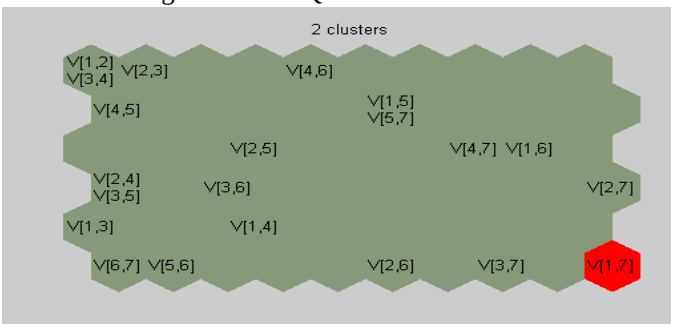

Fig. 4 Proposed Metrics clustering for Extensibility Metric

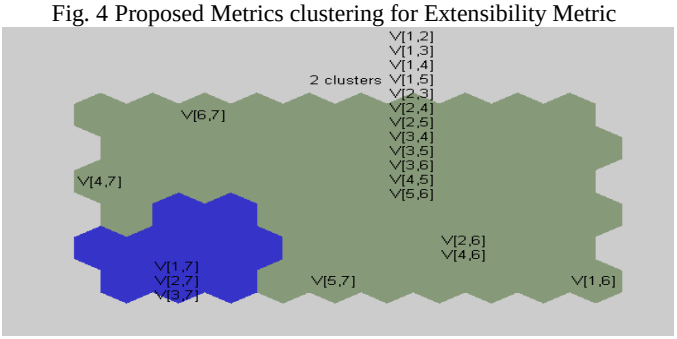

Fig. 5 Based Metrics clustering for Extensibility Metric

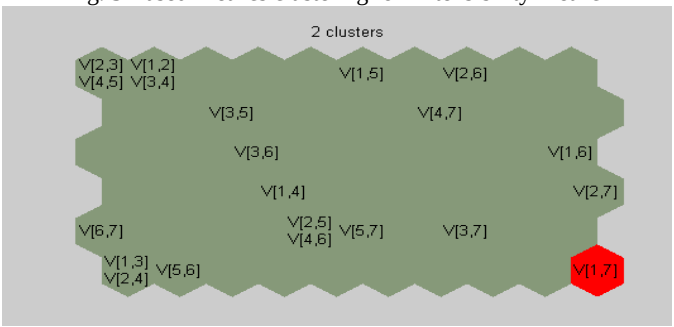

Fig. 6 Proposed Metrics clustering for Changeability Metric

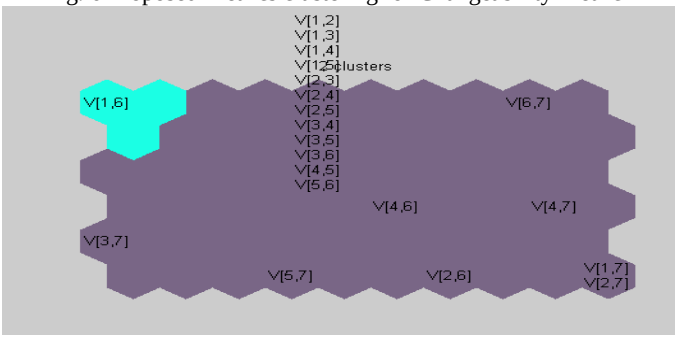

Fig. 7 Base Metrics clustering for Changeability Metric

VI. Related Works

Nowadays, one of the important aspects of research in the field of Software Engineering is the "Quantification of Parameters Affecting the Software Quality". Various characteristics and sub characteristics affecting the software quality have been quantified by using metrics to evaluate the software quality.

O. Lamouchi, R. Cherif and N.Lévyalso [14] attempted to quantify the software quality factors by subdividing the factors into criteria and sub criteria and by quantifying the metrics that are affecting them. They have elucidated their approach clearly by showing an example of quantifying portability.

R.Dadhich and B.Mathur [15] have also considered measuring reliability of an aspect-oriented software using fuzzy logic approach. B. Mathur, S.K. Nath [16] has been made to quantifying the portability of aspect oriented software using ISO/IEC 9126 Model. T.Z.Thaw [12] proposed measuring the qualities of XML schema documents and validating the proposed metrics by using SOM.

P.Kumar [10] is to propose a new quality model by integrating extensibility, sustainability, design stability and configurability as sub-characteristics under evolvability characteristic into AOSQUAMO Model, proposed a new quality model is called Aspect-Oriented Software Quality (AOSQ) Model which is derived from ISO/IEC 9126 quality.

VII. Conclusion and Future Works

In this paper, we have measured the extensible and changeable qualities of MobileMedia applications. According to our experiments, we can say that how many changes and how much extensibility are in AspectJ Projects. For improving aspect-oriented software applications, these measurements help software developers to evaluate the qualities of aspect-oriented software as a self-assessment tool and to save time and money of software development process.

Further extension for this research is to create metrics for measuring other qualities of aspect-oriented software such as sustainability, configurability, and design stability and so on. Moreover, the metrics will need to propose for measuring and improving evolvability of aspect-oriented programming. Therefore, aspect-oriented applications will be robust and widely used for business applications and other web-based information systems.

Acknowledgment

I would like to express my gratitude and my sincere thanks to my supervisor Dr. Khin Mar Myo, Associate Professor, of University of Computer Studies, Mandalay, Myanmar. I greatly appreciate the help I have received from her. Her excellent help and creative ideas have assisted me in broadening my research skills. I am extremely fortunate to work under her supervision of my research.

References

[1] M.Voelter, voelter at acm dot org , " Aspect oriented Programming in Java"

[2] R. Burrows, F. Ferrari, A.Garcia, and F. Taïani, "An Empirical Evaluation of Coupling Metrics on Aspect Oriented Programs," in ICSE Workshop on Emerging Trends in Software Metrics, pp. 53-58, 2010

[3] C. Sant'Anna, A. Garcia, C.Chavez, C. Lucena, and A. Staa, "On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework," in Brazilian Symposium on Software Engineering, 2003

[4] H. Shen, S. Zhang, and J.Zhao, "An Empirical Study of Maintainability in Aspect-Oriented System Evolution Using Coupling Metrics," in 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, pp. 233-236, 2008.
http://dx.doi.org/10.1109/TASE.2008.17

[5] P. Tonella and M. Ceccato, "Refactoring the aspectizable interfaces: an empirical assessment," IEEE Transactions on Software Engineering, vol. 31, pp. 819-832, 2005
http://dx.doi.org/10.1109/TSE.2005.115

[6] K. Mens and T. Tourw _ "Evolution Issues in Aspect-Oriented Programming"

[7] R. Laddad, "Aspect Oriemted Programming will improve Quality",2003,published by IEEE Computer Society 0740-745.

[8] E. Figueiredo, A. Garcia, C. Lucena, "AJATO: an AspectJ Assessment Tool"

[9] "WinMerge Tool", http://tour.winmerge.org/

[10] P.Kumar, "Aspect-Oriented Software Quality Model: The AOSQ Model", Advanced Computing: An International Journal (ACIJ), Vol.3, No.2, March 2012,

[11] J. Zhao, "Change Impact Analysis for Aspect-Oriented Software Evolution"

[12] T.Z.Thaw, "Measuring Qualities of XML Schema Documents"

[13] Z. He1, F. Shu, Y. Yang, W. Zhang, and Q. Wang, "*Data Unpredictability in Software Defect-Fixing Effort Prediction*" 2010 10th International Conference on Quality Software.

[14] O. Lamouchi, A.R. Cherif, and N. Lévy, — "A framework based measurements for evaluating an IS quality", Proceedings of the fifth on Asia-Pacific conference on conceptual modelling, Wollongong, NSW, Australia, January, 2008, pp.39-47.

[15] R. Dadhich, B. Mathur "Measuring Reliability of an Aspect Oriented Software Using Fuzzy Logic Approach."

[16] B. Mathur, S. K. Nath "Quantifying Portability of an Aspect Oriented Software Using Fuzzy Logic"

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

[18] "AJATO",http://homepages.dcc.ufmg.br/~figueiredo/ajato/

[19] A.Przybylek, "Analysis of the impact ot aspect-oriented programming on source code quality"

Khine Zar Ne Winn is a Ph.D. student in the University of Computer Studies, Mandalay, Myanmar. Her research interests mainly focus on evolvability of Aspect-Oriented software. To support such research goals, her research activities primarily include structural analysis of languages, aspect-oriented language design, aspect mining and aspect extraction.

Khin Mar Myo obtained Ph.D(IT) from University of Computer Studies, Yangon, Myanmar in 2005. She is presently served as associate professor in University of Computer Studies, Mandalay, Myanmer. She is interested in software quality, software development and requirement engineering.