International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-4, April 2013

Quantifying Portability of an Aspect Oriented Software Using Fuzzy Logic

Bhavesh Mathur, Satish Kumar Nath

***Abstract*—The quality of the software is measured in terms of its capability to fulfill the needs of the users and also its ability to achieve the developer's goal. Quality is mainly studied by quality model. In this paper an attempt has been made to quantifying the portability of aspect oriented software using ISO/IEC 9126 Model. Due to the unpredictable nature of software quality attributes, the fuzzy multi criteria approach has been used to evolve the quality of the software.**

***Index Terms*— Aspect Oriented Programming (AOP), Cross Cutting Concerns, ISO/IEC9126 Model, Portability**

I. INTRODUCTION

Fuzzy Logic is a powerful problem-solving methodology that can be used for applications in many areas such as embedded control and information processing. Fuzzy Logic provides an easier way to infer definite conclusions from highly imprecise, vague, and ambiguous information when compared with classical logic. Fuzzy Logic brings us close to human decision making, enabling one to analyze approximate data to precise solutions. Classical logic requires a high understanding of the system, whereas Fuzzy Logic allows for the modeling of a complex system using a higher level of abstraction originating from our experience and knowledge, without diving deep into the system [9].

Fuzzy Logic incorporates a simple, rule-based "If X and Y then Z" approach for solving the problem rather than solving it mathematically. The Fuzzy Logic model is completely empirical and relies on the experience of the operator rather than the technical understanding of the subject. The technique of triangular fuzzy has been adopted in this paper.

Software quality is a very important aspect for developers, users, and project managers. Different models are proposed for generic software applications. Out of these models, ISO/IEC 9126 model [1] is the most prominent model, which includes the findings of almost all other models. This is widely accepted and recognized in the industry and research community. Researchers made several efforts to implement this model for component based systems with minor modifications. This present work attempts to quantify the software portability using the ISO/IEC 9126 Model [1] as the base model with appropriate modifications to it. In order to deal with the fuzziness or uncertainty in quantifying the actual software parameters, the f*uzzy multi criteria* approach has been used. We used fuzzy logic approach to measure the portability of aspect oriented java application. [8]

Manuscript published on 30 April 2013.
* Correspondence Author (s)
Bhavesh Mathur, Department of Computer Engineering, Modern Institute of Technology & Research Centre, Alwar, India.
Satish Kumar Nath, Department of Computer Engineering, JIET School of Engineering & Technology for Girls, Jodhpur, India.

II. ASPECT ORIENTED PROGRAMMING

AOP provides a solution for abstracting cross-cutting code that spans object hierarchies without functional relevance to the code it spans. Instead of embedding cross-cutting code in classes, AOP allows you to abstract the cross-cutting code into a separate module (known as an aspect) and then apply the code dynamically where it is needed. [6] We can achieve dynamic application of the cross-cutting code by defining specific places (known as point cuts) in own object model where cross-cutting code should be applied. At runtime or compile time, depending on your AOP framework, cross-cutting code is injected at the specified pointcuts. Essentially, AOP allows us to introduce new functionality into objects without the objects' needing to have any knowledge of that introduction as discussed on webopedia [7].

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[2].AOP enables specific application that improve the quality of software.

System-wide policy enforcement- With AOP, you can create reusable aspects that enforce a variety of contracts and provide guidance in following "best" practices. For example, the Enterprise JavaBeans specification is about 600 pages long and describes many restrictions programmers should adhere to. Developer diligence alone can seldom achieve the required enforcement level [2].

III. PORTABILITY

Portability is related to the relative ease to transfer the software application from one environment to the other.[5] Portability is further subdivided into various sub characteristics that are: *adaptability, install-ability, co-existence, and portability compliance*. Each sub characteristic has certain metrics associated with them. The criteria to fuzzify these metrics are described below. The fuzzification criteria for different metrics have been described one by one with respect to different sub characteristics in the following section. [5]

1. **Adaptability:** Metrics describing adaptability are as follows: -

i. Compatibility in multiple OS: This parameter tells us how the compatibility of the software on multiple OS affects adaptability. If software is compatible with the most popular OS, the adaptability of the software is very high and vice versa. This metric can be fuzzified in the range of M to VH as [OS Compatible - Windows Only (M); Windows + Linux (H); Windows + Linux + Others (VH);].[5]

ii. Use of intrinsic tools: This parameter tells us how usage of intrinsic tools affects adaptability.

If software uses intrinsic tools, the adaptability of the software is medium, or else it is very high. This metric can be fuzzified in the range of M to VH as [Intrinsic Tools Usage – Yes (M); No (VH);]

iii. Pre-requisite packages needed: This parameter tells us how the number of pre-requisite packages (other than OS) required for the software affects adaptability. If no pre-requisite packages are required, the adaptability of the software is very high and vice versa. This metric can be fuzzified in the range of L to VH as [No.of Non OS Prerequisite Packages - Zero (VH); Popularly Available Packages (H); At Least One Package Not Popularly Available (L);].[5]

The value of adaptability sub characteristic is obtained by the weighted average of the above three metrics.

2. Install-ability: Metrics describing install-ability are as follows:-

i. Number of non-OS pre-requisite packages: This parameter tells us how the number of pre-requisite packages, other than OS, required for the software affects install-ability. If no non- OS pre-requisite packages are required; the install ability of the software is very high and vice versa. This metric can be fuzzified in the range of L to VH as [No.of Non OS Prerequisite Packages - Zero (VH); Popularly Available Packages (H); At Least One Package Not Popularly Available (L);].

The install-ability sub characteristic is simply obtained by the value of the above metric.

There is no need for any weighted average, as there is only one metric influencing the installability sub characteristic. [5]

3. Co-existence: Metrics describing co-existence are as follows: -

1. Frequency of deadlocks: This parameter tells us how the frequency of deadlocks in the running of the software affects co-existence. If deadlocks occur very frequently, the degree of the co-existence of the software is very low and vice versa. This metric can be fuzzified in the range of VL to VH as [Frequency of Deadlocks - Very Frequent (VL); Frequent (L); Sometimes (M); Rarely (H); Not at All (VH)].

The co-existence sub characteristic is simply obtained by the value of the above metric. There is no need for any weighted average as there is only one metric influencing the co-existence sub characteristic.

4. Portability Compliance: Metrics describing portability compliance are as follows:

i. Software adhering to portability compliance standards: This parameter tells us how the software's adherence to portability compliance standards affects portability compliance. This metric can be fuzzified in the range of L to VH as [Adheres to Compliance Standards (VH); doesn't Adhere to Standards (L)].

The portability compliance sub characteristic is simply obtained by the value of the above metric. There is no need for any weighted average as there is only one metric influencing the portability compliance sub characteristic.

After obtaining the values of all the sub characteristics under the portability characteristic, the value of the portability characteristic can be calculated simply by taking the weighted average of all the sub characteristics that were calculated above

$$r_{Portability} = r_1 \times w_1 + r_2 \times w_2 + \ldots r_n \times w_n = \Sigma r_i \times w_i$$

where i belong to the set {adaptability, install-ability, co-existence, and portability compliance}.[5]

IV. ASSUMPTIONS

- The values of all the parameters or characteristics have been quantified in the range 0 to 1. The overall quality of the software after quantification also appears in the range of 0 to 1.[8]
- Various characteristics and sub characteristics have been prioritized appropriately to calculate the total quality of the software. The weights considered vary from case to case.[8]
- Both ratings and weights have been quantified in terms of fuzzy, which are then converted into crisp numeric values using the Centroid Formula.[8]
- The fuzzy weighted average of all the quantified criteria and sub criteria is taken in order to arrive at the final quality. This has been done to maintain consistency so that the range of final values lies between 0 and 1. [8]

V. PROCEDURE

The exact procedure to quantify the software quality has been described in this section. As it has already been discussed in Section 3, the software quality is evaluated on the basis of the Software Quality Model that has been derived from the ISO/IEC 9126 Quality Model [1].

The procedure to quantify the software quality is as follows:

Step 1: Assign fuzzy ratings (r_i) to each and every metric that exists in the software model. [8]

Step 2: Assign fuzzy weights (w_i) to the sub characteristics, characteristics and perspectives. [8]

Step 3: Take the weighted average of the metrics (using their weights and ratings). [8]

Step 4: Take the weighted average of the sub characteristics for portability (using their weights and ratings) under the corresponding characteristics to evaluate the fuzzy rating. [8]

VI. MEASURING PORTABILITY OF ASPECT ORIENTED JAVA APPLICATION

Let us consider a Library application developed in Java which provides web access to College Library. A user can browse the various categories of books, add some books to a cart and finally checkout, do payment and get the books. For this app we might receive the requirements from a business analyst as follows:

- A login/registration screen to enter into Library.
- Users should be able to browse through various categories of books
- Users should be able to search books by name, author name, publisher
- Users should be able to add/remove the books to/from their cart
- Users should be able to see what items currently exist in their cart

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Users should be able to checkout and pay the corresponding amount through some payment gateway
- A successful message should be shown to users with all the details of their purchases.
- A failure message should be shown to users with the cause of failure.
- A Library administrator/manager should be granted access to add/remove/update book details.

All the above requirements fall under the "Functional Requirements" category. While implementing the above, we should also take care of the following things even though they are not explicitly mentioned:

- **Role based access to the UI**. Here, only Administrators/Managers should have access to add/remove/update book details. [Role based Authorization]
- **Atomicity in Purchasing.** Suppose a user logged into the Library and added 5 books into his cart, checked out and complete his

payment. In the back-end implementation we may need to enter this purchase details in 3 tables. If after inserting the data into 2 tables the system crashed, the whole operation should be rolled-back. [Transaction Management].

No one is perfect and no system is flawless. So if something went wrong and the development team has to figure it out what went wrong, logging will be useful. So, logging should be implemented in such a way that developer should be able to figure out where exactly the application failed and fix it. [Logging]The above implicit requirements are called Non-Functional Requirements. In addition to the above, performance should obviously be a crucial non-functional requirement for all public facing websites.

A. Calculation of PORTABILITY for Aspect Oriented Library application

Table 6.1.1 shows the real time values of the metrics related to portability. The values of these metrics have been acquired from five different users on the basis of a questionnaire.

Table 6.1.2 shows the ratings of the metrics corresponding to the *portability* characteristic, after they have been fuzzified on the basis of the criteria discussed in Sections 6. After classifying the metrics in the corresponding fuzzy sets, they have been assigned appropriate triangular fuzzy numbers as shown in the table 6.1.2.Table 6.1.3 shows the values of the weights that have been taken from five users. These weights have also been acquired via a questionnaire. This table also shows the fuzzified value of the weights after taking their average. Now the ratings (r_i) of the metrics (belonging to *portability*) have been multiplied by corresponding weights (w_i) and then added together to get the ratings of the corresponding sub characteristics.

Table 6.1.1 Values of Real time Metrics for the Portability

Sub Characteristic (Portability)	Questions (Metrics)	D1	D2	D3	Ratings
Adaptability	Compatibility		H		(0.5,0.7,0.9)
	Use of Intrinsic tool		H		(0.5,0.7,0.9)
	OS Pre-Requisite package needed		H		(0.7,0.9,1.0)
Install-ability	Non –OS Pre-Requisite package needed		H		(0.5,0.7,0.9)
Co-Existence	Frequency of deadlock		H		(0.8,0.9,1.0)
Portability Compliance	Whether software adheres to portability compliance standard or not		H		(0.5,0.7,0.9)

Table 6.1.2 Fuzzy Ratings of the Metrics Belonging to Portability Characteristic

Sub Characteristic (Portability)	Questions (Metrics)	D1	D2	D3	Ratings
Adaptability	Compatibility		H		(0.5,0.7,0.9)
Install-ability	Non –OS Pre-Requisite package needed		VH		(0.7,0.9,1.0)
Co-Existence	Frequency of deadlock		VH		(0.7,0.9,1.0)
Portability Compliance	Whether software adheres to Portability compliance standard or not		H		(0.5,0.7,0.9)

Table 6.1.3 Fuzzy Weights of the Metrics belonging to the Portability Characteristic

Sub Characteristic (Portability)	Metrics (weight)	D1	D2	D3	Weights
Adaptability	Relative importance for Compatibility	VH	H	M	(0.75,1.0,1.0)
	Non –OS Pre-Requisite package needed	H	H	M	(0.5,0.75,1.0)

Table 6.1.4 Fuzzy Ratings (calculated) of the Sub Characteristics belonging to the Maintainability Characteristic

Sub Characteristic (Portability)	Metrics	Average Weight	Rating
Adaptability	Compatibility	(0.75,1.0, 1.0)	(0.5,0.7,0.9)
Install-ability	Non –OS Pre-Requisite package needed		(0.7,0.9,1.0)
Co-Existence	Frequency of deadlock	NA	(0.7,0.9,1.0)
Portability Compliance	Whether software adheres to Portability compliance standard or not	NA	(0.5,0.7,0.9)

Now these ratings and weights of the sub characteristics such as Adaptability, Install- ability, Co-Existence, and Portability Compliance have to be combined by taking the weighted average to get the exact fuzzy rating of portability. This calculation is based on the formula:

$$r_{Portability} = r_1 \times w_1 + r_2 \times w_2 + \dots r_n \times w_n = \Sigma\, r_i \times w_i$$

Where i belongs to the set to the set { Adaptability, Install ability, Co-Existence, and Portability Compliance}.Here r_1 is rating by user 1, r_2 is rating by user2.Similarly w_1, w_2 are weights by different user.

VII. CONCLUSION

In this paper we have measured portability of Library application using fuzzy logic. By our calculation we can say that the Library software which is developed using Aspect oriented Approach is having high level adaptability, good install-ability, better co-existence and high level portability compliance standard. In future we can also compare these results with the same characteristics of any non aspect oriented application and we can also measure the remaining quality factors of ISO/IEC 9126 model.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

REFERENCES

[1] ISO/IEC 9126-1:2001, *"Software Engineering-Product Quality—Part 1: Quality Model"*, Int'l Organization for Standardization, 2001, Available at www.iso.org

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

[3] B. W. Boehm, J. R. Brown and M. L. Lipow, "*Quantitative Evaluation of Software Quality*," Proceedings of the 2nd International Conference on Software Engineering, San Francisco, CA, USA, October,1976, pp.592-605.

[4] R. G. Dromey, "*A model for software product quality*," IEEE Transactions on Software Engineering, Vol.21, No.2, February, 1995, pp.146-162.

[5] Jagat Sesh Challa et al "*Integrated Software Quality Evaluation: A Fuzzy multi criteria approach*", Journal of Information Processing System, Vol. 7, No.3, Sept.2011.

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

[7] "*Aspect Oriented Programming*" available at www.webopedia.com/TERM/A/aspect_oriented_programming.html

[8] Reena Dadhich and Bhavesh Mathur "*Measuring Reliability of an Aspect Oriented Software Using Fuzzy Logic Approach"* International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-5, June 2012,pp. 233-237.

[9] A.P.Singh and A. K. Vidyarthi, "*Optimal allocation of landfill disposal site: A fuzzy multi criteria approach*", Iranian Journal of Environmental Health Science & Engineering, Vol.5, No.1, 2008, pp.25-34.