Estimating Software Logical Stability using ANN from Class diagram and Sequence diagram

Dith Nimol* and Somchai Prakancharoen** and Pornsiri Muenchaisri***

Abstract

A problem that always occurs in software maintenance process is the ripple effect, which is when one object changed thus affecting the other objects in the same program. Stability can be defined as resistance to the ripple effect. This is divided into two aspects: logical stability and performance stability. In this paper, we emphasize the logical stability only. Logical stability of a class indicates its resistance to interclass propagation of changes when other classes are modified. However, logical stability can be obtained only after the program is executed. Thus, logical stability estimation at the design phase is very useful since knowing the amount of class logical stability may enable a designer to decide whether restructuring the design model is needed or not. In this paper, we propose a methodology to estimate logical stability of a class by using artificial neural network (ANN). Artificial neural network is an approach used to estimate the value of a class logical stability from historical data in the repository by choosing a multi layer Perceptron method. The establishment of this approach is evaluated for estimating the accuracy between the actual class logical stability measured from source code and an estimated class logical stability using ANN and Multiple Regression. The experiments result shows that Mean Magnitude Relation Error (MMRE) of ANN is 21.18 % and MMRE of Multiple Regression is 29.72 %. It shows that Artificial Neural Network is effective in the estimation.

Keywords: Object-oriented Software Measurement, Logical stability, Artificial neural network, Design metrics.

1. Introduction

The Software evolution refers to the on-going enhancement of existing software systems, involving both development and maintenance. Software main-tenance has been recognized as the most costly and difficult phase in the software life cycle. The software maintenance process and the important software quality attributes that affect the maintenance effort are discussed. One of the most important quality attributes of software maintainability is the stability of a program, which indicates the resistance to the potential ripple effect that the program would have when it is modified [1]. Ripple effect is the phenomena when changes to one program area have tendencies to be felt in other program areas. Ripple effect can be divided into 2 aspects. First is the logical ripple effect of change in one class causes an inconsistency in other classes that is related to the changed area. Second is performance ripple effect which is the effect of change in one class on performance of other classes. In this paper, we emphasize the logical stability.

Fixing errors, adding or removing functions of software, or enhancing function’s capability generates changes in components of software. Changes will affect the modified class and other related classes. Logical stability of a class indicates its resistance to interclass propagation of changes when other classes are modified. Logical stability prediction at the design phase is required because a designer can use it in order to decide whether restructuring the design model is needed or not.

Many researchers have focused on the software logical stability in object oriented software engineering. In the paper “Ripple effect analysis of software maintenance,” Yau, Collofello and MacGregor [2] discussed the ripple effect analysis of software maintenance. Ripple effect can be divided into two aspects: logical ripple effect and performance ripple effect. In [1], Yau and Collofello proposed an algorithm to compute logical stability. In [3], Elish and Rine adapted the Yau and Collofello’s alogithm to calculate the class logical stability in object oriented programming. Rangsiyawath and Muenchaisri [4] introduced a new model for estimating software logical stability from UML diagram by using multiple regressions. Cheewaviriyanon and Muenchaisri [5] introduced a new model for estimating software logical stability from

* *Department of Information Technology, Faculty of Information Technology, King Mongkut’s University of Technology North Bangkok.*

** *Department of Computer and Information, Faculty of Applied Science, King Mongkut’s University of Technology North Bangkok.*

*** *Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University.*

UML diagram by analogy. In [6, 7], the researchers applied Artificial Neural Network (ANN) for estimating software size.

In this paper, we propose an approach for estimating class logical stability using Artificial Neural Network (ANN). Thirty design metrics, which are derived from class and sequence diagrams, are source features to estimate class logical stability. The proposed approach uses artificial neural network is a technique to estimate the value of a class logical stability from historical data in the repository by choosing a multi layer perceptron method to train the data. With this approach, class logical stability can be estimated in the early phase of the software development. Some experiments using this technique are demonstrated with 115 classes of Library, Employee, Payroll, SHM, ShapeInheritence, Wavelets, ShapeDrawWithFiles, TextEditor, Benzene Reactions and Predict.

This paper is organized as follows. Section 2 provides technical backgrounds. Section 3 presents the technique for estimating class logical stability using multiple regression and an estimated class logical sta-bility using artificial neural network. Section 4 demon-strates the results from case study. Finally, section 5 concludes the paper and outlines the future works.

2. BACKGROUND

In this section, the basic notion about a logical stability and an artificial neural network estimation technique are summarized. This section is divided into 3 subsections: logical stability calculation, multiple regression estimation and artificial neural network estimation as follows.

2.1 Class Logical Stability (CLS)

Logical stability of a class indicates its resistance to interclass propagation of changes when other classes are modified. Logical stability prediction at the design phase is required because a designer can use it in order to decide whether restructuring the design model is needed or not.

The class logical stability is calculated using the equation proposed by Elish and Rine in [3] as follow:

$$CLS\neg i = 1 - CLREi \quad (1)$$

Where *CLSi* is Class Logical Stability of class *i* (*i*=1 to *n*) and *CLREi* is Class Logical Ripple Effect of class *i* (*i*=1 to *n*).

The class logical ripple effect can be calculated by the equation:

$$CLREi = \frac{NTE_i}{NTC} \quad (2)$$

Where *NTEi* is the Number of Times which class *i* (*i*=1 to *n*) received Effect from changes (add attribute, etc.) and TNC is Total Number of Changes.

2.2 Multiple Regression

Multiple Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships.

The result of a multiple linear regression analysis is an equation that describes the relation. The equation is in the following form:

$$\hat{Y} = a + b_1x_1 + b_2x_2 + b_3x_3 + ... + b_kx_k + e \quad (3)$$

which $\hat{Y}$ is an estimated value of dependant variable, *a* is a constant value, b_i is an estimated value of partial regression coefficient where *i*=1 to k and x_i is an independent variable's value where *i*=1 to *k*.

2.3 Artificial Neural Network

Artificial Neural Network technique [6] is kind of multilayer Feed-forward, the value of design metrics is used to estimate the value of class logical stability. The value of design metrics from historical data in repository will be used to input to the training set. The data was trained with multi layer perceptron method until the estimation value of design metric is output.

3. METHODOLOGY

3.1 Collecting Data

The resource used in this research is from the internet. Source code is an open source Java software collection from http://sourceforge.net/. The software is divided into 2 sets consisting of research set and evaluation set. The software used in this research is text editor software which is a commonly used software type.

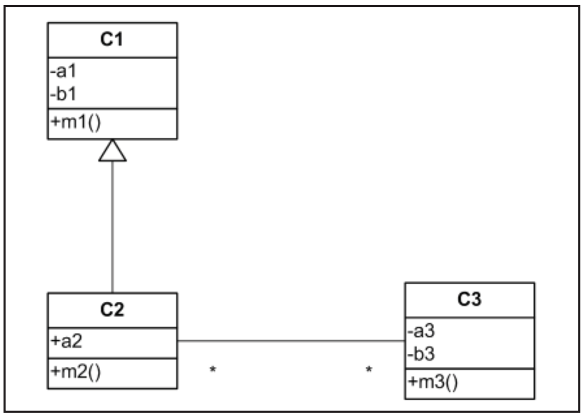

Figure 1 *An Example class diagram [4].*

3.2 Process for Study

Step1: Measure Class Logical Stability.

The class logical stability of software in research set is calculated. The method used to calculate class logical stability in this research is a modified algorithm based on an algorithm proposes by Elish and Rine in [3].

To further illustrates Appendix A [4], consider a simple class diagram in figure1. Figure 1 shows a simple class diagram contains 3 classes C1, C2 and C3. C2 is a sub class of C1. Method m3() of C3 uses attribute a2 of C2 and C3 is an association class of C2 (C3 uses C2). Suppose that we delete attribute a2 of C2, from Appendix A when a public attribute is deleted it will affect the class that contain the attribute, a sub classes of contained class and classes that use the attribute. In this case class C2 will receive the effect as a contained class and since method m3() of C3 uses attribute a2, class C3 will receive an effect as well.

To illustrate the class logical stability calculation, consider an example source code in figure 3 which is the source code of classes in figure 1. The calculation starts by setting TNC to zero and selecting C1 to be a focus class, and then applying changes to C1. If change can be applied then increase TNC by one and increase NTE of all affected classes by one. Some changes cannot be applied to some classes, such as change number 2 from Appendix A cannot apply to C1 because it does not has a parent relation. In this case the TNC and NTE is remaining the same. After apply all changes to C1, the next class will be selected. When changes are applied to all classes and the number of TNC and NTE has been collected, the class logical stability is calculated using the formula in section 2.1.

The research set has been reversed engineered to develop a class diagram and sequence diagrams of each software. We use MagicDraw UML version 9.5 [8] to reverse engineer the research set. For example, the source code in figure 3 can be reversed to class diagram in figure 1. The example of sequence diagram reverses from source code in figure 3 is shown in figure 4. After the class diagram and sequence diagram are reversed, the diagrams are converted to XML file.

The Diagram Metric is measured using SDMetrics version

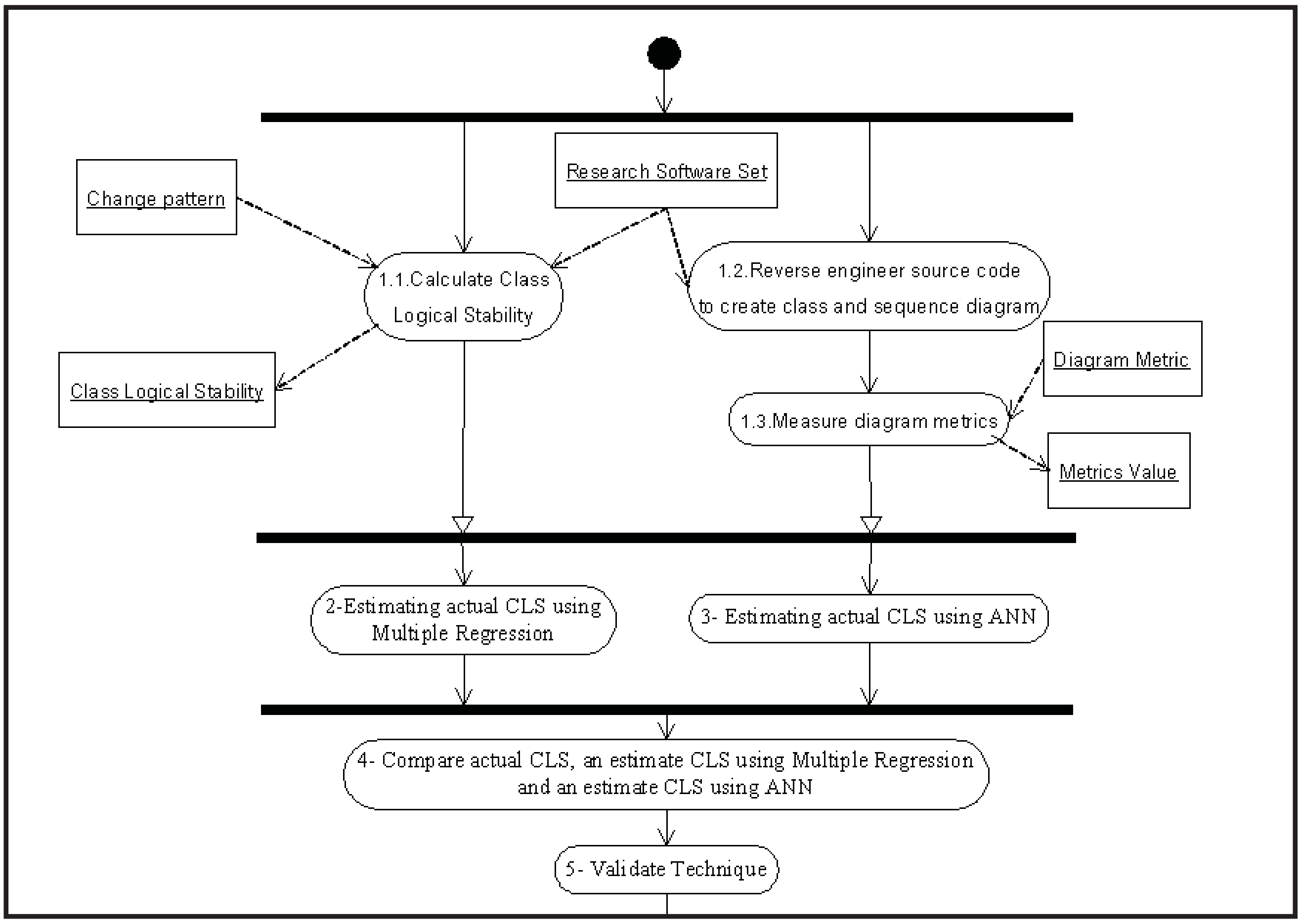
Research Software Set
Change pattern
1.1.Calculate Class Logical Stability
1.2.Reverse engineer source code to create class and sequence diagram
Class Logical Stability
Diagram Metric
1.3.Measure diagram metrics
Metrics Value
2-Estimating actual CLS using Multiple Regression
3- Estimating actual CLS using ANN
4- Compare actual CLS, an estimate CLS using Multiple Regression and an estimate CLS using ANN
5- Validate Technique

***Figure 2** The activity of this research.*

2.0 [9]. SDMetrics 2.0 is a tool that can measure several diagram metrics from a diagram that is in XML and XMI format. The reversed diagrams has been loaded to be analyzed in SDMetrics and the diagram metrics are measured from the XML file. The full detail of list diagram metrics description is in Appendix B.

Step 2: Estimate class logical stability using Multiple Regression.

The multiple regression [6] is an analysis technique used to find a relation between one dependant variable and many independent variables. The values of the design metrics are independent variables which will be used to estimate the value of dependent variable in class logical stability to get new classes using multiple regression.

Step 3: Construct new technique for estimating class logical stability using ANN.

The ANN technique [6] is a multilayer Feed-forward technique, the value of the design metrics is used to estimate the value of the class logical stability.

The value of the design metrics from historical data in the repository will be used to train data with multi layer percetron method. The calculation of this method is using WEKA program [10].

Step 4: Comparing actual class logical stability with an estimated CLS using Multiple Regression and an estimated CLS using ANN.

The establishment of the new technique will be evaluated for estimation accuracy between the actual class logical stability measured from source code and an estimated class logical stability using multiple regression and an estimated class logical stability using ANN.

The proposed new technique is evaluated using MMRE and Prediction at Level L [11].

Step 5: Technique Validation.

After the estimation technique is established, the technique is evaluated using prediction at level L. To evaluate the technique, the Mean Magnitude of Relative Error (MMRE) of each class needs to be calculated first. The MMRE of each class can be calculated using equation (5) in section 3.2, which is the relative error between the actual class logical stability measured from source code, the estimated logical stability estimated by using ANN and the estimated logical stability using multiple regression. After the MMRE of each class is calculated, we can calculate the prediction value of each interest level by counting the number of classes that have an MMRE less than the actual value of each level and dividing them by the number of total classes in the evaluation set.

public class C1{ private int a1; private int b1; public int m1() { a1= 2; return a1; } }	public class C2 ex-tends C1 { public int a2; public void m2(){ a2 = 0 ; } }	public class C3 { private int a3; private int b3; private C2 c; public void m3() { C2 c =new C2(); a3 = c.a2; } }

Figure 3 *A source code example.*

$$PRED\,(l) = \frac{k}{n} \qquad (4)$$

where l is a tolerance rate, *k* is a number of projects or software that has MMRE less than or equal to l and *n* is a number of projects or software.

Mean Magnitude of Relation Error [6] is an evaluation me-thod to find different error measurements in the prediction for software development effort estimation. MMRE can be calculated by the following equation:

$$MMRE = \frac{k}{n}\sum_{i=1}^{n}\left|\frac{M_{act} - M_{est}}{M_{act}}\right| \qquad (5)$$

Where M_{act} is Measurement of actual class logical stability and M_{est} is Measurement of estimated class logical stability.

4. CASE STUDY

In this section, the ANN technique is demonstrated with 115 classes of match, image processing and database applications. The results at each step of the methodology are shown as follows.

4.1 Class Logical Stability Repository

In this subsection, the data is collected from database, math and image processing applications, which are Java open source software in the same domain. The 115 classes of math, image processing and database applications consist of 30 classes of Library, 5 classes of Employee, 10 classes of Payroll, 8 classes of SHM, 3 classes of ShapeInheritence, 19 classes of Wavelets, 6 classes of ShapeDrawWithFiles, 18 classes of TextEditor, 10 classes Benzene Reactions and 5 classes of Predict [12]. The value of class logical stability is measured from Java source code of software and the value of 30 class design metrics are measured from class and sequence diagrams. Some of the data which are the class logical stability and the value of 30 class design metrics are shown in Table 1.

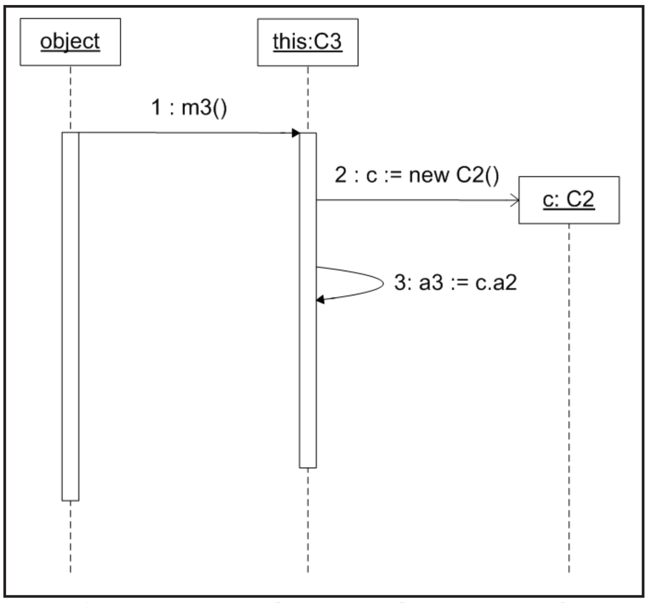

***Figure 4** A reversed sequence diagram example.*

***Table 1** Case representation of a class.*

Class name	ClassLogical Stability	Num Attr	Num Ops	Includes
Shapes-Inheritant	0.760000	0	0	0
Triangle	0.720000	0	4	0
TwoDShape	0.520000	5	5	0

4.2 Estimation Class Logical Stability using Regression

Thirty design metrics and actual class logical stability from the table 1 is used to estimate new class logical stability by using Multi Regression analysis. SPSS program [13] is used for calculation new estimation class logical stability.

4.3 Estimation Class Logical Stability using ANN

The Thirty design metrics diagram metric and actual class logical stability from the table 1 is used to estimate new class logical stability by using Multi Layer Perceptron method. WEKA program [10] is used for calculate new estimation class logical stability.

4.4 Evaluation

The proposed technique is validated using MMRE and PRED(0.25). The experiments result shown that MMRE of ANN is 78.81 % and MMRE of Regression is 70.28 %. It shown that Artificial Neural Network is effective in the estimation. The prediction value in general case is acceptable when a PRED(0.25) is at least 0.75 [14].

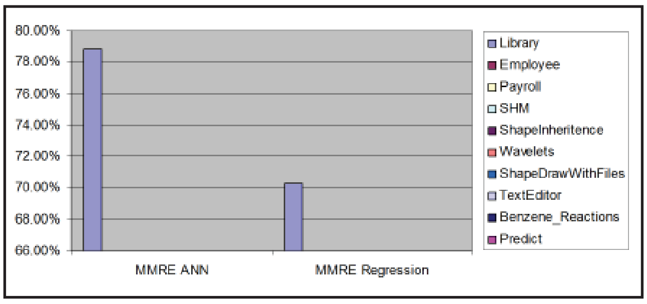

***Figure 5** Comparison of MMRE Multiple Regression and ANN.*

5. CONCLUSION

This study research is aimed to estimate the class logical stability using Artificial Neural Network (ANN) from class and sequence diagrams. This approach contributes to help software developers estimate software logical stability using the ANN algorithm from class diagram and sequence diagram. In addition, the approach helps developers in the design phase when they want to develop new software by understanding class logical ripple effect at an early phase.

This research has compared the actual class logical stability, an estimated class logical stability using multiple regression and an estimated class logical stability using ANN. The experiments result shows that MMRE of ANN is 21.18 % and MMRE of Multiple Regression is 29.72 %. It shows that Artificial Neural Network is effective in the estimation. With this result, the estimation of class logical stability is accepted because the prediction value at level 0.25 is more than 0.75.

This research is focus only logical ripple effect. There are two ripple effects that are occurred in software maintenance. We can expand our research to software performance ripple effect in the future.

6. REFERENCES

[1] S. Yau, and J. Collofello, "Some Stability Measures for Software Maintenance", *IEEE Transactions on Software Engineering*, vol. SE-6, no. 6, pp. 545-552, Nov. 1980.

[2] S. S. Yau, J. S. Collofello, and T. MacGregor, "Ripple effect analysis of software maintenance," *Computer Software and Applications Conference*, COMPSAC '78. IEEE, pp. 60-65, 1978.

[3] M. O. Elsih, and Rine,"Investigation of metrics for object-oriented design logical stability," *Proceeding of the Seventh European Conference on Software Maintenance and Reengineering*, 2003.

[4] S. Rangsiyawath and P. Muenchaisri, "Estimating software logical stability from class diagram and sequence diagram," *International Joint Conference on Computer Science & Software Engineering*, 2007.

[5] C. Cheewaviriyanon, and P. Muenchaisri, "Estimating software logical stability using analogy from class diagram and sequence diagram," *The 6th International Joint*

Conference on Computer Science and Software Engineering, 2009.

[6] I.F. Barcelos Tronto, J.D. Simoes da Silva, and N. Sant'Anna, "Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation," *International Joint conference on Neural Networks and IJCNN*, IEEE, pp. 771-776, 2007.

[7] J. Hakkarainen, P. Laamanen, and R. Rask, "Neural Networks in Specification Level Software Size Estimation," *Proceeding of the Twenty-Sixth Hawaii International Conference*, pp. 626-634, 1993.

[8] Magic Draw UML, Online at: http://www.magicdraw.com.

[9] SDMetrics, http://www.sdmetrics.com.

[10] WEKA, http://www.cs.waikato.ac.nz/ml/weka/.

[11] S.D. Conte, H.E. Dunsmore, and V.Y Shen, "Software engineering metrics and models," Benjamin/Cummings Publishing Company, 1986.

[12] T. Daniel Larose, "Data Mining Methods and Models," John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.

[13] Java Source Code, http://www.sourceforge.net.

[14] SPSS for Windows, http://www.spss.com.