Fuzzy Maintainability Model for Object Oriented Software System

Soumi Ghosh[1], Sanjay Kumar Dubey[2] and Prof. (Dr.) Ajay Rana[3]

Department of Computer Science and Engineering, Amity University
NOIDA, U.P., 201303, India

Abstract

In recent years for achieving considerable success in software system, maintainability plays a very crucial role and it is considered as an important quality characteristics. In this paper a maintainability model has been proposed to compare maintainability of object-oriented software system. Attempts have been made on software developed in different programming languages to make comparison of maintainability pattern using AHP and Fuzzy Index method.

Keywords: *Maintainability, AHP, software system, Object-oriented metrics.*

1. Introduction

For the entire life cycle of software products, the key factor maintainability is one of the important quality characteristics. The maintenance activities practically involve enhancement of software products, adapting existing products to new environment and correction of faults and errors. The ISO/IEC 9126 [5] standard defines maintainability as the capability of the software product to be modified, including corrections, improvements or adaptation of the software to changes in environment and in requirements and functional specifications. Good maintainability enables the software to meet the requirements of customers. Software maintainability is mainly evaluated through quantitative measurement, qualitative analysis and experiences of experts. On the basis of quantitative measurement some main problems required to be solved including the measurement, evaluation criteria and evaluation methods of software maintainability [12]. Software maintenance issues primarily dominate costs in software development.

Although now-a-days large amount of money is invested in software development within the life cycle of a software product but infact maintenance and adaptation of software is by far the costliest proposition. The maintainability of software is therefore, the most important part of the overall costs that has to be incurred during the lifetime of the system i.e. from the stage of development, its utility and till replacement. For this, the maintainability of software is by and large influenced by the quality of the source code. The main external quality attributes that have been identified by ISO/IEC 9126 [5] are functionality, reliability, usability, maintainability, portability and efficiency. The sub-characteristics of maintainability are defined as:

Analyzability- The capability of the software product to be diagnosed for deficiencies or causes of failures in the software or for the parts to be modified to be identified [6].
Changeability- The capability of the software product to enable a specified modification to be implemented [6].
Stability- The capability of the software product to avoid unexpected effects from modifications of the software [6].
Testability- The capability of the software product to enable modified software to be validated [6].

It is a fact that the most important indicators of final product quality are the external attributes but they can only be effectively measured after the product has been developed. The other approach for measuring the quality of a software system is through internal quality attributes. Such internal quality attributes are size, cyclomatic, coupling, inheritance etc. and these help the assessment of software quality in the early phase of development life cycle. The aim of this paper is to compare the maintainability of object-oriented system developed in different programming languages.

2. Object-oriented Metrics

The initial step involves selection of a group of object-oriented metric for each one of the internal quality attributes.

LOC (Line of Code): It is one of the earliest and simpler metrics for calculating the size of computer program. It is generally used in calculating and comparing the productivity of the programmers. Any line of program text excluding comment or blank line regardless of the no. of statements or parts of statements on the line is considered a line of code [10].

WMC (Weight Method per Class): - This metrics is count of methods implemented within a class or sum of complexities of methods. The number of methods and complexity of the methods involved is a prediction of how much time and effort is required to develop and maintain the class [11].

CBO (Coupling Between Objects): - Coupling is a measure of strength of association established by a connection from one entity to another. CBO counts the number of other classes to which a class is coupled. Number of distinct non- inheritance related class hierarchies on which a class depends gives CBO [11].

DIT (Depth of Inheritance): - Inheritance is a type of relationship among classes that enables programmers to reuse previously defined objects including variables and operators. DIT of a class within the inheritance hierarchy is the maximum length from the class node to the root of the tree and is measured by the number of ancestor classes [11].

3. Proposed Model

In terms of software quality one can easily distinguish internal and external quality [5]. The primary step of this paper is the establishment of a relationship between maintainability (external quality) and internal quality attributes such as size, complexity, coupling and inheritance.

Size- Size is used to evaluate the case of understandability of the code by the developers and maintainers. It may be measured in a variety of ways such as by counting all physical lines of code, the number of statements and the number of blank lines [8].

Complexity- By software complexity we mean the difficulty to preserve, modify and comprehend the software. It is the measure of how difficult a software system is and it is really desirable to achieve low complexity in software system [8].

Coupling- Coupling means the interdependence between different components or functions. It is the measure of interconnections among the modules in a software structure. It is the degree to which each program module depends on the other and it is required to achieve low coupling in software systems [8].

Inheritance- Inheritance is defined as classes having same methods and operations based on hierarchy. It is a mechanism whereby one object acquires the characteristics from one or more other objects [8].

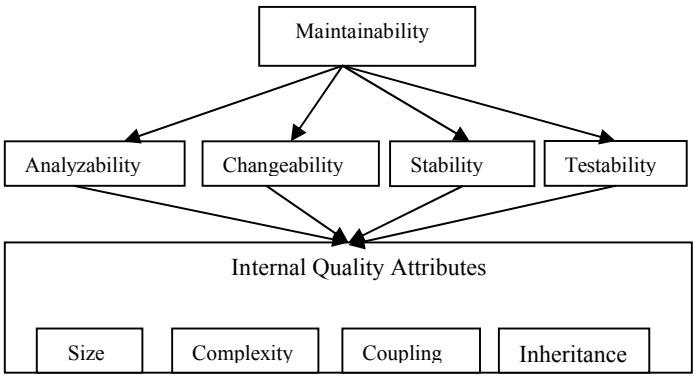

Fig1. Proposed Maintainability Model

4. Proposed Method

To evaluate the maintainability of object-oriented software systems, we use the AHP method [12] and fuzzy index method [7]. The method chosen for evaluation is shown in fig.1. The method uses following steps:-

i) To find weight matrix for maintainability sub-characteristics using AHP [12]
ii) To find weight matrix for object-oriented metrics using AHP [12]
iii) To find rank matrix using fuzzy index method [7]
iv) To find maintainability using step (i) to (iii)
v) To compare maintainability of software system developed in different languages

5. Case Study

For comparing maintainability, we selected two projects, scientific calculator and other tic-tac-toe game developed in Java, C++, and C# [4]. Tools selected as Analyst 4j, CCCC and Visual Studio code metrics power tool for Java, C++ and C #respectively. Using the fig.1 we applied AHP method [12] in order to determine the weight at all levels. The pairwise comparison method is used taking into consideration 1-9 scale to form the pairwise comparison matrix (A_k) between the 2nd and 3rd level metrics. We need to make consistency test. If C.R.=C.I./R.I.<0.1, then A_k is correct, otherwise modify A_k. From these $A_{k\ (k=1,\ 2,\ldots,n)}$ we will find $w_1^{(3)}$, $w_2^{(3)},\ldots,w_n^{(3)}$ and by comparing these weighted metrics we will get $w^{(3)}$. Then we have to calculate the eigen vector $w^{(2)}$ between the 1st and 2nd level metrics in the similar process. The weight metrics between the 2nd and 3rd level metrics as:

$w^{(3)} =$

0.1221	0.2483	0.2991	0.1073
0.1150	0.16062	0.26171	0.448
0.6162	0.44513	0.29378	0.3708
0.1467	0.14600	0.14543	0.0735

Fig.2 weighted matrix for maintainability sub-characteristics

The pairwise comparison metrics between 1st and 2nd level is given as follows:-

	Analyzability	**Changeability**	**Stability**	**Testability**	**Eigen vector (w)**
Analyz-ability	1	1	0.5393	0.6295	0.17015
Change-ability	1	1	0.3003	1	0.16500
Stability	1.8543	3.330	1	3.56	0.48265
Test-ability	1.5886	1	0.281	1	0.18220
Total					**1.0000**

$\lambda_{max} = 4.114$, C.I.= 0.038, C.R. = 0.0422

So, the matrix between 1st and 2nd level is given as:
$w^{(2)}= [0.17015, 0.16500, 0.48265, 0.18220]^T$

Then the values of each 3rd level metrics are calculated and determine whether the value is within the permission of range or not. If it is true then the value equals to 1, otherwise it is 0 [7]. Now we have to calculate the value vector (V_T) of the 2nd level metrics with the values of the 3rd level metrics and weight vector $w^{(3)}$. The binary values are described in Table 2 for scientific calculator developed in Java using tool [1].

	Threshold metrics [9]	Value (Analyst 4j)	Binary values
Size (LOC)	200-750	477	1
Complexity (WMC)	1-20	2.33	1
Coupling (CBO)	0-5	30	0
Inheritance(DIT)	0-3	0.88	1

Table2. Binary values of the 3rd level metrics

$V_{T1}^{1} = [0.3838, 0.55492, 0.70624, 0.6292]$

The binary values are described in Table 3 for scientific calculator developed in C++ using tool [2]

	Threshold metrics [9]	Value (CCCC)	Binary values
Size (LOC)	200-750	449	1
Complexity (WMC)	1-20	17	1
Coupling (CBO)	0-5	0	1
Inheritance(DIT)	0-3	0	1

Table3. Binary values of the 3rd level metrics

$V_{T1}^{2} = [0, 0, 0, 0]$

The binary values are described in Table 4 for scientific calculator developed in C# using tool [3]

	Threshold metrics [9]	Value (Code metric)	Binary values
Size (LOC)	200-750	1069	0
Complexity (WMC)	1-20	114	0
Coupling (CBO)	0-5	47	0
Inheritance(DIT)	0-3	7	0

Table4. Binary values of the 3rd level metrics

$V_{T1}^{3} = [1, 1, 1, 1]$

The binary values are described in Table 5 for tic-tac-toe game developed in Java using tool [1]

	Threshold metrics [9]	Value (Analyst 4j)	Binary values
Size (LOC)	200-750	396	1
Complexity (WMC)	1-20	2.86	1
Coupling (CBO)	0-5	16	0
Inheritance(DIT)	0-3	2	1

Table5. Binary values of the 3rd level metrics

$V_{T2}^{1} = [0.3838, 0.55492, 0.70624, 0.6292]$

The binary values are described in Table 6 for tic-tac-toe game developed in C++using tool [2]

	Threshold metrics [9]	Value (CCCC)	Binary values
Size (LOC)	200-750	228	1
Complexity (WMC)	1-20	4	1
Coupling (CBO)	0-5	4	1
Inheritance(DIT)	0-3	0	1

Table6. Binary values of the 3rd level metrics

$V_{T2}^{2} = [1, 1, 1, 1]$

The binary values are described in Table 7 for tic-tac-toe game developed in C# using tool [3]

	Threshold metrics [9]	Value (Code metric)	Binary values
Size (LOC)	200-750	440	0
Complexity (WMC)	1-20	151	0
Coupling (CBO)	0-5	33	0
Inheritance(DIT)	0-3	7	0

Table7. Binary values of the 3rd level metrics

$V_{T2}^{3} = [0.1221, 0.2483, 0.2991, 0.1073]$

The evaluation level is supposed to be M= {M1, M2, M3, M4} = {Poor, Fair, Good, Excellent} (1)

The threshold is taken as (0, 0.5, 0.75, 0.9, 1) [7] then we have c_1= 0.25, c_2= 0.625, c_3= 0.825, c_4= 0.95. With the value vector of the 2nd level metrics V_T, we put the values of analyzability, changeability, stability and testability into expert formulas and we get the membership functions. Finally we obtain the rank matrix as

$$R^1 = \begin{bmatrix} 1 & 0.56054 & 0 & 0 \\ 0.5352 & 1 & 1 & 1 \\ 1 & 0 & 4.8752 & 6.416 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$R^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^4 = \begin{bmatrix} 1 & 0.56054 & 0 & 0 \\ 0.5352 & 1 & 1 & 1 \\ 1 & 0 & 4.8752 & 6.416 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$R^6 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

We calculate maintainability (M) by using the formula rank matrix* $w^{(2)}$.

M^1= [0.26 0.92 3.7 0] M^2 = [0 0 0 1] M3 = [1 0 0 0]

M^4 = [0.26 0.92 3.7 0] M^5 = [0 0 0 1] M^6 = [1 0.95 0 0]

where M^1, M^2, M^3, M4, M^5, M^6 shows the quality of the maintainability in scientific calculator and tic-tac-toe game code.

By comparing with (1), we see that for scientific calculator and tic-tac-toe game, maintainability for Java code is good, for C++ code is excellent, for C# code is poor.

So, we conclude that software developed in C++ is more maintainable in comparison of code developed in Java and C#.

6. Conclusions

This paper proposes a method to compare the maintainability of object-oriented software system. The inputs for the method are size, complexity, coupling and inheritance which affect the maintainability of the software in different object-oriented programming languages such as Java, C++ and C#. These inputs were determined on the basis of survey from different experts which include project managers, system developers, researchers and other who are working on this field. This method shows any program build in the object-oriented language C++ has highest degree of maintainability as compared to other languages. In future we will try to evaluate the maintainability of object-oriented software system using the concept of FAHP (Fuzzy Analytic Hierarchy Process).

References

[1] http://www.Eclipse4you.com/?q=en/eclipse_plugins/analyst4j/accessed on 15/05/2012
[2] http://cccc.sourceforge.net/ accessed on 15/5/2012
[3] http://www.visualstudiomagazine.com/articles/2008/10/21/
[4] code-metrics.aspx accessed on 16/5/2012
[5] http://www.planet-source-code.com/ accessed on 14/5/2012
[6] ISO/IEC 9126, "International technology- software product evaluation- Quality characteristics and guidelines for their use", International Standard Organization, Geneva 1991
[7] ISO/IEC 9126-1 "Software engineering- Product quality – Part 1: Quality model", 2001
[8] J. Chen and X. Liu, "Software Maintainability Metrics Based on the Index System and Fuzzy Method", 1st International Conference on Information Science and Engineering (ICISE 2009)
[9] K. K. Aggarwal and Y. Singh, Software Engineering, New Age International (P) Limited, 2005
[10] L. H. Rosenberg and L. E. Hyatt, "Software Quality Metrics for Object-Oriented Environments", A report of SATC'S research on OO Metrics, Crosstalk Journal, vol. 10, issue: 2, pp. 1-6, 1997
[11] N. Fenton and S. Pfleeger, "Software Metrics: A Rigorous and Practical Approach", 2nd edition, 1997
[12] S. Chidamber and C. Kemerer, "A Metrics Suite for Object-Oriented Design", IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476-493, June 1994
[13] T. L. Saaty, "Multi criteria decision making: the Analytic Hierarchy process", RWS publications, Pittsburg, PA, 1988

[14] X.Wang, L. Chen and L. Guo, "The software Maintenance Measurement Research based on Ambiguous Synthesis Judge Method", Fire control and command control, vol. 33, pp. 14-16, April 2008

Soumi Ghosh is pursuing M. Tech (CS&E) at Amity University Uttar Pradesh, India. Her research areas include Software Engineering and Fuzzy Logic.

Sanjay Kumar Dubey is an Assistant Professor in Amity University Uttar Pradesh, India. His research areas include Human Computer Interaction, Software Engineering and Usability Engineering. He has published more than 30 research papers in reputed National & International Journals. He is member of IET. He is pursuing his Ph. D. in Computer Science and Engineering from Amity University, India.

Prof. (Dr.) Ajay Rana is a Professor and Director, Amity University Uttar Pradesh, India. He is Ph. D. in Computer Science and Engineering from U. P. Technical University. India. His research area includes Software Engineering. He has published more than 50 research papers in reputed National & International Journals. He is member of IEEE, CSI and IETE. He has received number of Best Paper awards for his work.