Predicting Faulty Classes using Design Metrics with Discriminant Analysis

Mathupayas Thongmak[1] and Pornsiri Muenchaisri[2]
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University, Thailand
Telephone No. +66 (2) 2186988
E-mail: Mathupayas@hotmail.com[1], Pornsiri.Mu@chula.ac.th[2]

Abstract

Nowadays risk assessment is one of software engineering processes that plays important role in software development life cycle. Applying risk assessment to software the earlier is the better. Developers should detect defects of software early at design phase so the improvement action such as refactoring can be taken. Constructing fault prediction model using design metrics is one approach that can help developers to identify the faulty classes at early phase. This paper collects object-oriented design metrics and introduces some new metrics that tend to affect the existing of faults in classes, then construct the fault prediction model with discriminant analysis technique. The prediction model was trained by data collected from sale system and was validated using data from CD-selection system. The result indicates that 12 of 14 design metrics are associated with fault-proneness and the model can be used to classify faulty level of new classes.

Keywords: Design metrics, Prediction model, Reliability, Discriminant analysis

1. Introduction

One of important software engineering processes is risk assessment. Faulty classes prediction is a type of risk assessment that plays important role in software development life cycle. Focusing on fault detection in classes can help developers better allocating testing resources or redesigning classes that are likely to cause failures.

One approach to discover flaws in classes at design phase is to construct a prediction model using object-oriented design metrics. Khale El Emam, Walcelio Melo and Javam C. Machado introduced a study that used object-oriented design metrics to construct the fault prediction model using logistic regression technique [2]. Fault-proneness prediction model built using multivariate adaptive regression splines technique was presented by Lionel C. Briand, Walcelio L. Melo and Jurgen Wust [7]. Previous prediction models can classify faulty level into only two groups: no fault and has faults. None of these studies proposed fault prediction model using discriminant analysis in which more than two levels of faulty can be classified.

The objective of the paper is to collect object-oriented design metrics, to propose some new metrics that may be related to fault-proneness, to show the processes of constructing fault prediction model that can predict multilevel of faults using discriminant analysis technique and to present the prediction model.

A number of object-oriented design metrics considered in this paper consist of two sets of metrics which are object-oriented design metrics proposed by Yongjun Zhang [4], and by Hyoseob Kim and Cornelia Boldyreff, and 8 proposed new metrics that are expected to be related to fault-proneness. After collecting metrics, the model is developed by using training data set from one system. The discriminant analysis technique is used to construct the model. The constructed model is validated by using it to identify possible defect classes in the same and in different applications.

The next section discusses the design of the study by describing dependent variable, independent variables, and data analysis methods. Section 3 shows the details of the result. The paper ends with some conclusion and suggests future work to be carried out.

2. Design of the study

The design of the study consists of setting dependent variable or the thing that we are interested to predict, collecting independent variables that are likely to have influence on dependent variable, and defining the data analysis method.

2.1. Dependent Variable

On the assumption that designed models are equivalent to the implemented system, we focus on predicting faulty classes at design phase. The dependent variable is the number of faults in a class (0,1-2, and more than 2). Table 1 shows the definition of faulty class in object-oriented design adapted from [3].

Table 1. The definition of faulty classes in object-oriented design.

Faults	General Descriptions	Applied to OO design
Omission	Important information about the system has been omitted from the software artifact.	- Some methods defined in requirements have been neglected from designed diagrams. - Some necessary relationships have not been represented in designed diagrams. - Some essential attributes have been omitted. - Some vital methods have been cut off.
Incorrect Fact	Some information in the software artifact contradicts with information in requirement documents, general domain knowledge or related design models.	- Methods in designed diagram contrast with methods defined requirement specification. - Some methods are stored in inappropriate classes. - Types of attributes, relationships and cardinality defined are incorrect.
Inconsistency	Information within one part of the software artifact is inconsistent with other information in the software artifact.	- Functions in one designed diagram are contradicted with another in related diagram.
Ambiguity	Information within the software artifact is ambiguous.	- Attribute names and method names are unclear, and could cause a user of documents to misinterpret or misunderstand the meaning of the concept.
Extraneous Information	Provided information is not needed or used.	- Some attributes and methods are unused.

2.2. Independent Variables

Classification of independent variables shows in Table 2. Three of all metrics are collected from Chidamber & Kemerer Suite (WMC, DIT and NOC) [4]. Two of them are gathered from Hyoseob Kim and Cornelia Boldyreff (NMSC and NMRC) [5]. One metric that first was proposed by Chidamber & Kemerer but was changed to a normalization version by Hitz & Montazeri (LCOM) is proposed in [2]. The rest metrics are new proposed metrics that may affect the existing of faults (NOA, NOHA, NOTA, NOHC, NOTC, NOTG, NOATPB, and NOATPT).

Table 2. The group of independent variables.

Type of metrics	Object-oriented design metrics
Size measurement metrics	WMC, NOATPB, NOATPT
Coupling measurement metrics	NMSC, NMRC, NOA, NOHA, NOTA, NOHC, NOTC, NOTG
Inheritance measurement metrics	DIT, NOC
Cohesion measurement metric	LCOM

2.2.1. Size measurement metrics Weight method per class (WMC), number of attributes typed public (NOATPB) and number of attributes typed protected (NOATPT) are size measurement metrics. We consider size measurement metrics because recent study indicates that it is necessary to consider class size when constructing a fault prediction model [2].

- **WMC**: WMC is defined as the sum of complexity of a class. In this paper, we assume that complexity of every method is one unit so WMC is equivalent to the number of methods in a class.
- **NOATPB**: NOATPB is defined as the number of attributes of a class typed public.
- **NOATPT**: NOATPT is defined as the number of attributes typed protected.

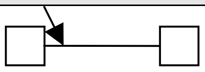
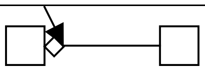
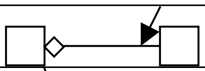
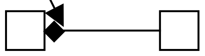
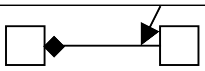
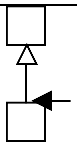
NOATPB and NOATPT are new metrics proposed because we suspect that different types of attributes may have impacts to the number of defects in classes.

2.2.2. Coupling measurement metrics Number of messages sent by the instantiated objects of a class (NMSC), number of messages received by the instantiated objects of a class (NMRC), number of associations (NOA), number of heads of aggregations (NOHA), number of tails of aggregations (NOTA), number of heads of compositions (NOHC), number of tails of compositions (NOTC), and number of tails of generalizations (NOTG) are coupling measurement metrics. Coupling between classes should be minimized because classes with high dependence tend to have more complexity than low dependence classes.

- **NMSC**: NMSC is defined as the sum of messages sent by objects instantiated from the class. This metrics can be used for finding out which classes are actively involved in interactions within a system.
- **NMRC**: NMRC is the number of messages received by created objects of the class. For example, Class1 has one instance of it, i.e. Object1, and Object1 receives 2 messages from Object2, thus NMRC value of Class1 should be 2.
- **NOA**: NOA is a count of association relationships of a class.
- **NOHA**: NOHA is a number of aggregation relationships that have hollow diamonds attached to a class.
- **NOTA**: NOTA is defined as the sum of aggregation relationships that have tail side closed to the class.
- **NOHC**: NOHC is the amount of solid diamonds bound with a class.
- **NOTC**: NOTC is defined as the counts of tails of compositions attached with the class.
- **NOTG**: NOTG is a count of generalization relationships that have tail side bound with a class.

Table 3 reveals the definition of metrics that we propose in the form of extracted pictures of class diagram.

Table 3. The proposed metrics definition.

Proposed metrics	Extracted pictures
NOA	
NOHA	
NOTA	
NOHC	
NOTC	
NOTG	

2.2.3. Inheritance measurement metrics Depth of inheritance tree (DIT) and number of children (NOC) are inheritance measurement metrics. Inheritance can affect the level of faults in classes since deeper trees constitute greater design complexity and the number of children related to the probability of improper abstraction of the parent class.

- **DIT**: DIT is defined as the maximum length from the node to the root of the tree. DIT measures how many ancestor classes can potentially affect a certain class. If DIT value is high, it is not easy to predict the behavior of the class.
- **NOC**: NOC is the number of immediate subclasses of a particular class. It measures how many classes can be affected by a particular class. The greater number of children a class has, the more testing required for the methods in that class.

2.2.4. Cohesion measurement metric Lack of cohesion in methods (LCOM) is cohesion measurement metric. High lack of cohesion is undesired because it indicates the improper design of a class abstraction. LCOM is defined as the cohesiveness of methods. Hitz and Montazeri presented a normalized version of LCOM [6]:

$$LCOM = \frac{2*|E|-(n-1)}{(n-1)(n-2)}.$$

|E| is the number of links and n is the number of methods. Links are defined as:

1. Association: one class references variables of another class.
2. Aggregation: the definition of one class involves objects of the other class.
3. Inheritance: one class inherits the features defined in another (parent) class.
4. Invocation: methods in one class invoke methods defined in another class.

2.3. Data analysis methods

We selected discriminant analysis as a data analysis method. Discriminant analysis is a technique used to categorize data [1]. The advantage of discriminant analysis over regression is that its predictive ability in terms of percent of correct classifications since it is used with categorical data.

Discriminant analysis consists of 5 steps as follows:

1. variable preparation,
2. sample selection,
3. information gathering,
4. prediction model construction, and
5. prediction model validation.

2.3.1. Variable preparation The objective of this step is to identify the object-oriented design metrics that can differentiate level of faults in classes (*No fault*/*Has one or two faults/ Has more than two faults*). These independent variables are then verified whether they should be included in the prediction model.

2.3.2. Sample selection The sample consists of training system and test system. Data collected from the training system is used as input for constructing a prediction model. Information gathered from the test system is used to validate the prediction model.

Sale system and CD-selection system are used as the training system and test system. The sale system provided support for sale process. The sales process starts when a customer orders products via sale person or company web-site. Then an operator will create purchase order and convert it to sale order. After that, the sale order will be sent to manager of each department to approve sale order. Finally, the operator will change the approved sale order to invoice and send it to shipping department. The sale system had a total of 41 classes. In total, 13 classes had no fault and 27 classes had one or two faults. We counted the number of faults according to faulty class definitions defined above.

The CD-selection system is selected to be the test system because CD-selection system is within the same domain with sale system. The CD-selection system supported sale CD process via Internet. The system provided sustained functions such as searching CD function, ordering CD function. For the CD-selection system, 15 of 23 classes had one or two faults and 7 classes had no fault.

2.3.3. Information gathering During this phase, we collect training data set and test data set to be input for further steps. The training data set composes of 14 object-oriented design metrics mentioned earlier and faulty level of classes collected from a training system. The test data set comprises of a set of metrics included in the predicting equation and faulty level collected from a test system. For each class, faulty level is characterized as *no fault* that is represented with zero or *has one or two faults* that is represented with one or *has more than two faults*, which is represented with two.

2.3.4. Prediction model construction In this phase, the training data set will be input into SPSS application to construct the prediction model. In this step, the independent variables would be checked whether they significantly relate to fault-proneness and whether they affect each other. If not, the metrics will be cut off by the program. The output from SPSS includes equations from Fisher's linear discriminant function. The number of discriminant functions is equal to the sum of classification groups. There are 3 equations as discriminant functions because we analyze and classify 3 groups of dependent variables, which are *no fault*, *has one or two faults* and *has more than two faults*. The general form of the equation is

$$D = a + b_1x_1 + b_2x_2 + \ldots + b_px_p,$$

where D is the discriminant score, a is a constant value, b_i is the classification function coefficient, x_i is discriminator variable or independent variable that is associated with fault-proneness and p is a number of discriminator variables.

Each equation is used to determine faulty level of each group. A class would be foretold to be group 0 or no fault if discriminant score calculated from the *no fault* equation is greater than the score computed from the *has one or two faults* and from the *has more than two faults* equations. The class would be predicted as group 1 or *has one or two faults* if the discriminant score estimated from the *has one or two faults* equation is more than the score calculated from the *no fault* and from the *has more than two faults* equations. On the other hand, the class would be predicted as group 2 or *has more than two faults* if the discriminant score computed from the *has more than two faults* equation is more than the score estimated from the *no fault* and from the *has one or two faults* equations.

2.3.5. Prediction model validation To evaluate the prediction model, we predict faulty level of each class and compare it with actual fault level. The following equation formulates the reliability of prediction model adapted from [2] by showing as the percentage of correct prediction:

$$A = 100 * ((p_1*h_1) + (p_2*h_2) + (p_3*h_3)),$$

where A is the percentage of prediction accuracy, h_1 is the proportion of *no fault* classes, h_2 is the proportion of *has one or two faults* classes, h_3 is the proportion of *has more than two fault* classes, p_1 is the proportion of *no fault* classes that are rightly classified as *no fault*, p_2 is the proportion of *has one or two faults* classes that are accurately classified as *has one or two faults*, and p_3 is the proportion of *has more than two faults* classes that are correctly predicted as *has more than two faults*.

3. Results

3.1. The result of variable preparation

Variable NOATPT has less than four observations that are non-zero so it was excluded from further analysis. Table 4 contains the result of variables failing tolerance test. Tolerance test is conducted to evaluate the independence of predictors. Whenever two independent variables overly correlated with each other, one of them must be eliminated. The result indicates that independent variables NOTG and NOATPT are failed after tolerance test. Therefore, 12 out of 14 metrics had a significant association with fault detection. Table 5 shows the result of variable validation by using standardized canonical discriminant function coefficients. The validation points that WMC, NOHC, NMRC, NOA, NOATPB, NOHA, NOTA, NOC, LCOM, NOTC, NMSC, and DIT have strong impact to weak impact on *has one or two faults* group classification respectively. In addition, NMRC, WMC, NOHA, NMSC, LCOM, NOC, NOA, NOTA, NOATPB, NOHC, DIT, and NOTC have strong influence to weak influence on *has more than two faults* group classification respectively.

Table 4. The result of variables failing tolerance test.

	Within-Groups Variance	Tolerance	Minimum Tolerance
NOTG	.167	.000	.000
NOATPT	.000	.000	.000

All variables passing the tolerance criteria are entered simultaneously.

a. Minimum tolerance level is .001.

Table 5. The result of variables validation.

Standardized Canonical Discriminant Function Coefficients

	Function	
	1	2
WMC	1.728	-2.354
LCOM	-.304	.764
DIT	.162	-.019
NOC	-.337	.662
NOA	.775	.432
NOHA	.443	1.117
NOTA	.356	-.432
NOHC	-1.514	-.062
NOTC	-.235	.004
NMSC	-.216	-.993
NMRC	-1.059	3.000
NOATPB	.593	-.242

3.2. The result of prediction model construction

Table 6 reveals Fisher's linear discriminant function in the form of the classification function coefficient. The coefficient forms 3 equations; the *no fault* equation, the *has one or two faults* equation, and the *has more than two faults* equation. Classes are classified to be *no fault* or *have one or two faults* or *have more than two faults* up to which equations have greatest discriminant score. The *no fault* equation is

$$D = -3.644 - 1.311WMC + 2.633*10^{-8}LCOM + 3.9070DIT + 1.190NOC - 0.190NOA - 1.514NOHA + 1.890NOTA + 3.570NOHC + 1.975NOTC - 2.247NMSC + 1.720NMRC + 0.179NOATPB,$$

the *has one or two faults* equation is

$$D = -4.284 + 0.489WMC + 1.481*10^{-8}LCOM + 4.654DIT - 0.271NOC + 0.837NOA - 0.352NOHA + 2.811NOTA - 0.238NOHC + 1.44NOTC - 2.843NMSC + 0.698NMRC + 0.563NOATPB.$$

the *has more than two faults* equation is

$$D = -8.45 - 4.027WMC + 6.499*10^{-8}LCOM + 4.308DIT + 3.467NOC + 1.526NOA + 4.375NOHA + 0.725NOTA + 0.515NOHC + 1.596NOTC - 7.498NMSC + 5.661NMRC + 0.2NOATPB.$$

Table 6. The classification function coefficient from Fisher's linear discriminant analysis.

	Faulty Level		
	.00	1.00	2.00
WMC	-1.311	.489	-4.027
LCOM	2.633E-08	1.481E-08	6.499E-08
DIT	3.907	4.654	4.308
NOC	1.190	.271	3.467
NOA	-.190	.837	1.526
NOHA	-1.514	-.352	4.375
NOTA	1.890	2.811	.725
NOHC	3.570	-.238	.515
NOTC	1.975	1.440	1.596
NMSC	-2.247	-2.843	-7.498
NMRC	1.720	.698	5.661
NOATPB	.179	.563	.200
(Constant)	-3.644	-4.284	-8.450

3.3. The result of prediction model validation

We used the model above to predict which classes in the test system may have faults. Test data from CD-selection system is used to validate reliability of the prediction model. Table 7 shows the results of prediction. Then the percentage of prediction accuracy calculated according to formula that is proposed in Section 2.4.5 would be approximately 65.22.

3.4. Discussion of results

The result indicates that metrics WMC, LCOM, DIT, NOC, NOA, NOHA, NOTA, NOHC, NOTC, NMSC, NMRC, and NOATPB are associated with defects in classes. The advantage of these metrics is that they can be collected at design time. So after combining these metrics in a prediction model, we can predict which classes may have faults at design phase.

Table 7. Prediction results of CD-selection system.

		Predicted faulty level			
		No fault	Has one or two faults	Has more than two faults	
Actual faulty level	No fault	2	4	1	7
	Has one or two faults	2	12	1	15
	Has more than two faults	0	0	1	1
		4	16	3	23

3.4.1. Relationship between size measurement metrics and fault in classes Two-Third size measurement metrics have impact to the number of faults in classes. Both WMC and NOATPB have strong impact on fault-proneness due to two possible reasons:

1. The larger number of methods in a class, the greater potential defects in classes.
2. Attributes typed public can be accessed by any other classes. So the greater number of attributes typed public, the greater possible misuse of attributes from other classes.

3.5.2. Relationship between coupling measurement metrics and fault in classes Seven-Eighth of coupling measurement metrics are related to defects in classes. Seven metrics consist of NMSC, NMRC, NOA, NOHA, NOTA, NOHC, and NOTC. Most of coupling metrics affect fault-proneness because if a class often use methods of others, it may be a case of miscall of methods and if a class is much used by other classes, it has to implement more complex methods to support others. Several possible reasons why these metrics associated with fault-proneness are:

1. The number of messages sent by instantiated objects of a class gives an idea of import coupling. The amount of calling methods of other classes reveals the possibility of incorrect method calls.
2. The number of messages received by instantiated objects of a class shows the export coupling. If a class has several export couplings, it may has faults within a class due to the complexity of methods of the class that have to design to sustain other classes.
3. Association between classes means two classes associated each other. After designing class diagram in details, association relationships can be changed to be composition or aggregation relationships. The more associations or dependents a class has, the harder it is to be implemented without faults.
4. The greater number of heads of aggregations and compositions, the greater likelihood of wrong implementation of method calls within classes because classes next to heads of aggregations and compositions tend to frequently call methods of other classes.
5. The higher amount of tails of aggregations and compositions, the more possibility of defects in classes because classes next to tails of aggregations and compositions may often be used by other classes.

3.5.3. Relationship between inheritance measurement metrics and fault in classes Both DIT and NOC affect the potential of faults within a class because of:

1. Deeper trees constitute greater design complexity since there are more methods a class can inherit. If there are greater number of DIT, it is difficult to predict the class behavior.
2. The greater number of children, the greater the possibility of improper abstraction of the parent class.

3.5.4. Relationship between cohesion measurement metric and fault in classes LCOM is associated with fault-proneness. Cohesiveness of methods within a class is desirable. Low cohesion raises the likelihood of errors in a class since it increases the complexity of the class.

3.5.5. Limitations This study has two limitations that should be considered in the analysis of the results:

1. The training system: The amount of classes within the training system is only 41 and the number of design metrics included in the study is quite large. So some metrics such as NOATPT that may affect faults of class has been cut off because it has less than four observations that are non-zero. We suspect that NOATPT metric could be added in the prediction model if we increase the size of the sale system.
2. The system to be test: Although CD-selection system is within the same domain with sale system, the system still differs from the training system in the details of business rules. We suspect that if the

prediction model was used to predict another version of sale system than CD-selection system, the percentage of prediction accuracy from the study would be increased.

4. Conclusion and Future Work

This paper collects object-oriented design metrics and introduces some new metrics that likely to influence on faults in classes, then validates the relationship of these metrics with fault-proneness by constructing the prediction model, which can classify 3 level of faults, using discriminant analysis. The data gathered from sale system and CD-selection system are used as input of training and testing processes respectively. Faulty level prediction model introduced after following discriminant analysis steps are then estimated the reliability of model by calculating the percentage of percent accuracy. Our results indicate that WMC, LCOM, DIT, NOC, NOA, NOHA, NOTA, NOHC, NOTC, NMSC, NMRC, and NOATPB metrics are affected the faulty level of classes prediction. Furthermore, the prediction model constructed with these metrics has fair prediction accuracy. The percentage of correct classification by prediction faulty level of classes in CD-selection system is 65.22.

In the future, the prediction model should be validated with the different version of sale system. More metrics should be added on the assumption that they may increase the percentage of prediction accuracy. Also, the size of training system should be expanded.

References

[1] Kalaya Wanichbuncha. Mutivaribles Analysis using SPSS for Windows. Chulalongkorn University, 2001.

[2] Khaled El Emam, Walcelio Melo and Javam C. Machado. "The prediction of faulty classes using object-oriented design metrics", The Journal of Systems and Software No.56, pp. 63-75, 2001.

[3] Guilherme H. Travassos, Forrest Shull, Michael Fredericks and Victor R. Basili. "Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality", Conference on Object-Oriented Programming, Systems, Languages, and Application(OOPSLA), pp. 47-56, 1999.

[4] Yongjun Zhang. "Metrics for Object-oriented Design", http://www.soberit.hut.fi/~tony/seminaari/reports/yongjun_zhang.doc.

[5] Hyoseob Kim and Cornelia Boldreff. "Developing Software Metrics Applicable to UML Models", 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), 2002.

[6] Hind Kabaili, Rudolf K. Keller and Francois Lustman. "Cohesion as Changeability Indicator in Object-Oriented System", Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR 2001), pp. 39-46, 2001.

[7] Lionel C. Briand, Walcelio L. Melo, and Jurgen Wust. "Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects", IEEE Transactions on Software Engineering Vol. 28 No.7, 2002.