Measuring Maintainability in Early Phase using Aesthetic Metrics

MATINEE KIEWKANYA[1] AND PORNSIRI MUENCHAISRI[2]
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University
THAILAND

Abstract: - Assessing maintainability at the design level will help software designers to decide if the design of the software should be altered for reducing cost of software maintenance in later phase. This paper presents a controlled experiment carried out to investigate whether aesthetic metrics can be indicators of class and sequence diagrams maintainability and to establish maintainability models from aesthetic metrics. The experimental result indicates that aesthetic metrics can be indicators of class and sequence diagrams maintainability. Two maintainability models are constructed using two techniques: Discriminant Analysis and Decision Tree in order to find the best one. Our preliminary result shows that the accuracy of model constructed using Decision Tree is higher than that of the other one.

Key-Words: - Maintainability, Aesthetic Metric, Class Diagram, Sequence Diagram

1. Introduction

Maintenance of software system is frequently viewed as a phase of lesser importance than the design and development phase of the system life cycle. In fact, 50-70% of the total life cycle cost is spent on software maintenance [8]. In order to reduce maintenance cost, maintainability should be measured in early phase.

General approach for capturing software maintainability is the utilization of software metrics. Zhuo et al. presented constructing and testing software maintainability assessment models in [2]. This work focused on assessing software maintainability using metrics based on features of the source program. Constructing maintainability model for industrial software from software metrics was presented in [12]. [10] suggested metrics for measuring the maintainability of a target software system and discussed how to combine these metrics into a single index of maintainability.

The Unified Modeling Language (UML) is accepted as an industrial standard for modeling object-oriented design. In its current form, class and sequence diagrams are two major artifacts acted as blueprints of object-oriented software. Therefore, quality of object-oriented software ultimately implemented is heavily dependent on quality of both diagrams.

An aesthetic criterion is a general graphical property of the layout that we would like to have. Drawing graph by conforming to aesthetic criteria is claimed that the resultant graph is improved its readability. Commonly used aesthetic criteria for traditional graph drawing were presented in [3,7]. Many researches on UML diagrams focused on aesthetical principal taken from graph drawing. Purchase and his colleagues presented an empirical study attempting to identify the most important aesthetics for the automatic layout of class diagrams from a human comprehension point of view [6]. They also performed experiments assessing the effect of individual aesthetics in the application domain of class and collaboration diagrams in order to create a priority listing of aesthetics for this domain [5]. In [4], Eichelberger proposed some aesthetic criteria that reflect the highly sophiticated structural and semantic features of class diagrams.

From now on, we interchange the term class and sequence diagrams with the term software design model. Goals of our work are to investigate whether aesthetic metrics can be indicators of software design model maintainability and to establish maintainability models from aesthetic metrics. Following Boehm model [9], this work focuses on two sub-characteristics of maintainability: understandability and modifiability. Our experimental result shows that aesthetic metrics can be good indicators of maintainability. We construct two maintainability models applying two classification techniques called Discriminant Analysis and Decision Tree. Both obtained models can classify maintainability of software design model into three levels: difficult, medium and easy.

This work was supported by "Chulalongkorn-Industry Linkage Research Fund".

The next section presents aesthetic criteria and metrics used in this work. Section 3 describes a controlled experiment carried out to accomplish research goals. Then, experimental results are presented in section 4. Conclusions and future work are given in the last section.

2 Metric Selection

In this work, we select a set of aesthetic criteria, then define a set of aesthetic metrics corresponding to the selected aesthetic criteria. The metrics consist of metrics for class diagram and metrics for sequence diagrams which are shown in Table 1. In this table, the metrics corresponding to aesthetic criteria for traditional graph drawing that can be applied for class diagram are Cross, UnifEdgeLen, TotEdgeLen, MaxEdgeLen, UnifBends, TotBends, MaxBens, and Orthogonal [3,7]. Metrics namely Join, Center, Below, SameCo and Indicator are metrics corresponding to aesthetic criteria for class diagrams proposed by Purchase et al. [5,6] and Eichelberger et al. [4]. Metrics for sequence diagrams are adapted from metrics proposed by Paranen et al. [14].

3 A Controlled Experiment

This section describes the experiment carried out to pursue the following goals:

- to investigate whether aesthetic metrics can be indicator of maintainability.
- to establish prediction models for maintainability from aesthetic metrics.

3.1 Subjects

The experimental subjects used in this work were 60 graduate students from the Department of Computer Engineering at Chulalongkorn University, Bangkok, Thailand, who passed classes on Software Requirements Engineering and Object-Oriented Technology. These classes were supplemented by practical lessons where the students had the opportunity to design real-world object-oriented software using UML diagrams. The subjects were classified into 20 teams by considering grades they obtained from classes mentioned above. Each team had one A, one B+ and one B students. This was performed in order to reduce the difference of ability among teams of subjects to understand and modify software design models.

3.2 Experimental Material and Task

Forty software design models with different domains were used in this experiment. The documentation of each software design model consisted of the general software description, the class diagram, the sequence diagrams and a set of the examinations for assessing understandability and modifiability. Each subject team was asked to complete the examinations of 2 software design models that were randomly assigned for the team. Timeout periods of the examinations for assessing understandability and modifiability were 30 minutes and 40 minutes respectively. These timeout periods were determined from a pilot test. There was 20-minute break between experimental task of 2 software design models.

3.3 Data Collection

Maintainability is an external quality characteristic that cannot be measured directly. Following Boehm model [9], maintainability in this work is considered from its two sub-characteristics: understandability and modifiability. For each software design model, data collected from the experiment can be listed as follows.

- Understandability score is quantified from the mean of 3 subjects' score of the examination for assessing understandability.
- Modifiability score is quantified from the mean of 3 subjects' score of the examination for assessing modifiability.
- Maintainability score is calculated from the sum of understandability score and modifiability score. Maintainability level is captured by converting maintainability score into 0, 1 or 2 which indicates maintainability level: difficult, medium, and easy respectively. Each maintainability score can be converted using the following condition:

  If Maintainability score < Average value of maintainability scores – Standard deviation value of maintainability scores

  Then Maintainability level = 0

  Else If Maintainability score > Average value of maintainability scores + Standard deviation value of maintainability scores

  Then Maintainability level = 2

  Else Maintainability level = 1

The result of Kolmogorov-Smirnov test on maintainability scores shows that maintainability scores have normal distribution. Therefore, our approach for converting maintainability score into maintainability level can be considered valid.

Table 1. The aesthetic metrics used in the experiment.

Metrics for Class Diagram		
Metric	**Description**	**Aesthetic Criteria**
Cross	Total number of edge crossings	The total number of edge crossings should be minimized.
UnifEdgeLen	Standard deviation of the edge length (In this work, the edge length is measured in unit of centimeter.)	Edge lengths should be uniform.
TotEdgeLen	Total edge length	The total edge length should be minimized.
MaxEdgeLen	Maximum edge length	The maximum edge length should be minimized.
UnifBends	Standard deviation of the number of bends on the edges	The standard deviation of the number of bends on the edges should be minimized.
TotBends	Total number of bends in the drawing	The total number of bends should be minimized.
MaxBends	Maximum number of bends on the edges	The maximum number of bends on the edges should be minimized.
Orthogonal	Total number of edges fixed to an orthogonal grid divided by total number of edges	Nodes and edges should be fixed to an orthogonal grid.
Join	Total number of joined hierarchies divided by total number of hierarchies	generalizations, aggregations and compositions should be joined.
Center	Total number of hierarchies that the parent is located as the center of its children divided by total number of hierarchies	A parent node should be positioned as close as possible to the median position of its children.
Below	Total number of hierarchies that the parent is located above its children divided by total number of hierarchies	A child node should be positioned below its parent.
SameCo	Total number of hierarchies that the children nodes are located on the same vertical or horizontal coordinate divided by total number of hierarchies	Nodes on the same hierarchy level should have the same vertical or horizontal coordinate.
Indicator	Total number of edges representing association relations that have clear label and directional indicator divided by total number of edges representing association relations	Edge should be clearly labeled and should have directional indicator.
Metrics for Sequence Diagrams		
Metric	**Description**	**Aesthetic Criteria**
ACrossS	Average number of crossings (Total number of crossings in all sequence diagrams divided by total number of sequence diagrams)	The total number of edge crossings should be minimized.
MaxEdgeLenS	Maximum edge length of all sequence diagrams	The maximum edge length should be minimized.
AUnifEdgeLenS	Average standard deviation of edge length (Summation of standard deviation of edge length of all sequence diagrams divided by total number of sequence diagrams)	Edge lengths should be uniform.
ASubsetSeptS	Average number of distinct subets of particiapants (total number of distinct subsets of participants of all sequence diagrams divided by total number of sequence diagrams)	The distinct subsets of participants should be maximized.

4 Experimental Results

This section presents metric validation and constructing maintainability models.

4.1 Metric Validation

The main goal of our experiment is to analyze class and sequence diagrams for validating the possibility of the aesthetic metrics being used as good indicators of class and sequence diagrams maintainability. We propose the following hypotheses:

H_0 : the aesthetic metrics cannot be indicators for classifying maintainability level.

H_1 : the aesthetic metrics can be indicators for classifying maintainability level.

In order to test this hypothesis, we select Multivariate analysis of variance (MANOVA) [1]. This is a test of overall relationship between groups and predictors by considering variance in the set of predictors that effects on group classification. The result of this test is shown in Table 2. H_0 is rejected because P-value is less than significant level (0.001). We can conclude that three groups of maintainability levels can be distinguished on the basis of the combination of the aesthetic metrics.

Table 2. MANOVA test.

Source of Variance	Wilks' Lambda	df1	df2	Multivariate F	P-value
Maintainability level	0.003094	32	44	23.3427	<0.001

4.2 Constructing Maintainability Models

The data collected from the experiment are analyzed in order to construct maintainability models.

4.2.1 Correlation Analysis

Correlation between each pair of metrics mentioned in Table 1 is considered in order to discard metrics that provide redundant information. This can be automated by applying Pearson's correlation test.

For each couple of highly correlated metrics, only one of them will be selected. Linear regression with one independent variable is performed for each metric. Then, adjusted R square value is used to determine the best choice. Adjusted R square value of independent variable (aesthetic metric) indicates that it can explain the variance of dependent variable (maintainability level) well or not. The metric with higher adjusted R square value will be chosen. Following this selection process, MaxEdgeLen, MaxEdgeBens and MaxEdgeLenS are discarded. One more discarded metric is Cross since its value obtained from sample class diagrams are not different (more than 80% are zero value). Therefore, it is useless for classifying maintainability level.

4.2.2 Discriminant Analysis

Discriminant analysis employs a concept very similar to the regression equation, and it is called the discriminant function [11]. The general form of the discriminant function is

$$D = a+b_1x_1+b_2x_2+\ldots+b_nx_n$$

where, D is the discriminant score, a is a constant value, b_i is the classification function coefficient and x_i is independent variable. We apply Fisher's linear discriminant function. For classifying object into one of N groups, this technique generates N discriminant functions representing N groups. New object will be assigned to the group giving the highest discriminant score. So, for classifying three levels of maintainability, three following discriminant functions are generated by this technique. From now on, this functions will be called 'Discriminant Analysis maintainability model'.

Difficult level's function:

D_Main = -1.276 × UnifEdgeLen +0.861 × TotEdgeLen - 2.868 × UnifBends-1.377 × TotBends+12.003 × Orthogonal-4.107 × Join+124.847 × Center+20.707 × Below+19.424 × SameCo - 2.953 × Indicator + 0.676 × ACrossS-2.314 × AUnifEdgeLen+1.176 × ASubSetSeptS-96.411

Medium level's function:

M_Main = -1.180 × UnifEdgeLen +0.967 × TotEdgeLen +2.674 × UnifBends-1.786 × TotBends +9.867 × Orthogonal-7.721 × Join+125.045 × Center +18.743 × Below+29.401 × SameCo - 2.283 × Indicator + 0.699 × ACrossS-3.064 × AUnifEdgeLen+2.289 × ASubSetSeptS-94.547

Easy level's function:

E_Main = -1.374 × UnifEdgeLen +0. 948 × TotEdgeLen - 3.766 × UnifBends-1.5 × TotBends +14.66 × Orthogonal-3.978 × Join+130.887 × Center+21.803 × Below+23.524 × SameCo - 3.914 × Indicator + 0.745 × ACrossS-3.429 × AUnifEdgeLen+0. 075 × ASubSetSeptS-94.199

In order to classify a new software design model, function D_Main, M_Main and E_Main will be calculated. Then the software design model will be allocated to the group that provides the highest value among 3 functions.

4.2.3 Decision Tree

In this work, we use ID3 algorithm for developing decision tree [13]. Measure called *information gain* is a statistical property that measures how well a given attribute separates the training examples according to their target classification. In order to define information gain precisely, a measure commonly used in information theory, called *entropy*, that characterizes the impurity of an arbitrary collection of examples, should be defined first. Let S is a collection of examples, c is the number of classes, then the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i$$

Given entropy as a measure of the impurity in a collection of training examples, information gain is a measure of the effectiveness of an attribute in classifying the training data. It is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. The information gain, Gain(S,A) of an attribute A, relative to a collection of examples S, is defined as

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

ID3 determines the information gain for each candidate attribute, then selects the one with highest information gain to be tested first in the tree. The process of selecting a new attribute and partitioning the training examples is repeated for each nonterminal descendant node.

The maintainability model obtained by applying Decision Tree is shown in Figure 1. From now on, this tree will be called 'Decision Tree maintainability model'. A new software design model is classified by starting at the root node of the tree, testing the metric specified by this node, then moving down the tree branch corresponding to the value of the metric. This process is then repeated for the subtree root at the new node until leaf node is reached.

4.3 Comparison between Discriminant Analysis and Decision Tree Maintainability Models

Decision Tree maintainability model indicates that the metrics that can be good indicators for maintainability are ACrossS, ASubsetSeptS, SameCo, TotEdgeLen, Below and UnifEdgeLen respectively. Metric testing in each node corresponds to metric criteria proposed in Table 1 except testing of UnifEdgeLen. For example, a software design model which have high value of ACrossS (more than 11) will be classified into Difficult level represented by 0. This classification corresponds to metric criteria for ACrossS as the total number of edge crossings should be minimized. We cannot conclude that which metrics are better than other metrics to be used as maintainability indicators for Discriminant Analysis maintainability model. The classification function coefficient of each metric cannot indicate that which metric is better because of difference of metric unit.

Both maintainability models are used to predict maintainability level of 40 software design models that are used for constructing the models. In order to validate model accuracy, we decide to use technique of cross validation [13] as a result of lacking new sample software design models. This technique constructs maintainability model from 39 sample software design models and leaves one out in order to validate model accuracy. So, maintainability model is constructed 40 times by changing a sample software design model that is left out. Table 3 and Table 4 show the results of classifying original group cases and cross-validated group cases. Table 5 shows misclassification rates of both maintainability models.

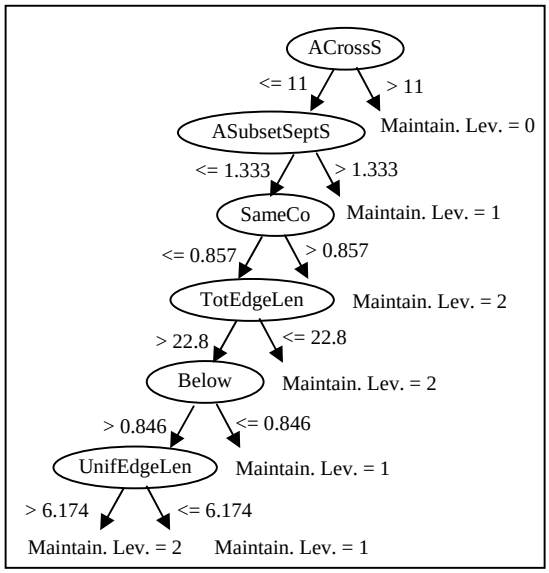

Figure 1. Decision Tree maintainability model.

Percent of Type I error is computed from the total number of cases that meet the following conditions and is divided by the total number of cases.

- case in group of 0 is classified as 1
- case in group of 1 is classified as 0
- case in group of 1 is classified as 2
- case in group of 2 is classified as 1

Percent of Type II error is computed from the total number of cases that meet the following conditions and is divided by the total number of cases.

- case in group of 0 is classified as 2
- case in group of 2 is classified as 0

Type II error is more fatal than Type I error. The result in Table 5 shows that Type I error percentage of Disciminant Analysis maintainability model is higher than that of Decision Tree maintainability model but Type II error percentage is in contrast. However, Overall error percentage of Decision Tree maintainability model is lower than that of Discriminant Analysis maintainability model.

5. Conclusions and Future Work

This paper presents a controlled experiment carried out in order to investigate whether aesthetic metrics can be indicators of class and sequence diagrams maintainability and to establish maintainability models from aesthetic metrics applying Discriminant Analysis and Decision Tree. Our result shows that aesthetic metrics can be indicators of class and sequence

Table 3. Classifying group cases using Discriminant Analysis maintainability model.

		Main. Level	Predicted Group Membership 0	1	2	Total
Original	Count	0	6	2	0	8
		1	0	17	2	19
		2	0	1	12	13
	%	0	75.0	25.0	0.0	100.0
		1	0.0	89.5	10.5	100.0
		2	0.0	7.7	92.3	100.0
Cross-validated	Count	0	4	2	2	8
		1	1	15	3	19
		2	0	3	10	13
	%	0	50.0	25.0	25.0	100.0
		1	5.3	78.9	15.8	100.0
		2	0.0	23.1	76.9	100.0

87.5% of original grouped cases correctly classified.
72.5% of cross-validated group cases correctly classified.

Table 4. Classifying group cases using Decision Tree maintainability model.

		Main. Level	Predicted Group Membership 0	1	2	Total
Original	Count	0	5	2	1	8
		1	0	19	0	19
		2	0	0	13	13
	%	0	62.5	25.0	12.5	100.0
		1	0.0	100.0	0.0	100.0
		2	0.0	0.0	100.0	100.0
Cross-validated	Count	0	4	2	2	8
		1	2	16	1	19
		2	1	2	10	13
	%	0	50.0	25.0	25.0	100.0
		1	10.5	84.2	5.3	100.0
		2	7.7	15.4	76.9	100.0

92.5% of original grouped cases correctly classified.
75% of cross-validated group cases correctly classified.

Table 5. Misclassification rates of maintainability models (%) .

Maintainability Model	Original			Cross-validated		
	Type I Error	Type II Error	Overall Error	Type I Error	Type II Error	Overall Error
Discriminant Analysis	12.5	0.0	12.5	22.5	5.0	27.5
Decision Tree	5.0	2.5	7.5	17.5	7.5	25.0

diagrams maintainability. The result of model validation indicates that accuracy of Decision Tree maintainability model is higher than that of Discriminat Analysis maintainability model. This result may be considered as the preliminary finding. For future work, this experiment should be repeated with more number of sample software design models. Other classification techniques will be used for constructing maintainability model in order to improve its accuracy.

Acknowledgement:
The authors would like to thank Prof. Ross Jeffery for his valuable comments and suggestions.

References:

[1] B.G. Tabachnick and L.S. Fidell, *Using Multivariate Statistics*, Allyn&Bacon, 2001.

[2] F. Zhuo, B. Lowther, P. Oman, and J. Hagemeister, "Constructing and Testing Software Maintainability Assessment Models," *Proc. of the 1st International Software Metrics Symposium*, 1993, pp. 61-70.

[3] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis, *Graph Drawing*, Prentice Hall, 1999.

[4] H. Eichelberger, "Aesthetics of Class Diagrams," *Proc. of the 1st IEEE Interation Workshop on Visualizing Software for Understanding and Analysis*, 2002, pp. 23-31.

[5] H.C. Purchase, J-A. Allder and D. Carrington, "Graph Layout Aesthetic in UML Diagrams: User Preferences," *Jounal of Graph Algorithms and Applications*, Vol.6, No. 3, 2002, pp. 255-279.

[6] H.C. Purchase, L. Colpoys and M. Mcgill, "UML Class Diagram Syntax: An Empirical Study of Comprehension", *Proc. of the Australian Symposium on Information Visualisation*, 2001.

[7] H.C. Purchase, R.F. Cohen, and M. James, Validating Graph Drawing Aesthetics, *Lecture Notes in Computer Science*, 1027:435-446,1996.

[8] I. Sommerville, *Software Engineering*, Addison Wesley, 1996.

[9] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing, 1997.

[10] P. Oman and J. Hagemeister, "Metrics for Assessing a Software System's Maintainability," *Proc. of Conference on Software Maintenance 1992*, 1992.

[11] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall International, 1988.

[12] S. Muthanna, K. Kontogiannis, K. Ponnambalam, and B. Stacey, "A Maintainability Model for Industrial Software Systems using Design Level Metrics," *Proc. of the 7th Working Conference on Reverse Engineering*, 2000, pp.248-256.

[13] T. M. Mitchell, *Machine Learning*, The McGraw-Hill Companies, Inc., 1997.

[14] T. Poranen, E. Makinen, and J. Nummenmaa, "How to draw a sequence diagram," *Proc. of the 8th Symposium on Programming Languages and Software Tools*, 2003.