EVALUATING STRUCTURAL VALIDITY OF UML CLASS DIAGRAMS BY MEASURING THE NUMBER OF HIGHLY RESPONSIBLE CLASSES

ABSTRACT

Design models are often developed using UML class diagrams. In past questionnaire surveys, Lange and Nugroho reported that designers tend to write the important or complex parts of a design model in detail. Thus, we presume that a design class diagram in which some parts are more important will contain classes with both high and low responsibility. Moreover, we hypothesize that a design class diagram containing few highly responsible classes has low validity.

In this study, we calculated four basic class metrics (the number of attributes (NAttr), operations, associations and subclasses) and our novel metrics (e.g. the number of larger values of NAttr). All metrics were calculated using 65 design class diagrams, which were originally submitted to a Robot Contest on the domain of embedded systems and evaluated by software development experts based on the understandability of the system, adequacy of responsibility assignment, etc. Then the relations between our metrics and the experts' qualitative assessment were analyzed. Consequently, the usefulness of our metrics and our hypothesis are confirmed empirically.

KEY WORDS

Software Metrics, Software Design and Development, UML Class Diagram, Responsibility Assignment

1 Introduction

At software development using design models, quality of source code becomes high[14] and the code change-proneness becomes low.[17] To improve design quality, qualitative guides such as design principles are used. However, these guides are subjective and the judgement standards are ambiguous. Thus, quantitative guides for specific features are desirable.

Table 1. Basic class metrics indicating size of responsibility

Abbrv	Class Metrics Description
NChld	The Number of Subclasses (Owned Children)
NAttr	The Number of Attributes (Owned)
NOp	The Number of Operations (Owned)
NAssoc	The Number of Associations (Owned)

Design models are often developed using UML class diagrams [15], which have quantitative features (i.e., the numbers of attributes, associations etc.). The relations between features and software quality have been studied (e.g., the relation between total number of associations in a class diagram and software maintainability)[3] [5] [6] [4] [11].

These findings about the total of class metric values are useful for assessing the quality of a design class diagram. However, the total value can not indicate the proportion of metric values, while findings about the proportion of metrics values is more useful when creating or modifying a class diagram (i.e., assigning attributes, operations, associations etc.). For example, is it better to have associations concentrated in a few classes or to have associations evenly assigned to many classes?

In past questionnaire surveys, Lange[8] and Nugroho[13] reported that designers tend to write the important or complex parts of a design model in detail. From this, we presume that a design class diagram in which some parts are more important will contain classes with both high and low responsibility. Consequently, we construct the following hypothesis about responsibility assignment.

- Hypothesis: A design class diagram containing few highly responsible classes has low validity.

A design class diagram is generally omitted some elements[15]. We assume that a class diagram containing few highly responsible classes is abbreviated and not clarified important parts because the class diagram is not properly considered. Additionally, we assume that conventional values, such as the total or mean of class metric values, can not indicate the proportion of metric values. Thus, to confirm the above hypothesis quantitatively, we propose novel metrics derived from a set of another metric values.

- NL: Number of Larger values.
- RL: Ratio of Larger values.

Additionally, tendency of some class metric values probably differ among diagrams because the tendency to abbreviate or describe in detail varies among the diagrams. Thus, we propose two methods to distinguish larger values, and Figure 1 shows the concept of the methods.

- C_D: Compare metric values from multiple models in the same domain.
- C_M: Compare metric values from a single model.

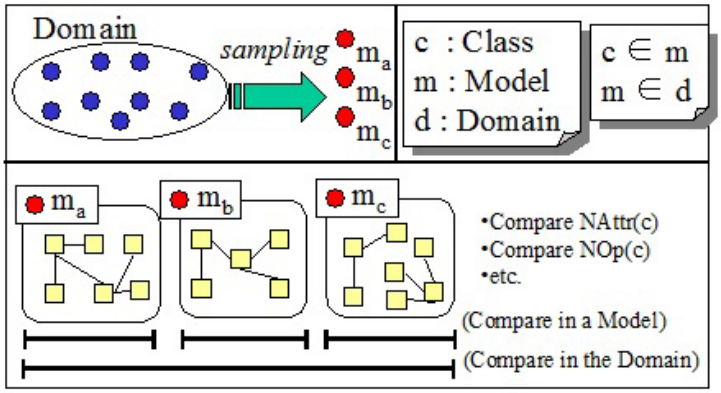

Figure 1. The concept to distinguish larger values: Comparing metric values from multiple models in the same domain or from a single model

Table 2. Our four novel metrics

	NL	RL
C_D	NL_D	RL_D
C_M	NL_M	RL_M

Consequently, we propose four novel metrics shown in Table 2 to indicate the proportion of metric values.

We set RQ1 as research question to confirm usefulness of our metrics, while RQ2 was set to verify our hypothesis about the responsibility assignment.

- RQ1: Is there a relation between our metrics and the validity of design class diagrams?
- RQ2: Dose a design class diagram containing few highly responsible classes have low validity?

In this study, we calculated four basic class metrics shown in Table 1, and calculated our metrics values (e.g., NL_D of NAttr). All metrics were calculated using 65 design class diagrams, which were originally submitted to a Robot Contest [7] on the domain of embedded systems and evaluated by software development experts based on the understandability of the system, adequacy of responsibility assignment, etc. Then the relations between our metrics and the experts' qualitative assessment were analyzed. Consequently, the usefulness of our metrics and our hypothesis are confirmed empirically. Finally, contributions in this study are the following.

- We provided novel viewpoints (C_D and C_M) to distinguish larger metric values in a class diagram.
- We defined metrics to indicate the proportion of metric values for UML design class diagrams.
- We empirically analyzed the feature of responsibility assignment using our metrics.

2 Background

2.1 Size of Responsibility Assigned to a Class

There are two types of responsibilities: Responsibility of Knowledge and Behavior [10][2]. Four basic class metrics shown in Table 1 indicate the size of responsibility assigned to a class. NAttr, NChld and NAssoc indicate the degree of Responsibility of Knowledge (e.g. size information of the encapsulated data, related objects, etc.), while NOp indicates the degree of Responsibility of Behavior. Consequently, focusing on these metrics can detect highly responsible classes.

2.2 Motivating Example

Figures 3-5 are parts of UML design class diagrams (the original diagrams of them are written in Japanese). These diagrams are products submitted to a Japanese contest on the domain of embedded systems "ET Robot Contest 2010" (ET RoboCon) [7]. In the contest, participants let uniform robots run along a black line on a white stage (Fig. 2 shows), and the robots have same body (hardware) and different software.

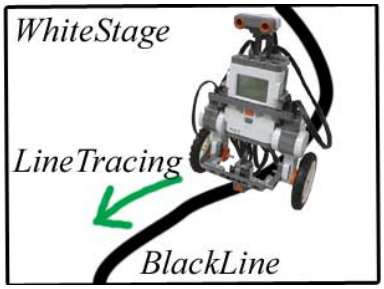

Figure 2. The robot running along a black line on a white stage

Figs. 3-5 are design models of Calibration in different level of detail. Calibration is a mechanism to reduce the error rate of discriminating black and white, and it is used because the robots run along the black line (line-trace) using a light sensor which measures brightness of a robot's underfoot. The light sensor sample the brightness from multiple points (black or white) to discriminate, and then a touch sensor is often used to determine the timing of the sampling.

Fig. 3 is a part of a model extracted from the class diagram whose structure was assessed as high quality (B) by software development experts while Figs. 4 and 5 are extracted from low quality (C) class diagrams. In Fig. 3, important classes to calibrate was specified and only they are detailed (have many attributes and operations). Fig. 3 contains necessary responsibility (attributes and operations) to calibrate while Figs. 4 and 5 lack them.

In past questionnaire surveys, Lange[8] and Nugroho[13] reported that designers tend to write the

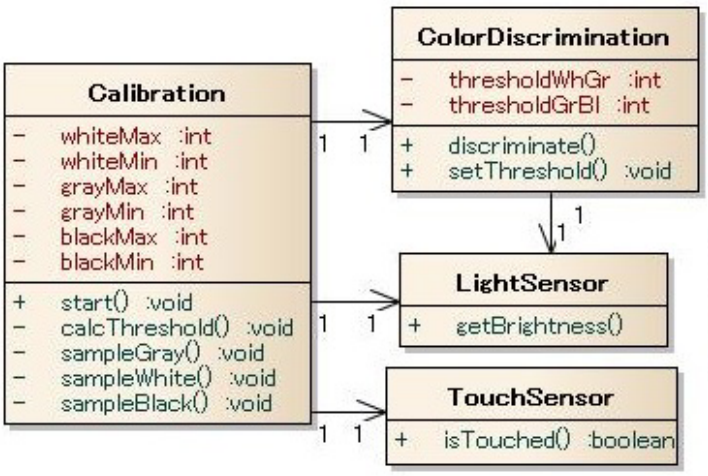

Figure 3. Calibration model in design class diagram (Diagram 1) submitted to the ET RoboCon 2010, Structure-Assessment(B)

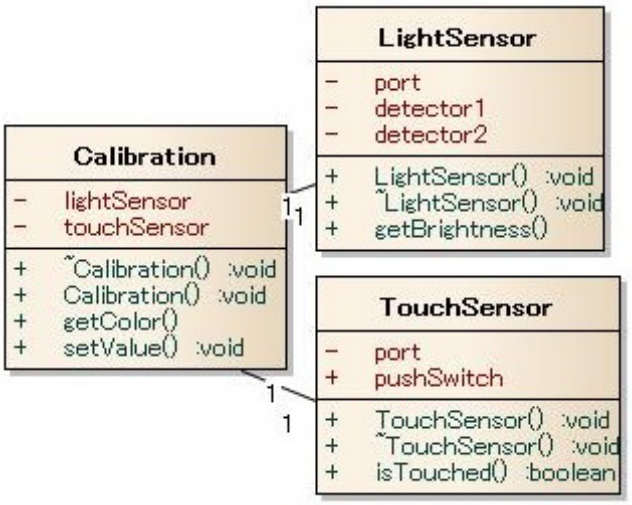

Figure 4. Calibration model in design class diagram (Diagram 2) submitted to the ET RoboCon 2010, Structure-Assessment(C)

important or complex parts of a design model in detail. From this, we presume that a design class diagram in which some parts are more important will contain classes with both high and low responsibility.

Details of Figs. 3-5 are described below. In Fig. 3, two sensor classes are written simply while Calibration and ColorDiscrimination are detailed. ColorDiscrimination is a class to discriminate a white stage or a black line, on which the robot stay. The color is discriminated using thresholds derived from multiple brightness values which are sampled by LightSensor via Calibration. Additionally, Fig. 3 indicate the necessity to save Max and Min brightness values of white, gray and black when deriving thresholds.

In Fig. 4, all classes are detailed however almost all operations are constructor and destructor, namely Fig. 4 contains less responsibility (attributes and operations) to discriminate color than Fig. 3.

In Fig. 5, all classes are not detailed and have only basic operations (e.g., getBrightness()).

2.3 Deriving Threshold by Alves' Method

In this section, Alves' method is described and it is used by our metrics described in section 3.1.

Preparing a threshold for a metric, which is one way to distinguish larger values. If multiple class diagrams representing specific domain are available, Alves ' method can derive a metric thresholds [1].

$$A_i = \{a_{i,1}, ..., a_{i,ni}\} \quad (1)$$

$$T(q) = \{t_1, ..., t_k | t_i = Quantile(A_i, q)\} \quad (2)$$

$$th(q) = Median(T(q)) \quad (3)$$

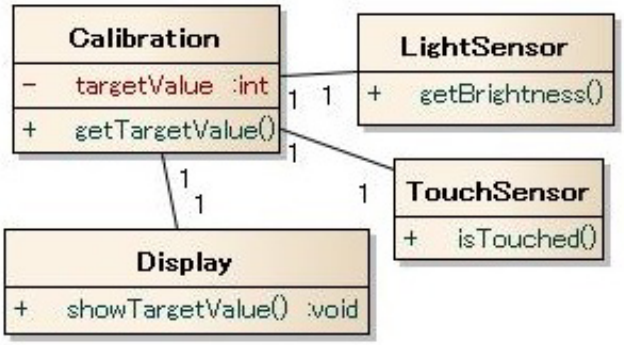

Figure 5. Calibration model in design class diagram (Diagram 3) submitted to the ET RoboCon 2010, Structure-Assessment(C)

As an example, the following shows a process to derive thresholds for NAssoc. Let A_1 (A_i) be a set of values of class' NAssoc in $Diagram_1$($Diagram_i$), and let t_1 (t_i) be the quantile of A_1 (A_i) against a real number q ($0 \leq q \leq 1$). When q=0.70 , t_1 (t_i) becomes 70% quantile of A_1 (A_i). Then the threshold is the median of { t_1, t_2, .., t_i }.

3 Our Metrics

In past questionnaire surveys, Lange[8] and Nugroho[13] reported that designers tend to write the important or complex parts of a design model in detail. From this, we presume that a design class diagram in which some parts are more important will contain classes with both high and low responsibility. Thus, we assume that designer should focus on the proportion of metric values.

Additionally, we assume that conventional values (i.e., Total, Mean, Max, etc.) can not indicate the proportion of metric values. Thus, we propose novel metrics derived from a set of another metric values.

- NL: Number of Larger values.
- RL: Ratio of Larger values.

Additionally, tendency of some class metric values probably differ among diagrams because the tendency to abbreviate or describe in detail varies among the diagrams. Thus, we propose two methods to distinguish larger values, and Figure 1 shows the concept of the methods.

- C_D: Compare metric values from multiple models in the same domain.
- C_M: Compare metric values from a single model.

The C_D uses the threshold derived by Alves' method (described in section 2.3) while the C_M uses the threshold derived by the quantile. Consequently, we propose four novel metrics shown in Table 2 to indicate the proportion of metric values. It is assumed that the hypothesis in RQ2 can be verified by statistical analyses between our metrics and the expert's qualitative assessment against a class diagram.

3.1 Our C_D Metrics (NL_D and RL_D)

Figure 6 illustrates image of our metrics NL_D70 (described later). Details of our metrics are described below.

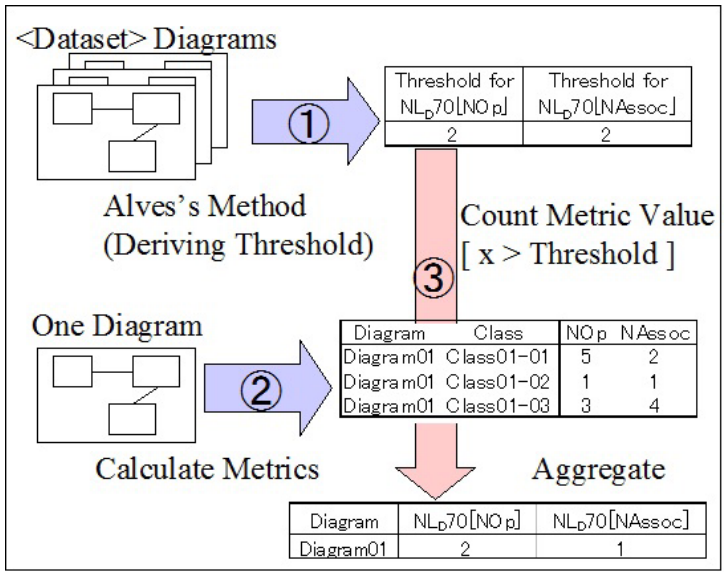

Figure 6. Process to Calculate NL_D70, Using Alves' Method to Derive the Threshold in the Domain

Conveniently writing the threshold at q=0.70 as $th70$; in the Alves' method, values $< th70$ are normal, $th70 <$ values $\leq th80$ are slightly large, $th80 <$ values $\leq th90$ are large, $th90 <$ values are very large.

Using the threshold derived by Alves's method, we propose two novel metrics NL_D (Number of Larger values in the Domain) and RL_D (Ratio of Larger values in the Domain). NL_D is the number of classes containing larger metric value, and RL_D is the percentage of these classes in the class diagram. NL_D count larger values in a set of metric values using the threshold prepared specifically for the domain while RL_D calculates the ratio of the larger values in the set. In this study, NL_D70 (RL_D70) is NL_D (RL_D) calculated at $q = 0.70$.

$$X = \{x_1, ..., x_n\} \quad (4)$$

$$NL_D(X, th) = |\{x \in X | x > th\}| \quad (5)$$

$$RL_D(X, th) = \frac{NL_D(X, th)}{|X|} \quad (6)$$

$$th = Predetermined\ \ Threshold \quad (7)$$

As an example, the calculation processes of NL_D70 of NAssoc are described below. Using Alves' method, $th70$ of NAssoc is derived from a dataset of class diagrams representing a specific domain. Let X be a set of NAssoc values in a class diagram. The number of x_i greater than the $th70$ becomes the NL_D70 of NAssoc value of the class diagram.

3.2 Our C_M Metrics (NL_M and RL_M)

NL_D and RL_D are calculated using a predetermined threshold. However, the tendency for the metric value distribution depends on the utilization purpose. Hence, we propose two novel metrics, NL_M (Number of Larger values in a Model) and RL_M(Ratio of Larger values in a Model), which can be calculated without a predetermined threshold. NL_M is the number of classes containing a larger metric value while RL_M is the percentage of these classes in the class diagram.

$$X = \{x_1, ..., x_n\} \quad (8)$$

$$NL_M(X, th) = |\{x \in X | x > th\}| \quad (9)$$

$$RL_M(X, th) = \frac{NL_M(X, th)}{|X|} \quad (10)$$

$$th = Quantile(X, q) \quad (11)$$

As an example, Let X be a set of NAssoc values from a single class diagram, and let $th70$ be the 70% quantile of X. At a glance, the number of x_i grater than $th70$ becomes $|X| * 0.3$ however it is not necessarily so. Giving a counterexample, when $X = \{2, 2, 2, 2\}$, $th70$ becomes 2 and all x_i are not grater than $th70$, and consequently the NL_M value becomes 0.

The NL_M value is not necessarily equal to the number of classes * 0.3. Consequently, NL_M is meaningful because it can indicate the proportion of metric values. Additionally, these discussion is summarized in Fig. 7.

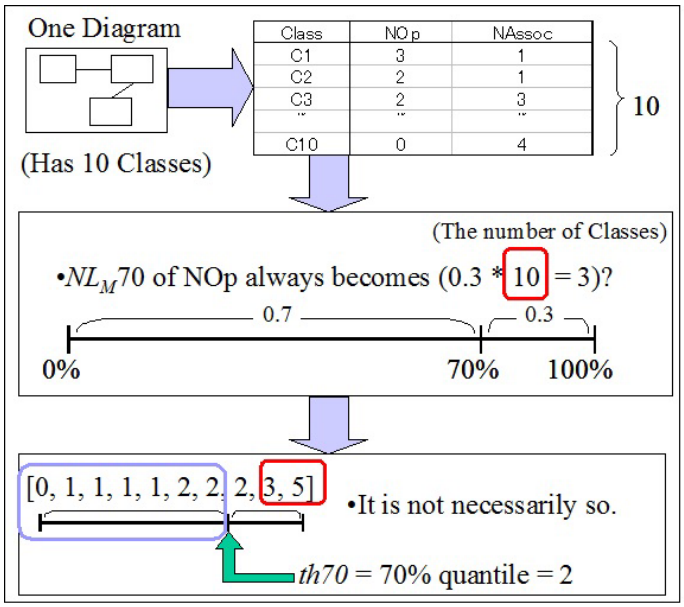

Figure 7. NL_M70 is meaningful because it is not always equal to the number of classes * 0.3.

The threshold at q=0.70 is conveniently written as $th70$. As well as the Alves' method, values $< th70$ are normal, $th70 <$ values $\leq th80$ are slightly large, $th80 <$ values $\leq th90$ are large, $th90 <$ values are very large. NL_M and RL_M are calculated in the same process of NL_D and RL_D except the process to determine the threshold.

3.3 Apply to Motivating Example

Figures 3-5 are parts of UML design class diagrams (Diagram 1-3). Tables 3 and 4 show NL_D and NL_M of NAttr calculated from Diagram 1-3, and values of NL_D were calculated using thresholds shown in Table 6. Details of the dataset to derive the thresholds are described in section 4.2.

Because Diagram 1 was received the higher assessment than Diagrams 2 and 3, NL_M70 and NL_M80 of NAttr will be positively correlated with the qualitative assessment. A large value of NL_M of NAttr indicates that the class diagram contains multiple classes with higher NAttr and many classes with less NAttr. This result correspond to our hypothesis.

Table 3. NL_D of NAttr calculated from Diagram1-3 (all parts of Figs 3-5)

Diagram	NL_D70	NL_D80	NL_D90
Diagram1	5	1	1
Diagram2	8	5	2
Diagram3	2	0	0

Table 4. NL_M of NAttr calculated from Diagram1-3 (all parts of Figs 3-5)

Diagram	NL_M70	NL_M80	NL_M90
Diagram1	5	5	1
Diagram2	2	2	2
Diagram3	2	2	2

4 Evaluation Experiment

4.1 Experiment

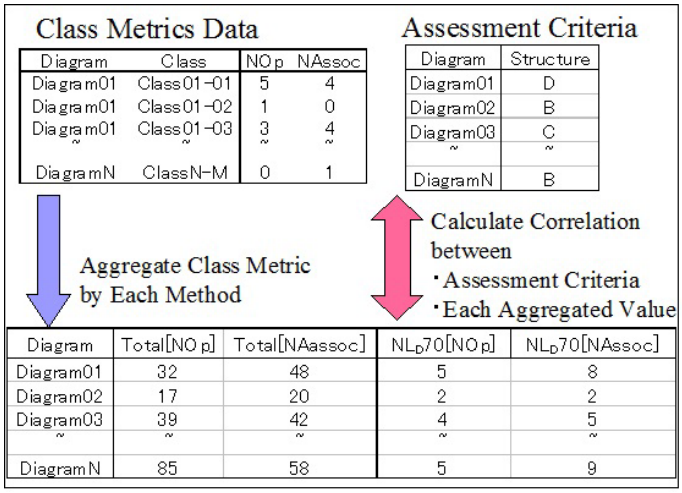

Figure 8. Experiment to calculate metrics values, aggregate metric values to investigate the correlation between the qualitative assessment score

Table 1 shows the four basic class metrics that indicate the size of responsibility assigned to class. Using automatic measurement tools (we developed), we measured these class metrics using the 65 design class diagrams. Additionally, we measured our metrics values (i.e., NL_D70 of NAttr, NL_D70 of NOp etc.).

After the measurement, we investigated Spearman's rank correlation between aggregated value (e.g. our metrics) and the qualitative assessment against a class diagram. For comparison, we calculated the aggregated values in multiple ways: Total, Mean, Median, Max, Min, NL_D, RL_D, NL_M and RL_M with $th70$, $th80$ and $th90$.

Additionally, we derived the thresholds for NL_D and RL_D from the dataset (all 65 diagrams), and prepared the thresholds for each metrics (e.g., a threshold for NL_D70[NAssoc]). Figure 8 depicts the experimental process, and the details of experimental data are described in Section 4.2.

4.2 Experimental Data

The dataset includes 65 design class diagrams submitted to the Japanese software development contest "ET Robot Contest 2010" [7], which aimed to improve technical education of embedded systems. To design an autonomous robot, the class diagrams were written by students and working adults with software development experience In the contest, software development experts qualitatively assessed the submitted models. Because the ratings are assigned on an ordinal scale (A-D) after multiple experts consult a checklist, we assume that the ratings are valid.

The qualitative assessment of the experts consists of multiple criterion, and some criterion target the design models besides class diagrams. Therefore, we extracted the criteria "Structsure", which is strongly related to design class diagram, and used it for analyses. The rank of "Structure" (A-D) indicate degree of understandability of system, adequacy of responsibility assignment etc.

4.3 Results

Table 5 shows the size of Spearman's rank correlation between the aggregated value and qualitative assessment as an ordinal scale. "p" denotes the rank correlation is insignificant because p-value > 0.05, and "NA" means that rank correlation could not be determined. Table 6 show the thresholds of four metrics derived by Alves' method for NL_D and RL_D. The thresholds for NChld are 0 because almost all classes did not have a subclass (NChld=0).

Table 5. Rank Correlation with the "Structure" Assessment

	NChld	NAttr	NOp	NAssoc
Total	p	0.33	0.31	0.41
Mean	p	p	p	p
Median	NA	p	p	p
Min	NA	p	p	p
Max	p	p	0.30	p
NL_D70	p	0.33	p	0.53
NL_D80	p	p	p	0.31
NL_D90	p	p	p	0.31
RL_D70	p	p	p	0.42
RL_D80	p	p	p	p
RL_D90	p	p	p	p
NL_M70	p	0.38	0.37	0.25
NL_M80	p	0.41	0.35	p
NL_M90	p	0.43	0.36	p
RL_M70	p	p	p	p
RL_M80	p	p	p	p
RL_M90	p	p	p	p

Table 6. Thresholds of the four basic metrics derived by Alves' method for NL_D and RL_D

	NChld	NAttr	NOp	NAssoc
$th70$ for NL_D70	0	1	2	2
$th80$ for NL_D80	0	2	2	3
$th90$ for NL_D90	0	3	2.9	3.7

5 Discussion

5.1 RQ1: Is there a relation between our metrics and the validity of a design class diagrams?

Some of our metrics are significantly correrated with the expert's qualitative assessment which indicate the validity of design. NL_M is useful to analyze the proportion of class metrics (NAttr and NOp) that are affected by designer's characteristic. Conversely, NL_D is useful to analyze the proportion of other class metrics (NAssoc).

5.1.1 Discussion about NL_D

Table 5 shows that NL_D of NAssoc is significantly correlated with the qualitative assessment. Class diagrams containing more classes with NAssoc $\geq 3(or4)$ (Table 6) received a higher assessment.

In a class diagram, placed elements as attributes, operations and classes are often either omitted or detailed. However, the associations between already placed classes are less omitted. Thus, NL_D of NAssoc is correlated to the qualitative assessment, while NL_M of NAssoc is not.

5.1.2 Discussion about NL_M

Table 5 shows that NL_M of NAttr and NL_M of NOp are significantly correlated to the qualitative assessment, but RL_M of NAttr and RL_M of NOp are not. In a single design class diagram, a diagram receives a higher assessment when more classes have many attributes or operations. This result shows that disproportions of NAttr and NOp in the diagram are desirable.

In a class diagram, attributes and operations are often omitted or detailed, but whether they are omitted or detailed differs among the class diagrams due to the designer ' s characteristic Therefore, experts assess a class diagram by partly by intercomparing NAttr and NOp in a single diagram. Thus, NL_M of NAttr (NL_M of NOp) is significantly correlated to the qualitative assessment.

5.1.3 Comparison of Our Metrics and Conventional Aggregated Values

Total, but not Mean, is significantly correlated with the qualitative assessment, while almost all RL_D and RL_M are not. Thus, normalizing the scale by dividing by the number of classes, will lead to a decorrelation of metrics that

indicate size of responsibility. It is conceivable that the assessments by the judges of the ET Robot Contest 2010 were focused on the total size of responsibility and the existence of higher responsibility, and not the average (normalized) amount of responsibility in a class diagram.

Because this contest involved multiple experts and checklists, we assume that their assessments of the class diagrams are valid. Consequently, the findings herein confirm that there is a relation between class diagrams, validating RQ1.

Moreover, we assume that the conventional aggregated values and our metrics have different perspectives. Metric values aggregated by Total can not be tell apart. Therefore, the correlation between Total and the qualitative assessment provides only rough findings about the design construction.

Table 5 shows that a large value of Max of NOp leads to a high assessment. Classes with many operations are core classes, and class diagrams containing detailed parts will have these classes. However, the ideal number of these core classes remains unclear based on the findings about Max of NOp.

5.2 RQ2: Dose a design class diagram containing few highly responsible classes have low validity?

Our hypothesis is confirmed. NL_D of NAssoc, NL_M of NAttr and NL_M of NOp are significantly correlated with the qualitative assessment. In other words, if our metrics have a low value, the quality of the class diagram is low. This result shows that the validity of a design class diagram with a low value of these metric is low. The qualitative assessment indicates the degree of understandability of system, adequacy of responsibility assignment etc., which are indicators of the design validity.

In this study, highly rated class diagrams contain many highly responsible classes. Although highly responsible classes are regarded undesirable for source code [12], this is not necessarily the case for design class diagrams, which are generally omitted elements[15]. For a design class diagram to be considered sufficiently, important or complex parts are desclibed in detail, while the other parts are abbreviated. Furthermore, in a diagram not properly considered, some latently important parts are also abbreviated.

6 Threats to Validity

6.1 Threats to Internal Validity

We assumed that the qualitative assessments of the class diagrams are valid because the qualitative assessments involved multiple experts consulting checklists.

In this study, we extracted the criteria "Structsure", which is strongly related to design class diagram, and used it for analyses. However, the assessments may have been influenced by factors other than the class diagram. For future studies and analyses, it is desirable to use qualitative assessments for class diagrams only.

6.2 Threats to External Validity

In this study, the experimental data consisted of 65 design class diagrams in the domain of embedded system, which is not a large-scale domain. Thus, our findings should be applied to similar domains.

Additionally, in this study, we assumed that values $<$ $th70$ are normal, $th70 <$ values $\leq th80$ are slightly large, $th80 <$ values $\leq th90$ are large, $th90 <$ values are very large. However, these settings are subjective based on past studies and previous experiences. In the future, what adequate values for q of NL_D and NL_M should be determined.

7 Related Works

To provide insight into software evolution when analyzing maintainability, Vasilescu reported that micro-level metrics should be aggregated at the macro-level [18]. Our metrics can provide insight because they aggregate metric values at the macro-level.

To aggregate metric values by a system, Serebrenik cited the "Theil Index", which is an inequality measure in econometrics [16] that requires knowledge about the inequality to interpret. Our metrics (NL_D and NL_M) are easy to interpret because they directly indicate the number of classes for a metric value that exceeds a threshold.

Larman proposed the GRASP pattern as a software design principle [10]. This design principle provides a policy to handle responsibility assignment in object-oriented designs. Fowler introduced heuristics where highly responsible classes have a "bad smell" in the source code [12]. We quantitatively analyzed the feature of responsibility assignment and confirmed that highly responsible classes are not necessarily bad in a design class diagram.

Lange reported that the abstraction levels of UML differ according to the utilization purpose [9]. Because variations in the abstraction levels are assumed to affect the metric distributions in a class diagram, it is desirable to unify the abstraction-level and utilization purpose in class diagrams of a dataset when deriving threshold by Alves' method.

8 Conclusions and Future Work

Herein we confirm the usefulness of our proposed novel metrics, which focus on the proportion of metric values and responsibility assignment. Analysis using our metrics provides a quantitative guide for developing a design class diagram. Design class diagrams, which are regarded as valid by experts, contain more classes (NAssoc $\geq$ 3) and more

classes with value of NAttr (NOp) that is larger for an intercomparison in a single class diagram.

Our metrics can be used to analyze class diagrams, and they may be useful in evaluating, assessing, analyzing, and studying software metrics. Additionally, our metrics may help analyze source code features.

References

[1] T. L. Alves, C. Ypma, and J. Visser. Deriving metric thresholds from benchmark data. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[2] D. Bellin and S. S. Simone. *The CRC card book*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[3] M. Esperanza Manso, J. A. Cruz-Lemus, M. Genero, and M. Piattini. Models in software engineering. chapter Empirical Validation of Measures for UML Class Diagrams: A Meta-Analysis Study, pages 303–313. Springer-Verlag, Berlin, Heidelberg, 2009.

[4] M. Genero, M. Piatini, and E. Manso. Finding "early" indicators of uml class diagrams understandability and modifiability. In *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, ISESE '04, pages 207–216, Washington, DC, USA, 2004. IEEE Computer Society.

[5] M. Genero, M. Piattini, and C. Calero. Empirical validation of class diagram metrics. In *Proceedings of the 2002 International Symposium on Empirical Software Engineering*, ISESE '02, pages 195–, Washington, DC, USA, 2002. IEEE Computer Society.

[6] M. Genero, M. Piattini, E. Manso, and G. Cantone. Building uml class diagram maintainability prediction models based on early metrics. In *Proceedings of the 9th International Symposium on Software Metrics*, METRICS '03, pages 263–, Washington, DC, USA, 2003. IEEE Computer Society.

[7] Japan Embedded Systems Technology Association. *ET Robot Contest 2010 Assessment Criterion for Model (Japanese)*, 2010. http://www.etrobo.jp/2010/gaiyou/model.php.

[8] C. Lange, M. Chaudron, and J. Muskens. In practice: Uml software architecture and design description. *IEEE Softw.*, 23(2):40–46, Mar. 2006.

[9] C. F. J. Lange and M. R. V. Chaudron. Managing model quality in uml-based software development. In *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice*, STEP '05, pages 7–16, Washington, DC, USA, 2005. IEEE Computer Society.

[10] C. Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[11] M. E. Manso, M. Genero, and M. Piattini. No-redundant metrics for uml class diagram structural complexity. In *Proceedings of the 15th international conference on Advanced information systems engineering*, CAiSE'03, pages 127–142, Berlin, Heidelberg, 2003. Springer-Verlag.

[12] F. Martin, B. Kent, and et al. *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[13] A. Nugroho and M. R. Chaudron. A survey into the rigor of uml use and its perceived impact on quality and productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, pages 90–99, New York, NY, USA, 2008. ACM.

[14] A. Nugroho and M. R. Chaudron. Evaluating the impact of uml modeling on software quality: An industrial case study. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, MODELS '09, pages 181–195, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] M. Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 722–731, Piscataway, NJ, USA, 2013. IEEE Press.

[16] A. Serebrenik and M. van den Brand. Theil index for aggregation of software metrics values. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pages 1–9, Washington, DC, USA, 2010. IEEE Computer Society.

[17] R. T. Vargas, A. Nugroho, M. Chaudron, and J. Visser. The use of uml class diagrams and its effect on code change-proneness. In *Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling*, EESSMod '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.

[18] B. Vasilescu, A. Serebrenik, and M. van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, ICSM '11, pages 313–322, Washington, DC, USA, 2011. IEEE Computer Society.