

# Evaluation of Understandability of UML Class Diagrams by Using Word Similarity

Yuto Nakamura

Nomura Research Institute, Ltd.  
n\_y\_1121@ruri.waseda.jp

Kiyohisa Inoue

Dept. Computer Science, Waseda University  
inoue.k@ruri.waseda.jp

Kazunori Sakamoto

Dept. Computer Science, Waseda University  
kazuu@ruri.waseda.jp

Hironori Washizaki and Yoshiaki Fukazawa

Dept. Computer Science, Waseda University  
{washizaki, fukazawa}@waseda.jp

**Abstract**—UML class diagrams representing the static structure of the relations between different concepts existing in a problem are widely used in model-based software development. However, no effective measures of a class diagram’s understandability yet exist. We have devised quantitative measures of a class diagram’s understandability and evaluated their validity. We obtained strong correlations between the domain experts’ subjective evaluations of the understandability of a class diagram and the measurements of our methods. These results indicate that our measures can effectively quantify the understandability of class diagrams.

**Keywords**—Software Metrics; Software Evaluation; Software design metrics; UML class diagrams; Maintainability; Understandability.

## I. INTRODUCTION

Unified Modeling Language (UML) [1] is a useful modeling language that standardizes the notation of various models being used for object-oriented development. Knowledge sharing is supposed to be easier if the reader of the UML diagrams understands the notation of the UML models. The class diagram in the definition of the UML represents the concepts existing in the problem and a static structure of the relations between these concepts.

The class diagram is understood by evaluating the validity of the relations between the classes. In particular, this is achieved by comparing the relations between the concepts described in the class diagram with the relations between the concepts that the reader tacitly possesses. When both structures are similar, the reader normally believes that the relational structure of the class diagram is valid and that its understandability is high. When the structures are dissimilar, the reader may feel that the validity and understandability of the diagram are poor.

In light of this, the following self-review method has been suggested [2]: Read “word A and word B have a relationship” by using the class names of a related edge. If you feel uncomfortable, you should review the layout of the responsibilities. However, a correlation between the validity evaluation of the class relations like mentioned above, and the understandability of the class diagram have not yet been elucidated.

A wide variety of metrics, i.e., objective standards for measuring the quality of software, have been proposed for a class diagram. For example, there are CK Metrics [3] that can measure the sum of the complexities of the class’s operation (*WMC*) and lack of cohesion of methods (*LCOM*), etc. However, there is no metric for measuring the understandability of a class diagram. There is an article [5], but the correlation between the qualitative evaluation of the models covering various qualities and the complexity of the structure based on uniting the classes [4] is rather weak.

If valid metrics of the class diagram’s understandability were to be defined, they would be useful for estimating the time-cost involved in understanding a class diagram, for creating objective reviews based on the understandability of the diagram, for selecting the best models from among all the others for the same problem, and for detecting the parts of the class diagram in which the understandability is poor.

Our fundamental hypothesis is as follows: if the similarity of two words is a high degree, the reader of the class diagram will recognize the close relation between them. Furthermore, if two closely related words are near each other on the class diagram, the understandability of the class diagram will be high. The above hypothesis led us to propose three metrics for the understandability of a class diagram. We evaluated their validity and the hypothesis by analyzing the correlation between the subjective evaluations of understandability and measurements given by the metrics.

## II. METRICS CONCERNING UNDERSTANDABILITY OF THE CLASS DIAGRAM

A class diagram has three aspects, as Fig. 1 shows, the class structure and package structure as a macro view, the attributes, and the operations as the micro views.

We shall now define the fundamental hypothesis. If two words are similar, the reader of a class diagram would naturally recognize the close relation between them. And also, if the relation between two elements on the class diagram is close, the reader would expect the words of the elements are similar. That is, if the relation is close and the two words are similar, the reader would easily understand the class diagram. The above hypothesis can be recast into three versions based on the three aspects of the class diagram.

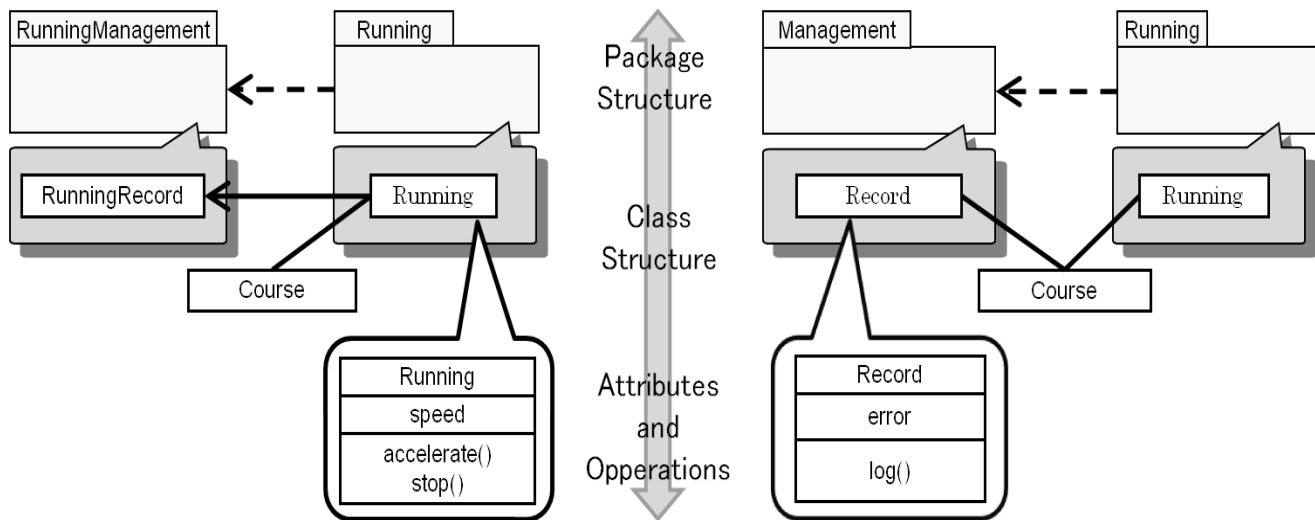


Figure 1. Three aspects of class diagram (Left: Figure. 1.1, Right: Figure. 1.2)

### H1) Class Structure Hypothesis

If the following two detailed hypotheses hold for the entire class diagram, the class diagram will be easily understandable.

- The similarity between two class names increases as the relation between the two classes in the class diagram is strengthened in terms of the class's static relational structure.
- The similarity between two class names decreases as the relation between two classes in the class diagram weakens in terms of the class's static relational structure.

Fig. 1.1 and 1.2 have similar classes. However, Fig. 1.1 seems to have a higher level of understandability, because the "RunningRecord" class and "Course" class in Fig. 1.1 are not similar class names, and also are indirectly connected with each other in the class diagram. On the other hand, the "Course" class and "Record" class in Fig. 1.2 are also not similar class names, but are directly connected with each other in the class diagram.

### H2) Package Structure Hypothesis

If the following detailed hypotheses hold for the entire class diagram, the understandability will be high.

- The similarity between two package names increases as the relation between the two packages in the class diagram is strengthened in terms of the package's static relational structure.
- The similarity between the two package names decreases as the relation between two packages in the class diagram weakens in terms of the package's static relational structure.

Fig. 1.1 has a higher level of understandability than Fig. 1.2, because the "RunningManagement" package and "Running" package in Fig. 1.1 are similar package names, and also are directly connected with each other in the class diagram. On the other hand, the "Management" package and "Running" package in Fig. 1.2 are not similar package names, but are directly connected with each other in the class diagram.

### H3) Attributes and Operations Hypothesis

If the attributes and operations with similar names are put into one class, the cohesion of that class will be high. Furthermore, the understandability increases with the degree of cohesion.

In Fig. 1.1, the class name, attributes, and operations in the "Running" class, such as (Running, speed), (Running, accelerate) and (speed, stop) are closely related to each other. If H3 were true, the "Running" class would have high cohesion and understandability. In contrast, the cohesion and understandability of Fig. 1.2 are lower than in Fig. 1.1 because the relation between "Record" and "error" in Fig. 1.2 seems weak.

In order to measure the understandability of the class diagram by using the above hypotheses, we need to devise methods for analyzing the class structures and measuring the word similarities. The class structure can be analyzed by using graph theory. The way of measuring the word similarity is defined in the next paragraph.

#### A. Word Similarity

People can naturally decide whether two words are similar in meaning. However, they do so qualitatively, not quantitatively. To dispense with personal biases, we shall measure the similarity between two words by using the PathLength method defined in WordNet::Similarity [7]. WordNet::Similarity is created from the WordNet [6] thesaurus.

First, the PathLength method looks for common upper concepts and determines the shortest path between two words. Next, the word similarity between word  $x$  and word  $y$   $Wsim(x,y)$  is defined by taking the reciprocal of the number of nodes (the shortest path distance + 1) on the shortest path:

$$Wsim(x, y) = \frac{1}{\text{conceptual distance between } x \text{ and } y + 1}$$

Fig. 2 shows an example of a similarity measurement between "running" and "record".

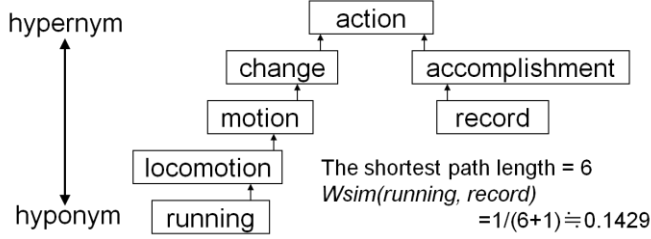


Figure 2. Word similarity between “running” and “record”

Class diagrams frequently describe “compound words”, which are a combination of two or more words.

A similarity measure for compound words consists of a number of steps. 1) The compound words  $cw_a$  and  $cw_b$  are divided into resolution words  $cw_a = \{w_{a1}, \dots, w_{am}\}$  and  $cw_b = \{w_{b1}, \dots, w_{bn}\}$  by using a capitalization naming rule<sup>1</sup>, respectively. The number of resolution words in  $cw_a$  is  $|cw_a| = m$ . 2) The pair of the word in  $cw_a$  and the word in  $cw_b$  with the highest  $Wsim$  is added to the sum of  $Wsim$ . 3) The selected words in  $cw_a$  and  $cw_b$  in the 2nd step are removed from  $cw_a$  and  $cw_b$ , respectively. 4) The 2nd and 3rd steps are repeated  $n$  (The smaller number of resolution words) times until a smaller number of resolution words. It means Eq. (1) is calculated repeatedly. 5) The average value  $CWsim$  (compound word similarity) is computed by dividing the sum of  $Wsim$  by the larger number of resolution words:

$$\begin{aligned}
 cw_a &= \{w_{a1}, \dots, w_{am}\} & |cw_a| &= m \\
 cw_b &= \{w_{b1}, \dots, w_{bn}\} & |cw_b| &= n \quad (m \geq n)
 \end{aligned}$$

$$mp_k(cw_a, cw_b) = \max(\{Wsim(w_{ax}, w_{by}) \mid (w_{ax}, w_{by}) \in cw_a - \{w_{ax}\} \times cw_b - \{w_{by}\}\}). \quad (1)$$

( $w_{ax}$ : resolution words of  $cw_a$  used in before term)  
( $w_{by}$ : resolution words of  $cw_b$  used in before term)

$$CWsim(cw_a, cw_b) = \sum_{k=1}^n mp_k(cw_a, cw_b) / m. \quad (2)$$

To calculate Eq. (1), which accurately reflects the information concerning the compound words, a resolution word is removed from the after term’s choice after it is used once. Moreover, dividing the sum by the larger number of resolution words is meant to reflect the information concerning the words excluded from Eq. (2).

The following sentence shows an example of measuring the  $CWsim$  of “sensorID” and “getSensorID”. Table 1 lists the  $Wsim$  measurements of all the pairs of resolution words.

TABLE I. WSIM MEASUREMENTS OF RESOLUTION WORDS

	Get	Sensor	ID
Sensor	0.05	1	0.0769
ID	0.0625	0.0769	1

<sup>1</sup> This rule is well known as the “Camel Case”. The naming rule that the writer observes may be different depending on where he belongs. However, “Camel Case” may be the most common naming rule. Thus, a capitalization naming rule was adopted for our resolution method.

The smaller number of resolution words is 2. Thus, two pairs of resolution words that have high  $Wsim$  are needed and one pair’s words should be different from the other pair’s words. In the example, the correct pairs are {(Sensor, Sensor), (ID, ID)}. The larger number of resolution words is 3. Therefore, we get  $CWsim$  by dividing the sum of the selected pairs’  $Wsim$ s by 3:

$$CWsim(SensorID, getSensorID) = (1+1)/3 = 0.6667$$

## B. Class Relational Structure Understandability

**Definition.**  $C$  represents a set of classes in a class diagram,  $|C|$  represents the number of classes, and a class used as the element is defined as  $c_x$  ( $x = 1, \dots, |C|$ ).  $div$  represents the distance of the shortest path between two classes that can be reached by using the tracing relation while taking the reference direction into consideration.

For example, the value of  $div$  between the “RunningRecord” class and the “Course” class in Fig. 1.1 is 2 and the value of  $div$  between the “Record” class and the “Course” class in Fig. 1.2 is 1.

Regarding  $HI$ , we define the expected similarity between two class names on the basis of the shortest distance of the class’s static relational structure as  $ESim$  (expected similarity in terms of relational distance).  $ESim$  represents the strength of relation in terms of the class’s static relational structure. The equation is as follows:

$$ESim(c_x, c_y) = 1 / (div(c_x, c_y) + 1) \quad (x \neq y)$$

Here,  $0 < ESim(c_x, c_y) \leq 1/2$ .  $ESim$  approaches  $1/2$  as the shortest distance between two classes decrease.

$CWsim(c_x, c_y)$  represents the similarity between two class names. The difference between  $ESim(c_x, c_y)$  and  $CWsim(c_x, c_y)$  is normalized so that the understandability will be higher when the difference approaches 1 and the understandability will be lower when the difference approaches 0. We define the normalized difference as the relational distance understandability ( $RDU$ ):

$$RDU(c_x, c_y) = 1 - |ESim(c_x, c_y) - CWsim(c_x, c_y)|$$

The class relational structure understandability ( $CRSU$ ) is defined by taking the average of the  $RDU$ s for the entire class diagram. The equation is as follows:

$$CRSU(C) = \frac{\sum_{c_x \in C} \sum_{c_y \in C - \{c_x\}} RDU(c_x, c_y)}{|\{(c_a, c_b) \in C \times C \mid c_a \neq c_b \wedge div(c_a, c_b) \geq 1\}|}$$

If  $HI$  is valid, the  $CRSU$  should have the following characteristics: as the  $CRSU$  approaches 1, the understandability of the class diagram to the reader increases. Moreover, as the  $CRSU$  approaches 0, the understandability decreases.

TABLE II. DATA USED IN MAKING  $CRSU$  MEASUREMENT IN FIG. 1.1

$c_x$	$c_y$	$Div$	$ESim$	$CWsim$	$RDU$
Course	Running	1	0.5	1	0.5
Running	RunningRecord	1	0.5	0.5	1
Course	RunningRecord	2	0.333	0.125	0.791

**Process of CRSU measurement.** We developed a system that automatically measures the *CRSU*. Fig. 3 shows an overview of the system. Let us give an example for measuring the *CRSU* in Fig. 1.1. Table 2 lists the data from the *CRSU* measurements.

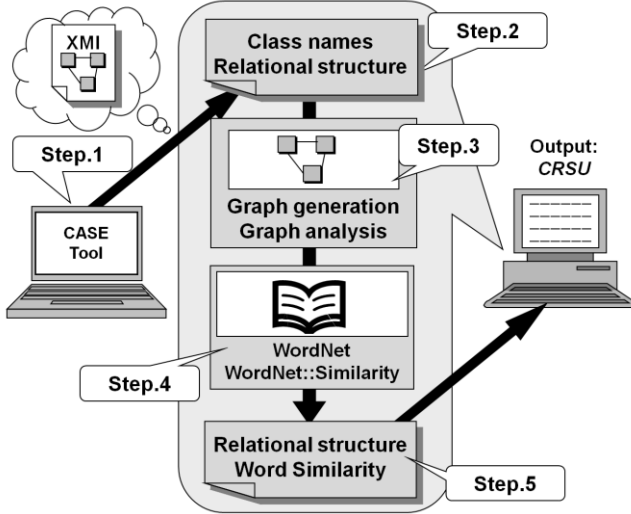


Figure 3. Overview of CRSU measuring system

*Step1. Get XMI data from the CASE tool and input the data.* The class diagram expressing Fig. 1.1 is drawn up using the CASE tool, and the class diagram must be output in the XMI [8] format. It is necessary to perform this process manually.

*Step2. Extract class names and relational structure.* By analyzing the XMI data obtained in step 1, the system acquires the class names and relations between the classes. When the class names are in Japanese, the class names are translated into English by using a translation service [9]. Translations are sometimes necessary because WordNet uses only English words. We decided to use a translation service because it would remove as much of the personal biases of the translation as possible.

*Step3. Generate and analyze the graph.* A directed graph is generated from the results of step 2 reflecting the direction of the reference and the direction of the dependency, and the distance of the shortest path between the two classes where the route exists is calculated. When the direction of the reference is not clearly described in the relation, the system treats the relation as an interactive one. Furthermore, the system treats all kinds of relations in the same way.

*Step4. Measure the word similarity.* Paragraph 2.2 discusses the methods for measuring the “word similarity” and “compound word similarity”.

*Step5. Calculate the CRSU using the results from Steps 3 and 4.* The *CRSU* is calculated by using the defining equation in paragraph 2.2.1. In this case,  $CRSU = (0.5+1+0.7917) / 3 = 0.7639$ . The *CRSU* of Fig. 1.2 is about 0.6944. When comparing Figs. 1.1 and 1.2 again, we see that both diagrams have similar classes, but different class structures. Fig. 1.1 does not have any doubtful relations and seems valid. In contrast, the relation between the “Course” class and “Record” class in Fig. 1.2 does not seem to be very close. Therefore, the validity of Fig. 1.2 is doubtful.

The *CRSU* measurements seem to correspond to the reader’s evaluations. Therefore, we can say that in the case of Fig. 1, the *CRSU* is appropriate as a measure of understandability for the class diagram and that hypothesis *H1* is valid.

### C. Package Relational Structure Understandability

$P$  represents a set of packages in the class diagram, and  $|P|$  represents the number of packages. A package is defined as  $p_x$  ( $x = 1, \dots, |P|$ ).  $div$  represents the distance of the shortest path between two packages that can be reached by tracing the relation in the direction of the dependency.

For example, the value of  $div$  between the “RunningManagement” package and the “Running” package in Fig. 1.1 is 1 and the value of  $div$  between the “Management” package and the “Running” package in Fig. 1.2 is 1.

Regarding *H2*, we denote the expected value of the similarity between two package names on the basis of the shortest distance in the static relation-structure as the expected similarity in terms of relational distance (*ESim*):

$$ESim(p_x, p_y) = 1 / (div(p_x, p_y) + 1) \quad (x \neq y)$$

Here,  $0 < ESim(p_x, p_y) \leq 1/2$ . *ESim* approaches 1/2 as the shortest distance between two packages gets smaller.

$CWsim(p_x, p_y)$  represents the similarity between two package names. The difference between  $ESim(p_x, p_y)$  and  $CWsim(p_x, p_y)$  is normalized so that the level of understandability increases as the difference approaches 1. We denote the normalized difference as the relational distance understandability (*RDU*). The equation is:

$$RDU(p_x, p_y) = 1 - |ESim(p_x, p_y) - CWsim(p_x, p_y)|$$

The package relational structure understandability (*PRSU*) is defined as the average *RDU* for the entire class diagram. The equation is:

$$PRSU(P) = \frac{\sum_{p_x \in P} \sum_{p_y \in P - \{p_x\}} RDU(p_x, p_y)}{|\{(p_a, p_b) \in P \times P \mid p_a \neq p_b \wedge div(p_a, p_b) \geq 1\}|}$$

If *H2* is valid, the *PRSU* should have the following characteristics: the understandability should improve as the *PRSU* increases. Moreover, the understandability decreases when the *PRSU* decreases.

The *PRSU* of Fig. 1.1 is 1, and the *PRSU* of Fig. 1.2 is about 0.6429. The package dependency between “Running” and “RunningManagement” in Fig. 1.1 seems more valid than that of Fig. 1.2. The *PRSU* values correspond to the reader’s instinctive assessments of the package structure. As a result, it seems that the *PRSU* and hypothesis *H2* are appropriate in the case of Fig. 1.

### D. Class Cohesion based on Word Similarity

$C$  represents a set of the classes in the class diagram,  $|C|$  represents the number of classes, and a class is defined as  $c_x$  ( $x = 1, \dots, |C|$ ).  $A_x$  represents a set of the attributes of  $c_x$ ,  $|A_x|$  represents the number of attributes of  $c_x$ , and an attribute is defined as  $a_x$  ( $x = 1, \dots, |A_x|$ ).  $M_x$  represents a set of the

operations of  $c_x$ ,  $|M_x|$  represents the number of operations of  $c_x$ , and an operation is defined as  $m_x$  ( $x = 1, \dots, |M_x|$ ).

Regarding *H3*, we define three equations: *simCA* (average similarity between the class name and attribute names), *simCM* (average similarity between the class name and operation names), and *simAM* (average similarity between the attribute names and operation names):

$$simCA(c_x) = \frac{1}{|A_x|} \sum_{a_{xi} \in A_x} CWsim(c_x, a_{xi})$$

$$simCM(c_x) = \frac{1}{|M_x|} \sum_{m_{yj} \in M_x} CWsim(c_x, m_{yj})$$

$$simAM(c_x) = \frac{1}{|A_x| + |M_x|} \sum_{a_{xi} \in A_x} \sum_{m_{yj} \in M_x} CWsim(a_{xi}, m_{yj})$$

The averages are also calculated for all items in the entire class diagram. Note that the classes without item values are ignored in these calculations:

$$simCA(C) = \frac{\sum_{c_x \in C} simCA(c_x)}{|\{c_y \in C \mid simCA(c_y) > 0\}|}$$

$$simCM(C) = \frac{\sum_{c_x \in C} simCM(c_x)}{|\{c_y \in C \mid simCM(c_y) > 0\}|}$$

$$simAM(C) = \frac{\sum_{c_x \in C} simAM(c_x)}{|\{c_y \in C \mid simAM(c_y) > 0\}|}$$

By summing these three equations, we arrive at class cohesion based on word similarity (*CCS*):

$$CCS(C) = simCA(C) + simCM(C) + simAM(C)$$

Fig. 1.1 has a *CCS* of 1.11667, and Fig. 1.2 has one of 0.4442. The *CCS* values correspond to the reader's qualitative assessment of class cohesion. Therefore, it seems that *CCS* and *H3* are appropriate in the case of Fig. 1.

### III. EVALUATION

We used the three metrics on 29 models extracted at random from the 41 models submitted to the Tokai regional preliminaries of the ET Robot Contest 2008 [10] to save the effort of drawing class diagrams from submitted models. There is the detail of the extracted models in the appendix. The participating teams created the UML diagrams while developing the control software for a robot that automatically traces lines in the course. They competed on the speed of a robot and the quality of the model of the created software.

Robotics domain experts evaluated the models according to constant criteria with the consensus-building. There were professors of software engineering, developers of embedded device and engineers of quality evaluation. They tried to objectively evaluate the models to remove their bias. Their evaluations ranged through letter grades, with "A" meaning excellent and "D" meaning poor, for 5 "Writing of the model" items and 6 "Content of the model" items.

We show the histogram of the all models and the extracted models about the numbers of classes and subjective evaluation in Fig.4, 5, 6 and 7 to show the random extraction.

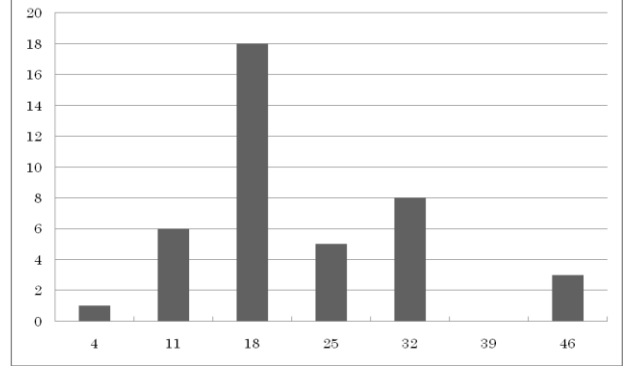


Figure 4. histogram of all models (x-axis: numbers of classes, y-axis: numbers of models)

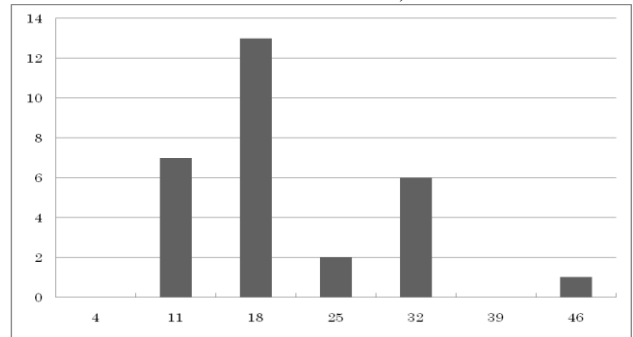


Figure 5. histogram of randomly extracted models (x-axis: numbers of classes, y-axis: numbers of models)

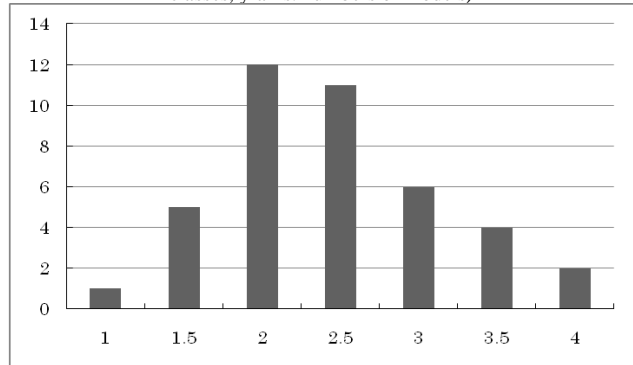


Figure 6. histogram of all models (x-axis: subjective evaluation, y-axis: numbers of models)

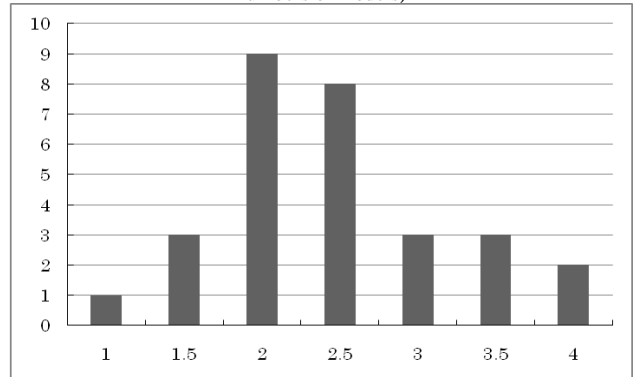


Figure 7. histogram of randomly extracted models (x-axis: subjective evaluation, y-axis: numbers of models)

We selected two items concerning the understandability of the class diagram as suitable comparative contrasts with our metrics:

- Easy to read (in “Writing of the model”): Does the reader of the class diagram find the model easy to read? Is the layout of the class diagram appropriate?
- Structure (in “Content of the model”): Are all classes and relations between them valid?

#### A. Evaluation of CRSU

Fig. 8 shows a distribution chart with the *CRSU* on the x-axis and the average of the subjective evaluation on the y-axis. The average was calculated by converting the grade of A into 4 points, B into 3, C into 2, and D into 1 point. The model review especially valued “Content of the model”. Therefore, the “Structure” item was weighted twice in our evaluation. For example, if “Easy to read” is C and “Structure” is B, the average is  $(2 + 3 \times 2) / 2 = 4$ .

The correlation coefficient between the *CRSU* and the subjective evaluation was 0.6400; therefore, there was a strong correlation. The numerical formula in Fig. 8 is a simple linear regression equation; the dependent variable is the average of the subjective evaluation, and the independent variable is the *CRSU* values.

In this paragraph, let us test the hypothesis that the coefficient of the regression equation becomes 0 with statistical hypothesis testing. The  $|t|$  value of the 1% significance level in the amount of freedom 27 is 2.771, whereas the  $|t|$  value of the regression expression is 4.328. Thus, the hypothesis that the coefficient becomes 0 at a significance level of 1% can be rejected, and the validity of the regression equation is proven.

As a result, the subjective evaluation concerning the understandability of the class diagram can be predicted by substituting the *CRSU* measurements for the variable of the regression equation.

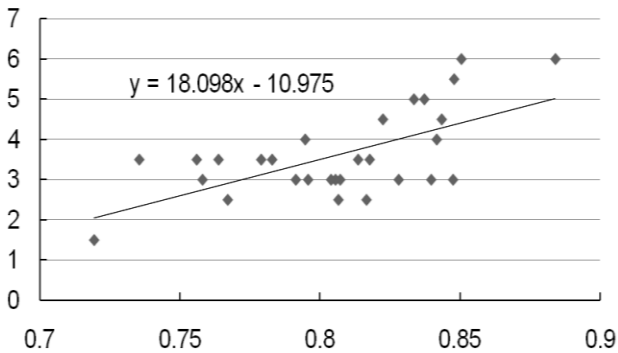


Figure 8. Distribution chart (x-axis: *CRSU*, y-axis: subjective evaluation)

Here are two examples.

#### 1) Class diagram with high *CRSU* and subjective evaluations

Fig. 9 shows part of a class diagram that had scores distributed near the approximate line in Fig. 8. The *CRSU* measurements and the subjective evaluation were both high. The *CRSU* measurements of the class diagram are 0.8434,

whereas the subjective evaluations are “B” for “Easy to see” and “B” for “Structure”. A “B” grade means the class diagram is sufficiently easy to understand and the relations between classes are valid.

In Fig. 9, all of the classes related to the “LineTrace” class are named “XXXPart”. This naming rule may increase the consistency of the class names and increase the similarity between the class names. Furthermore, the relations between the classes concerning “hardware” and “behavior” are valid.

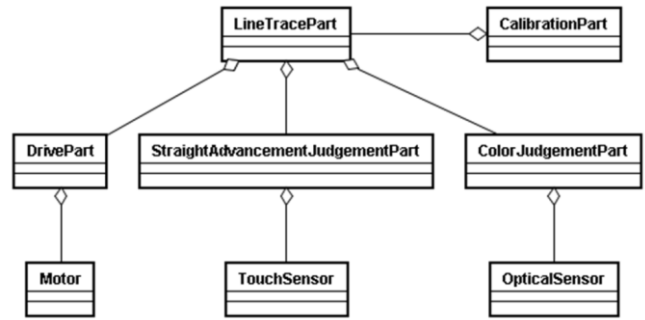


Figure 9. Class diagram (high *CRSU* - high subjective evaluation)

#### 2) Class diagram with low *CRSU* and subjective evaluations

Fig. 10 shows part of a class diagram that was far from the approximate line in Fig. 8 and for which the *CRSU* and subjective evaluation were low. The *CRSU* measurements of the class diagram are 0.7670, and the subjective evaluations were a “D” for “Easy to see” and a “C” for “Structure”.

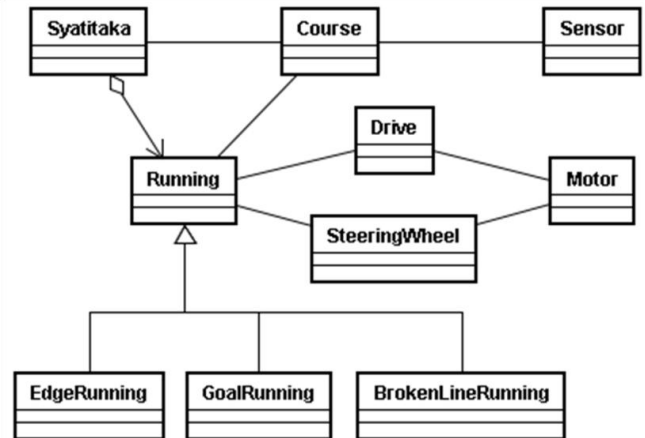


Figure 10. Class diagram (low *CRSU* - low subjective evaluation)

The coined word “Syatitaka” doesn't exist in the WordNet thesaurus. The similarity of the class name between the classes in which one edge of the relation is “Syatitaka” becomes 0. Thus, the “Syatitaka” class decreases the *CRSU* measurements. Moreover, the responsibility of the “Syatitaka” class isn't clear, and the “Structure” evaluation is accordingly lower.

We shall discuss the measurement and the evaluation of the *CRSU* in what follows:

#### 1) Development phase of class diagram

In software development, the class diagram represents the structure of the relations between the concepts in the object problem. Its understandability can be evaluated by using a thesaurus that compiles the relational structures of general concepts not depending on a specific domain.

On the other hand, the class diagram has different roles in the analysis and design phases. In the analysis phase, the class diagram represents the concepts and their relational structure between the concepts in the object problem. In the design phase, however, the concepts and relations that are not part of the problem are added in order to control the computer. Therefore, experimentation is necessary when the analysis phase and the design phase are separate.

### 2) Direction of references

The *CRSU* treats only the direction of the reference described by the navigable property of the class diagram. We thought the validity of the relational structure may be judged from the classes that can be reached by the routes in the reference direction, and the defined *CRSU*. However, there is a possibility that the validity of the relational structure is judged by not taking into account the reference direction.

If the reference direction is not taken into consideration, i.e., all of the relations are treated as an interactive one, the correlation coefficient is 0.2816. This value indicates that the reader of the class diagram makes an allowance for the reference direction when they understand the class diagram. Fig. 11 shows the distribution chart of the subjective evaluation and the *CRSU* when the reference direction is not taken into account.

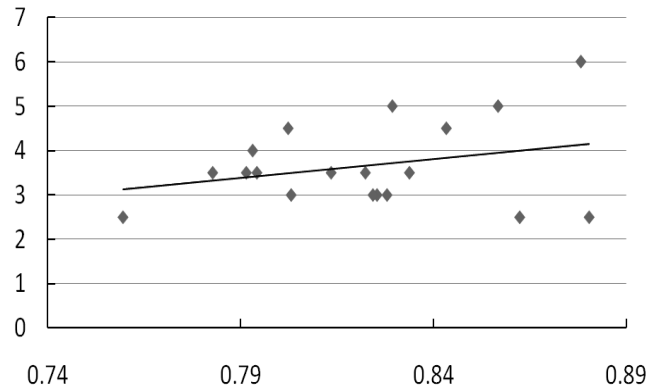


Figure 11. Distribution chart (x-axis: CRSU ignoring reference direction, y-axis: subjective evaluation)

### 3) Comparison with other metric

The coupling factor (*COF*) is a measure of the static complexity based on the coupling between classes of the object-oriented software [4]. The *COF* is measured by regularizing the number of single direction relations excluding the inheritance and the dependency from 0 to 1 regardless of the scale of the class diagram. It is defined as:

$$COF(C) = \frac{|NOR|}{|C|^2 - |C| - 2|NOS|}$$

$C$  represents a set of classes in a class diagram,  $|C|$  represents the number of classes,  $|NOR|$  represents the number of single-direction relations excluding the inheritance and dependency, and  $|NOS|$  represents the number of sub classes.

The correlation coefficient between the *COF* value and the subjective evaluation was -0.3880 in Fig. 12. The negative correlation means the subjective evaluation concerning the understanding decreases if the *COF* value, measurements of the static complexity of the class diagram, increases. By comparing the correlation coefficient of the *COF* and *CRSU*, the *CRSU* has a stronger correlation than that of the *COF*. This result shows that the *CRSU* is more appropriate as a measure of the understandability than the *COF*, because *CRSU* takes into account the class structure and word similarity, but *COF* takes into account only the class structure.

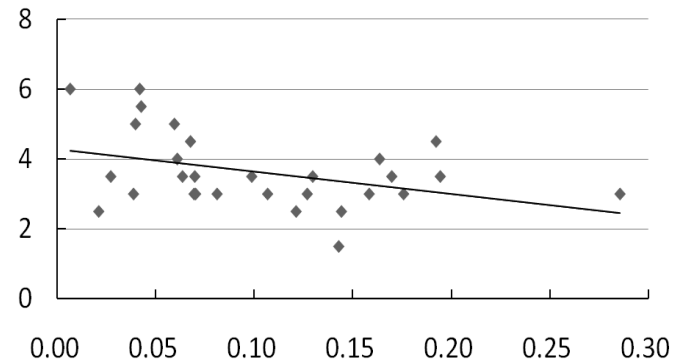


Figure 12. Distribution chart (x-axis: COF, y-axis: subjective evaluation)

### B. Evaluation of PRSU

Fig. 13 shows the distribution chart of the average subjective evaluation versus the *PRSU* measurements. The correlation coefficient between the *PRSU* and the subjective evaluation is -0.3219. The number of targeted class diagrams is only nine because there are class diagrams that don't possess a package structure.

The *PRSU* is inappropriate because the correlation is weak and the distribution does not support our hypothesis. We will collect more samples and evaluate the *PRSU* again.

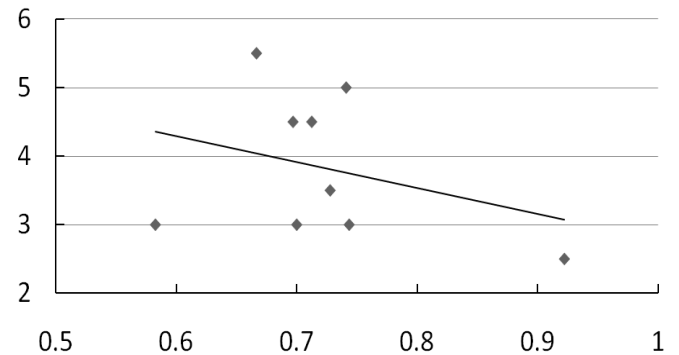


Figure 13. Distribution chart (x-axis: PRSU, y-axis: subjective evaluation)

### C. Evaluation of CCS

We measured the *CCS* measurements of 12 class diagrams that accurately describe the attributes and operations. Fig. 14 shows the distribution chart. The correlation coefficient between the *CCS* and subjective evaluation was 0.1564. The correlation is weak, and therefore, the results do not prove the validity of the *CCS*.

From the results in paragraphs 3.2 and 3.3, we can assume that the understandability of the class diagram is more strongly influenced by the class structure than by the package structure or class cohesion.

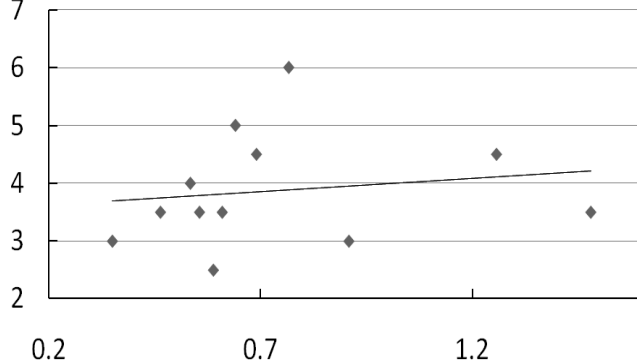


Figure 14. Distribution chart (x-axis: CCS, y-axis: subjective evaluation)

### IV. PREVIOUS WORK

CK Metrics [3] were defined by Chidamber and Kemerer in 1994. These metrics are the measures of the structures in the object-oriented design. However, they don't measure the class diagram's understandability.

SD Metrics [11] is a well-known metrics measure tool for UML models. SD Metrics comprise 32 kinds of metrics for the class diagram. Moreover, they can verify whether the class diagram satisfies the design rules. For example, they can verify whether the number of attributes and operations that the class possesses is less than 60, and verify that the class diagram does not have any unnamed classes. However, they do not measure the understandability of the class diagram.

There is a study on using a thesaurus to support the requirements analysis [12]. The thesaurus was created by domain experts extracting words from the software package. In the case of our measurements, the class diagram of the specific domain was measured by using a domain-free thesaurus. However, the accuracy of our measurements would have been higher if we had used a domain-specific one.

José et al. [13] referred to the understandability of UML Statechart diagrams. Their research showed the correlation is strong between certain structural metrics concerning the complexity and understandability of the Statechart diagram.

There was an attempt to predict a comprehensive qualitative evaluation from the structural complexity of a class diagram [5]. This paper shows that the correlation between the subjective evaluations of various qualities by domain experts and the *COF* was weak. This means that there is a factor in the UML models' evaluation that is more

essential than the class structure. We supposed that this essential factor is the "word similarity".

### V. CONCLUSION AND FUTURE WORK

We focused on the word similarity as calculated from the WordNet thesaurus, and proposed three metrics concerning the understandability of three aspects of the class diagram. Furthermore, the three metrics were evaluated by analyzing the correlation between the metrical measurements and the subjective evaluations of the class diagram's understandability made by domain experts.

The correlation between the subjective evaluation and class relational structure understandability (*CRSU*) was strong. Therefore, hypothesis *H1* and *CRSU* are valid. *CRSU* would be useful for estimating the time-cost spent for understanding the class diagram, and it could be used to select a more understandable class diagram from among the models concerning the same problem. Various usages will be investigated.

The validity of class cohesion based on word similarity (*CCS*) and package relational structure understandability (*PRSU*) were not proven in this paper. We will attempt to improve *PRSU* and *CCS* in the future.

Our future work is as follows:

1) Measurement and Evaluation of different domains

The candidate models in this study were for built-in software. However, there is a possibility that different results will be obtained for different domains. Therefore, we will evaluate our metrics in different domains.

2) Validation of *RDU*

The validity of *CRSU* was proven in this paper, but the validity of *RDU* was not proven because the subjective evaluation that was used for the correlation analysis was for the entire class diagram.

If the validity of *RDU* can be proven in a subjective experiment, it can be used to detect the structural parts that make the class diagram less understandable.

3) Propose comprehensive understandability metric

The class diagram includes not only the class names, but also the role names, multiplicity, etc. We will study the influence that such information has on the understandability of the class diagram.

Moreover, the weights of the paths in the directed graph will be set according to the kind of relation, and thus, we will refine the proposed metrics.

### REFERENCES

- [1] Object Management Group: Unified Modeling Language Specification, Version 1.5, OMG document formal/03-03-01, 2003.
- [2] Reiko Arai: Self-review notebook for the Object-oriented design using UML, D ART, pp. 199, 2005 (in Japanese).
- [3] S.R. Chidamber and C.F. Kemerer: A Metric suite for Object-Oriented Design, IEEE Trans. Software Eng, vol. 20, no. 6, pp. 476-493, 1994.
- [4] F.B. Abreu, M. Gonlao and R. Esteves: Toward the Design Quality Evaluation of Object-Oriented Software Systems, Proc.5<sup>th</sup> International Conference on Software Quality, 1995.
- [5] H. Washizaki, Y. Kobayashi, H. Watanabe, E. Nakajima, Y. Hagiwara, K. Hiranabe and K. Fukuda: Experiments on quality evaluation of embedded software in Japan robot software design



- contest, International Conference of Software Engineering, pp. 551-560, 2006.
- [6] WordNet-Princeton University Cognitive Science Laboratory, <http://wordnet.princeton.edu/>.
- [7] Ted Pedersen - WordNet::Similarity, <http://wn-similarity.sourceforge.net/>.
- [8] Object Management Group: MOF 2.0/XMI Mapping, Version 2.1.1, OMG document formal/07-12-01, 2007.
- [9] Excite Translation, <http://www.excite.co.jp/world/> (in Japanese).
- [10] ET Robot Contest 2008, <http://www.etrobo.jp/ETROBO2008/index.html> (in Japanese).
- [11] SD Metrics - the design quality tool for UML Models, <http://www.sdmetrics.com/>.
- [12] J. Kato, M. Saeki, A. Ohnishi, M. Nagata, H. Kaiya, S. Furumiya, S. Yamamoto, and H. Horai: Supporting the development of a thesaurus for requirements analysis, Technical report of IEICE, KBSE 103(217), pp. 41-46, 2003 (in Japanese).
- [13] J.A. Cruz-Lemus, A. Maes, M. Genero, G. Poels and M. Piattini: The Impact of Structural Complexity on the Understandability of UML Statechart Diagrams, Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium, 2007.

## APPENDIX

TABLE III. THE DETAIL OF EXTRACTED MODELS

Team ID	Easy to read	Structure	Average of subjective evaluations	#Classes	#Association	#Inheritance
1	1	4	2.5	31	19	12
2	4	8	6	26	27	2
3	1	4	2.5	12	16	0
4	4	6	5	19	20	7
5	3	6	4.5	13	30	0
6	3	4	3.5	26	17	11
7	4	6	5	27	41	6
8	2	4	3	17	34	2
9	2	4	3	14	32	0
10	2	4	3	16	16	6
11	3	4	3.5	13	15	2
12	3	4	3.5	11	18	2
13	2	4	3	13	16	3
14	3	6	4.5	16	15	9
15	3	4	3.5	12	8	3
16	2	6	4	16	14	5
17	3	4	3.5	12	8	3
18	3	4	3.5	9	14	0
19	1	4	2.5	11	15	3
20	4	8	6	46	13	29
21	3	8	5.5	25	25	6
22	2	4	3	27	48	5
23	1	2	1.5	8	8	0
24	4	4	4	11	18	0
25	3	4	3.5	14	12	5
26	2	4	3	16	38	0
27	2	4	3	8	16	0
28	2	4	3	31	48	12
29	2	4	3	14	19	2