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

Naohiko Tsuda
Dept. Computer Science and Engg
Waseda University
3-4-1 Ohkubo
Shijuku-ku, 169-8555, Japan

Hironori Washizaki
Dept. Computer Science and Engg
Waseda University
3-4-1 Ohkubo
Shijuku-ku, 169-8555, Japan
e-mail: washizaki@waseda.jp

Yoshiaki Fukazawa
Dept. Computer Science and Engg
Waseda University
3-4-1 Ohkubo
Shijuku-ku, 169-8555, Japan
e-mail: fukazawa@waseda.jp

ABSTRACT
Design models are often developed using UML class diagrams. Based on past questionnaire surveys reported by Lange and Nugroho, we assumed that the existence of highly responsible classes indicate that a class diagram was created through the proper consideration of the structure of the system. Thus, we analyzed the structure of UML design class diagrams. Specifically, we measured our novel metrics (the amount of highly responsible classes in class diagrams), and investigate the correlations between our metrics and the structural validity of design. In this study, we propose two viewpoints to distinguish large values which indicate high responsibility. Additionally, we conducted the evaluation experiment 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 structural validity. Then the correlations between our novel metrics and the experts’ qualitative assessment were analyzed.

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

1 Introduction

Design models are often developed using UML class diagrams [1], 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) [2] [3] [4] [5]. 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 are 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 [6] and Nugroho [7] reported that the important or complex parts of design model tend to be drawn in detail (e.g., properly named labels, avoidance of omitting elements, etc.). We regard detailed classes as highly responsible because such classes tend to be assigned many attributes, operations or etc. Additionally, we assume that unimportant classes tend to be not drawn in detail. If designers do not regard a class as important, the class will be not drawn in detail. Furthermore, if designers do not properly consider the structure of a system, they can not distinguish importance of classes. Thus, we construct the following hypothesis.

Hypothesis: If designers do not properly consider the structure of a system when creating a design class diagram, the class diagram will not contain especially detailed classes (highly responsible classes).

Table 1. Basic class metrics indicating size of responsibility

<table>
<thead>
<tr>
<th>Abbrev</th>
<th>Class Metrics Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NChld</td>
<td>The Number of Subclasses (Owned Children)</td>
</tr>
<tr>
<td>NAttr</td>
<td>The Number of Attributes (Owned)</td>
</tr>
<tr>
<td>NOp</td>
<td>The Number of Operations (Owned)</td>
</tr>
<tr>
<td>NAssoc</td>
<td>The Number of Associations (Owned)</td>
</tr>
</tbody>
</table>

We verified our hypothesis using our novel metrics to measure amount of highly responsible classes in a class diagram. The class metrics shown in Table 1 indicate the size of responsibility assigned to a class. For example, classes with large NAttr or large NAssoc will be highly responsible. Focusing on NAttr (NOP, or etc.) is one way to measure the amount of highly responsible classes. The number of highly responsible classes can be regarded as the number of large values in the set of values of NAttr (NOP, or etc.) from classes in the class diagram. Based on the above idea, we define two metrics to indicate the proportion of metric values: NL (Number of Large values) and RL (Ratio of Large values).

Preparing a threshold for a metric, which is one way to distinguish large values. The threshold can be derived

---

1"Owned" in Table 1 means that we count attributes (operations, etc.) directly owned by a class, but not including attributes (operations, etc.) inherited from parents of the class.
from the metric values distribution of a dataset. In this paper, we propose following two methods to derive the threshold.

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

For example, to derive the threshold to determine large values in a set of NAttr values taken from a class diagram ($D_x$), $C_D$ requires sets of NAttr values taken from multiple class diagrams in the same domain as $D_x$. By contrast, $C_M$ requires a set of NAttr values taken from only $D_x$.

We used $NL_D$, $RL_D$, $NL_M$ and $RL_M$ to verify our hypothesis. $NL_D$ and $RL_D$ ($NL_M$ and $RL_M$) are NL and RL measured by $C_D$ ($C_M$). $NL_D$ and $NL_M$ ($RL_D$ and $RL_M$) do not always show same tendency. For example, assume that there is a class diagram $D_i$ contains several classes having 100 attributes (NAttr = 100) for each. Additionally, in the same domain as $D_i$, there are other class diagrams contain classes having less than 10 attributes (NAttr < 10) for each. In this case, $NL_D$ of NAttr of $D_i$ is large because classes in $D_i$ have larger NAttr than classes in other class diagrams. By contrast, $NL_M$ of NAttr of $D_i$ is 0 because CM can not determine larger NAttr values in $D_i$. All classes in $D_i$ have same NAttr (=100).

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: Does 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., NLD of NAttr). All metrics were calculated using 65 design class diagrams, which were originally submitted to a Robot Contest [8] on the domain of embedded systems and evaluated by software development experts based on structural validity (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 large 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 [9][10]. Four basic class metrics shown in Table 1 indicate the size of responsibility assigned to a class. NAttr, NChild 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 2-4 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) [8]. In the contest, participants let uniform robots run along a black line on a white stage (Figure 1 shows), and the robots have same body (hardware) and different software.

![Figure 1](image.png)

**Figure 1.** The robot running along a black line on a white stage

Figures 2-4 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 using a light sensor which measures brightness of a robot’s underfoot. The light sensor samples 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.

Figure 2 is a part of a model extracted from the class diagram whose structure was assessed as high quality (B) by software development experts while Figures 3 and 4 are extracted from low quality (C) class diagrams. In Figure 2, important classes to calibrate was specified and only they are detailed (have many attributes and operations). Figure 2 contains necessary responsibility (attributes and operations) to calibrate while Figures 3 and 4 lack them.

In past questionnaire surveys, Lange[6] and Nugroho[7] reported that designers tend to write the important or complex parts of a design model in detail.
Figure 2. Calibration model in design class diagram (Diagram 1) submitted to the ET RoboCon 2010, Structure-Assessment(B)

Figure 3. Calibration model in design class diagram (Diagram 2) submitted to the ET RoboCon 2010, Structure-Assessment(C)

Figure 4. Calibration model in design class diagram (Diagram 3) submitted to the ET RoboCon 2010, Structure-Assessment(C)

(e.g., properly named labels, avoidance of omitting elements, etc.).

Details of Figures 2-4 are described below. In Figure 2, 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, Figure 2 indicate the necessity to save Max and Min brightness values of white, gray and black when deriving thresholds.

In Figure 3, all classes are detailed however almost all operations are constructor and destructor, namely Figure 3 contains less responsibility (attributes and operations) to discriminate color than Figure 2.

In Figure 4, 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 large values. If multiple class diagrams representing specific domain are available, Alves’ method can derive a metric thresholds [11].

\[ A_i = \{a_{i,1}, ..., a_{i,n}\} \]  \hspace{1cm} (1)

\[ T(q) = \{t_1, ..., t_k | t_i = \text{Quantile}(A_i, q)\} \]  \hspace{1cm} (2)

\[ th(q) = \text{Median}(T(q)) \]  \hspace{1cm} (3)

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 Diagram1(Diagrami), 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[6] and Nugroho[7] reported that designers tend to write the important or complex parts of a design model in detail (e.g., properly named labels, avoidance of omitting elements, etc.). We regard detailed classes as highly responsible because such classes tend to be assigned many attributes, operations or etc. Additionally, we assume that unimportant classes tend to be
not drawn in detail. If designers do not regard a class as important, the class will be not drawn in detail. Furthermore, if designers do not properly consider the structure of a system, they can not distinguish important classes which should be detailed (highly responsible). Thus, we assume that designer should focus on the proportion of metric values shown in Table 1.

Additionally, we assume that conventional values (e.g., 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 Large values.
- **RL**: Ratio of Large values.

Preparing a threshold for a metric, which is one way to distinguish large values. The threshold can be derived from the metric values distribution of a dataset. 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 derive the threshold:

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

For example, to derive the threshold to determine large values in a set of NAttr values taken from a class diagram ($D_x$), $C_D$ requires sets of NAttr values taken from multiple class diagrams in the same domain as $D_x$. By contrast, $C_M$ requires a set of NAttr values taken from only $D_x$. We propose $NL_D$, $RL_D$, $NL_M$ and $RL_M$. $NL_D$ and $RL_D$ ($NL_M$ and $RL_M$) are NL and RL measured by $C_D$ ($C_M$).

### 3.1 Our $C_D$ Metrics ($NL_D$ and $RL_D$)

Figure 5 illustrates image of our metrics $NL_D$70 (described later). Details of our metrics are described below.

Conveniently writing the threshold at $q=0.70$ as $th_{70}$; in the Alves’ method, values $< th_{70}$ are normal, $th_{70} <$ values $\leq th_{80}$ are slightly large, $th_{80} <$ values $\leq th_{90}$ are large, $th_{90} <$ values are very large.

Using the threshold derived by Alves’ method, we propose two novel metrics $NL_D$ (Number of Large values in the Domain) and $RL_D$ (Ratio of Large values in the Domain). $NL_D$ is the number of classes containing large metric value, and $RL_D$ is the percentage of these classes in the class diagram. $NL_D$ count large values in a set of metric values using the threshold prepared specifically for the domain while $RL_D$ calculates the ratio of the large values in the set. In this study, $NL_D$70 ($RL_D$70) is $NL_D$ ($RL_D$) calculated at $q = 0.70$.

\begin{align*}
X &= \{ x_1, \ldots, x_n \} \quad (4) \\
NL_D(X, th) &= |\{ x \in X | x > th \} | \\
RL_D(X, th) &= \frac{NL_D(X, th)}{|X|} \\
th &= Predetermined\ Threshold \quad (7)
\end{align*}

As an example, the calculation processes of $NL_D$70 of NAssoc are described below. Using Alves’ method, $th_{70}$ 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 $th_{70}$ becomes the $NL_D$70 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 Large values in a Model) and $RL_M$ (Ratio of Large values in a Model), which can be calculated without a predetermined threshold. $NL_M$ is the number of classes containing a large metric value while $RL_M$ is the percentage of these classes in the class diagram.

\begin{align*}
X &= \{ x_1, \ldots, x_n \} \\
NL_M(X, th) &= |\{ x \in X | x > th \} | \\
RL_M(X, th) &= \frac{NL_M(X, th)}{|X|} \\
th &= Quantile(X, q) \quad (11)
\end{align*}

As an example, Let $X$ be a set of NAssoc values from a single class diagram, and let $th_{70}$ be the 70% quantile of $X$. At a glance, the number of $x_i$ greater than $th_{70}$ 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 greater 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 Figure 6.

<table>
<thead>
<tr>
<th>Classes</th>
<th>NOp</th>
<th>NAssoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

The threshold at \(q=0.70\) is conveniently written as \(th70\). As well as the Alves’ method, values < \(th70\) are normal, \(th70 < \text{values} \leq \text{th80}\) are slightly large, \(th80 < \text{values} \leq \text{th90}\) are large, \(th90 < \text{values}\) 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 2-4 are parts of UML design class diagrams (Diagram 1-3). Tables 2 and 3 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 5. 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 corresponds to our hypothesis.

### 4 Evaluation Experiment

#### 4.1 Experiment

Enterprise Architect (EA) [12], developed by Sparx Systems, is a software to create software design models. We developed an automatic tool, which uses API provided by EA, to measure the class metrics from UML class diagrams in EA format.

Table 1 shows the four basic class metrics that indicate the size of responsibility assigned to class. Using the automatic measurement tool, we measured these class metrics using the 65 design class diagrams. Additionally, we measured our metrics values (i.e., \(NL_D70\) of NAttr, \(NL_D80\) of NOp etc.).

After the measurement, we investigated Spearman’s rank correlation between aggregated value and the quali-
tative 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 7 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 in pdf format submitted to the Japanese software development contest “ET Robot Contest 2010” [8] held by “Japan Embedded Systems Technology Association”. The contest aimed to improve technical education of embedded systems. To measure class metrics automatically, we manually reconstructed UML class diagrams in Enterprise Architect (EA) [12] format.

The class diagrams show the structure of software to control autonomous robots which run along a line using a light sensor. To design an autonomous robot, the class diagrams were written by students and working adults with software development experience. In the contest, around 4-5 software development experts qualitatively assessed each submitted model by the consultation. The experts had about 10-20 years of experience in software development. Members of reviewers (experts) vary depending on the year of the contest. Because the ratings are assigned on an ordinal scale (A-D) after multiple experts consult a check-list, 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 ”Structure”, which is strongly related to design class diagram, and used it for analyses. The rank of ”Structure” (A-D) indicates degree of understandability of system, adequacy of responsibility assignment etc.

4.3 Results

We used 65 class diagrams in our experiment. Speaking about the number of classes (NOC) of these class diagrams, Max is 48, Min is 6, Mean is 20.8, Median is 19. Almost all of these class diagrams are small.

Table 4 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 5 show the thresholds of four metrics derived by Alves’ method for $NL_D$ and $RL_D$.

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 correlated with the experts’ 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 4 shows that $NL_D$ of NAssoc is significantly correlated with the qualitative assessment. Class diagrams containing more classes with NAssoc $\geq 3$[or4] (Table 5) 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 4 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 inter-comparing 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 check-lists, 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 4 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 [13], this is not necessarily the case for design class diagrams, which are generally omitted elements[1]. For a design class diagram to be considered sufficiently, important or complex parts are described 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 check-lists.

In this study, we extracted the criteria “Structure”, 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

We conducted the experiment using small size design class diagrams. The findings we got are about small size design class diagrams. However, the findings are significant because they clarify the relations between the structural validity of design and quantitative features of class diagrams.

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 [14]. 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 [15] 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 [9]. This design principle provides a pol-
icy to handle responsibility assignment in object-oriented designs. Fowler introduced heuristics where highly responsible classes have a “bad smell” in the source code [13]. 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 [16]. 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 Conclusion

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 ≥ 3) and more classes with value of NAttr (NOp) that is large for an inter-comparison 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


