Maintenance Model for Open Source Object Oriented Designs

Er. Harinder Kaur
BBSBEC Fatehgarh Sahib

Er. Satwinder Singh
BBSBEC Fatehgarh Sahib

Dr. M. Javed
PAU Ludhiana

Abstract

Software reusability has considerable effect on software maintainability. Software maintainability increases as reuse of software components increases. Earlier model was developed which finds the maintainability of the class diagrams on the basis of understandability and modifiability on the basis of object oriented metrics of class diagrams This paper developed Enhanced model for Object-Oriented Softwares in Design phase which estimates the Maintainability of UML class diagrams in terms of their understandability, modifiability and reusability. As class diagrams play a key role in the design phase of object-oriented software therefore early estimation of their maintainability may help designers to incorporate required enhancements and corrections in order to improve their maintainability and consequently the maintainability of the final software to be delivered in future.

1. Introduction

Ever-changing world makes maintainability a strong quality requirement for the majority of software systems. The maintainability measurement during the development phases of object-oriented system estimates the maintenance effort, and also evaluates the likelihood that the software product will be easy to maintain . The maintainability is defined by IEEE standard glossary of Software Engineering as "the ease with which a software system or component can be modified to correct faults, improve performance or the ease with which an existing application or component can be reused. As class diagrams play a key role in the design phase of object-oriented software therefore early estimation of their maintainability may help designers to incorporate required enhancements and corrections in order to improve their maintainability and consequently the maintainability of the final software to be delivered in future. Hence, there is a need of developing a maintainability estimation model, which quantifies the maintainability of object-oriented software at the design stage.

2. Metric Selection

Metric Selection is very objective in nature. There are several ways in which these metrics can be picked up based on which a particular model can be developed. The goal of metric selection is to select such metrics which are Statistically significant, relevant in coherent context of Object Oriented Programming for developing following models:

- Maintenance Model
- Understandability Model
- Modifiability Model
- Reusability Model

For each model, we have to do extensive research to find out which metric or which measure of attribute of software will be highly relevant for the above said models. For this purpose, recent literature survey as well as cross and within company data set was chosen with help of experts and their performance, measurable expects for developing such model were studied. Here is list of metrics for each model:

Number of classes: The total number of Classes.
Number of Generalizations (NGen): The total number of Generalization relationships within a class diagram.
Number Of Generalizations Hierarchies (NGenH): The total number of generalization hierarchies within a class diagram.
Maximum DIT: It is the maximum DIT value obtained for each class diagram. The DIT value for class is the longest path from the class to the root of the tree value obtained for each class diagram. The DIT value for class is the longest path from the class to the root of the tree.
Number Of Aggregation Hierarchies (NAggH): The total number of aggregation hierarchies within a class diagram.

Method Hiding Factor (MHF) :MHF is defined as the ratio of the sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration.
Attribute Hiding Factor (AHF) :AHF is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration.
Method Inheritance Factor (MIF) :MIF is defined as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods (locally defined plus inherited) for all classes.
Attribute Inheritance Factor (AIF) :AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes (locally defined plus inherited) for all classes.
Polymorphism Factor (PF) :PF is defined as the ratio of the actual number of possible different polymorphic situation for clas to the maximum number of possible distinct polymorphic situations for class.

3. Models Development

Quantification of class diagram‟s understandability and modifiability is prerequisite for the maintainability estimation model. Therefore before developing Proposed Model, the paper has developed two models for understandability and modifiability:

3.1 Modifiability Model

In order to establish a multivariate model for modifiability of class diagram, metrics listed, will play the role of independent variables while modifiability will be taken as dependent variable. To identify metrics those are effectively contributing in the prediction of modifiability, the technique of backward stepwise multiple regression has been used. This procedure starts with a model, which initially includes all the independent variables and gradually eliminates those, one after another, that does not explain much of the variation in the dependent variable, until it ends with an optimal set of independent variables. Now applying backward stepwise regression, on the available data has resulted into the following modifiability model (1). This model has been taken from MEMOOD Model [4].

Modifiability=0.629+0.471*NC–0.173*NGen–0.616*NAggH–0.696*NGenH+0.396*MaxDIT (1)

Where, NC is the „Number of Classes‟, NGen is „Number of Generalizations‟, NAggH is „Number of Aggregation Hierarchies‟, NGenH is „Number of Generalization Hierarchies‟ and MaxDIT is Maximum DIT. From the model it can be interpreted that modifiability of class diagram is DIT‟, while „NGen‟ and „Number of directly proportional to „Number of Classes‟ and „Maximum Generalization and Aggregation Hierarchies‟ are inversely proportional to modifiability of class diagram.

3.2 Understandability Model

After establishing a model for modifiability the next task is to build a similar model for understandability also.Applying the same technique of stepwise backward multiple regressions on the available data resulted into the following understandability model (2)

Understandability=1.166+0.256*NC–.0.394*NGenH (2)

where, NC is the „Number of Classes‟ and NGenH is „Number of Generalization Hierarchies‟. From (3) it could be interpreted that understandability of class diagram is directly proportional to „NC‟, while „NGenH‟ is inversely proportional to the understandability of class diagram.

3.3 Reusability Model

Our approach is to derive formula to measure reusability of a class diagram based on following principles(3):

- Deeper a particular class is in the hierarchy, the greater the potential for reuse of inherited methods [6]. It states that reusability of a class increases with increase in inheritance of a class. So Inheritance has positive impact on reusability of a class.
- Another factor which affect the reusability potential is Encapsulation which enhance reusability of a software[5]. Encapsulation has positive impact on reusability of a class.
- Polymorphism indicates weakness of class understandability and may inhibit reuse. It enhances unnecessary complexity and overgeneralization. It indicates that Polymorphism has negative impact on reusability of a class.

Reusability of a class =a*(MIF+AIF)+ b*(MHF+AHF)–c*(PF) (3)
where a, b, c are empirical constants where a=1, b=1 and c= 0.5

3.4 Maintenance Model

In order to estabilish a multivariate model for

Maintainability of class diagram, Reusability, Modifiability and Understandability of class diagrams becomes independent variables while maintainability will be taken as dependent variable(4) .

Maintainability=.60-.15*Reusability+*Modifiability +.80*Understandability (4)

3.4.1 Statistical significance of the Model Observing the significance for the F-test in the last column of Analysis of Variance (Table I), it can be concluded that the Maintainability Model is statistically significant at a confidence level of more than 99%.

Table.I Anova for maintainability model

	Sum Of Squares	DF	Mean Square	F	Significance
Regression	8.54	3	2.85	46.80	.00
Residual	.36	6	.06		
Total	8.96	9			

Also the value of R2 (Coefficient of Determination) and Adjusted R2 in the Table II, is also very encouraging. As, it refers to the percentage or proportion of the total variance in maintainability by all the five metrics (independent variables) participating in the model.

Table.II Model summary for maintainability model

R	R Square	Adjusted R Square	Std. Error of the Estimate
.98	.96	.95	.25

Table.III Coefficients and statistical significance of Independent variables

	B	Std. Error	Beta	t	Significance
(Constant)	.60	.26	.00	2.31	.05
Reusability	-.15	.04	-.38	-3.84	.01
Modifiability	.55	.17	.52	3.25	.01
Understandability	.80	.18	.66	4.35	.00

As long as statistical significance and relevance of Individual independent variables in the Maintainability model is concern, It can be noticed from the last column of Table III, that each of the four metrics participating in the model is statistically significant at a significance level of 0.05 (equivalent to a confidence level of 95%).

4. Conclusion

The paper has developed model to quantify Maintainability of the class diagrams. This model is quantified in terms of Modifiability, Understandability and Reusability on basis of Encapsulation, Inheritance and Polymorphism metrics of class diagrams. Model has been developed through the technique of multiple linear regression. The paper also validates the quantifying ability of developed model for class diagrams.

5. References

[1] Mehwish Riaz, Emilia Mendes, Ewan Tempero "A Sysytematic Review of Maintainability Prediction and Metrics" Third International Symposium on Empirical Software Engineering and Measurement 2009.

[2] M. Genero, E. Manso, A. Visaggio, and M. Piattini, "Building Measure-Based Prediction Models for UML Class Diagram Maintainability," *Journal of Empirical Software Engineering*, vol. 12, no. 5, pp. 517 -549, 2007.

[3] P. Antonellis, D. Antoniou, Y. Kanellopoulos, C. Makris, E.Tjortjis, and N. Tsirakis, "A Data Mining Methodology for Evaluating Maintainability According to ISO/IEC-9126 Software Engineering Product Quality Standard," *Proc. 11th IEEE Conference on Software Maintenance and Reengineering (CSMR2007)*,21– 23 Mar. 2007, Amsterdam, Netherlands, 2007.

[4] S. W. A. Rizvi and R. A. Khan, "Maintainability Estimation Model for Object- Oriented Software in Design Phase (MEMOOD), 2010.

[5] Kung-Kiu Lau and Faris M. Taweel, "Data Encapsulation in Software Components," School of Computer Science, The University of Manchester Manchester M13 9PL, United Kingdom, LNCS 4608, pp. 1–16, 2007. _c Springer-Verlag Berlin Heidelberg 2007.

[6] Pradeep Kumar Bhatia, Rajbeer Mann, "An Approach to Measure Software Reusability of OO Design", National Conference on Challenges & Opportunities in Information

Technology (COIT-2008) RIMT-IET, Mandi Gobindgarh. March 29, 2008

[7] Satwinder Singh, K.S. Kahlon, "Effectiveness of Encapsulation and Object oriented Metrics to Refactor code and identify error prone Classes using Bad smells" ACM SIGSOFT Software Engineering Notes Page 1-10 September 2011, Volume 36, Number 5.

[8] Tong Yi, Fangjun Wu Chengzhi Ga , "A Comparison of Metrics for UML Diagrams**" ,** ACM SIGSOFT Software Engineering Notes Page 1 September 2004, Volume 29, Number 5.

[9] M. Dagpinar and J. Jahnke, "Predicting Maintainability with Object- Oriented Metrics –an Empirical Comparison," *Proc. 10th Working Conference on Reverse Engineering (WCRE'03)*, 13 - 17 Nov. 2003, pp. 155 - 164, 2003.

[10] Puneet Mittal, Satwinder Singh, K.S. Kahlon, "Identification of error prone Classes using Object oriented metrics" Advances in Computing and communications Volume 191, 2011, pp 58-68.