Rubric Creation Support System for Electronic Portfolio

Naoshi Sakamoto

***Abstract*—In order to improve education in university, e-portfolio is being introduced. An e-portfolio is an electronic aggregation of students' results. Using rubrics to evaluate students enables the e-portfolio to calculate the degree of students' various ability.**

In this study, we develop conditions for creating rubrics to aggregate, then, discuss that we can adopt principles of object oriented analysis to create rubrics. Moreover, we propose a rubric creation support system.

***Index Terms*—Rubric, e-portfolio, principles of object-oriented analysis.**

I. Introduction

Recently, it is being considered to introduce an e-portfolio (also known as an electronic portfolio, digital portfolio, or online portfolio) into education. An e-portfolio puts electrically students' study results together. This can be applied to check accomplishments, to review, and to plan learning goals. Moreover, this can also be used for quality assurance.

On the other hand, in Japan, MEXT, the Ministry of Education, Culture, Sports, Science and Technology of the Japanese government, told universities to open their diploma policy, which consists of necessary abilities to obtain the diploma.

There are several viewpoints of graduation of a university. One can obtain the diploma from a university;

1) if they attain all of abilities of which the diploma policy consists,
2) if they take many lectures, then write a graduation thesis,
3) or if they fill their e-portfolio with learning results of a standard quantity.

For equivalence among these viewpoints, the following conditions are required:

1) the diploma policy is a set of measurable abilities;
2) leaning results to put into an e-portfolio are divided, classified, and accumulated according to the diploma policy;
3) Each lecture has learning goals. Students are evaluated according to them. All learning goals of lectures form the diploma policy.

In each lecture, students are used to be evaluated by giving an integer number or a symbol, A, B, C, ..., that belong to the finite total order set. Recently, rubrics are introduced, which can evaluate in many-sided. A rubric is a clear scoring standard table. It lists the criteria established for a particular task and the levels of achievement associated with each criterion. It is often developed in the form of a matrix. A student passes a lecture, if they are evaluated and get at least one of success levels for every criterion.

	Level 0	Level 1	Level 2
Criterion 1	Quality Definition$_{1,0}$	Quality Definition$_{1,1}$	Quality Definition$_{1,2}$
Criterion 2	Quality Definition$_{2,0}$	Quality Definition$_{2,1}$	Quality Definition$_{2,2}$

Fig. 1. Rubric.

An e-portfolio puts results of lectures together for each student where lectures evaluate the student with rubrics, then aggregates results of criteria according to abilities of the diploma policy. It enables students to make sure how they develop their ability to be graduated.

In this study, we develop conditions that rubrics require, where the rubrics form an e-portfolio. Then we propose a prototype of a rubric creation support system for e-portfolio. Section II describes fundamentals of e-portfolios and rubrics, and related literature. Section III considers problems for creating rubrics for an e-portfolio. In Section IV, we formalize conditions for rubrics, and apply principles yielded by object-oriented analysis to create rubrics. In Section V, we propose a prototype of a rubric creation support system satisfying the discussed conditions. Finally, Section VI concludes.

II. Fundamentals of e-Portfolio and Rubrics and Related Works

A. Rubric

Rubrics are introduced as a standard to evaluate quality of students' responses [1]. A rubric contains evaluative criteria, quality definitions for those criteria at particular levels of achievement, and a scoring strategy. Levels consist of a failure and one or more successes. For each criterion and each level, a quality definition is defined (Fig. 1). To evaluate a student by using a rubric, for every criterion in the rubric, one quality definition is selected according to the student's response. A student is success when a quality definition of a success level is selected for every criterion.

Moreover, we can utilize rubrics not only to evaluate the students' ability, but also to apply to manage an organization, to graduate a department, and to describe the policy of the founding of a school [2].

Recently, online rubric creation tools are developed [3], [4].

B. E-Portfolio

An e-portfolio (also known as an electronic portfolio, digital portfolio, or online portfolio) puts students' electronic

Manuscript received August 15, 2017; revised November 28, 2017.

Naoshi Sakamoto is with the Department of Communication and Information Engineering, Tokyo Denki University, Tokyo, Japan (e-mail: sakamoto@c.dendai.ac.jp).

doi: 10.18178/ijiet.2018.8.6.1071

evidences together. An e-portfolios is both a demonstration of the user's abilities and a platform for self-expression.

Comparative research by M. van Wesel and A. Prop between paper-based portfolios and electronic portfolios in the same setting tentatively suggests that use of an electronic portfolio may lead to better learning outcomes [5].

In education e-portfolios have six major functions:

1) Document skills and learning;
2) Record and track development within a program;
3) Plan educational programs;
4) Evaluate and monitor performance;
5) Evaluate a course;
6) Find a job.

In this study, we focus on the case where an e-portfolio consists of rubrics. It can calculate how many credits a student obtains, since a rubric decides whether a student succeeds a lecture. Moreover, by aggregating criteria among lectures where the criteria concern the same ability, the e-portfolio can measure the students' ability corresponding to a criterion.

Originally, a curriculum is a set of lectures to satisfy the diploma policy. Then, a purpose of an individual lecture is to improve some students' abilities. By studying lectures one by one, a student improves their abilities gradually. Then, finally, if a student succeeds in all lectures in the curriculum, it is expected that they acquire the abilities of the diploma. Thus, by evaluating the students' ability by using rubrics, and by aggregating improvement of the ability by using an e-portfolio, we can visualize the improved ability, and give evidence that a student fulfills the diploma.

In order to realize such a function of e-portfolios, some consistency is required between the diploma policy of a department and the criteria of each lecture rubrics.

Some e-portfolios dealing with rubrics have already been provided [4]. However, they have no function about such consistency. Now, we discuss and analyze this consistency below.

III. Formulation on e-Portfolio and Rubrics

In this section, we develop conditions that an e-portfolio can give evidence that a student satisfies the diploma policy.

Basically, a diploma policy consists of the minimum requirements for abilities, where if a student obtains enough abilities that satisfy all of the minimum requirements, they are graduated.

A. Department Rubric

Here, we assume that we can create a department rubric according to the diploma policy of a department. It is expected that a diploma policy of a department is described as a list of abilities that students should obtain. Even though the abilities might concern each other, we would assume that abilities are independent each other for convenience's sake.

If there is no university rubric, we create a department rubric according to only the diploma policy of the department. We create one or more criteria corresponding to each ability in the policy.

On the other hand, if the university rubric has already been created, we create one or more criteria in the department rubric with inheriting each criterion in the university rubric.

In order that created department rubrics in above two situations are similar, the diploma policy of the department should inherit the diploma policy of the university. In this case, "inherit" means the followings:

1) the number of abilities in both policies is same,
2) the context of each ability of the department policy is similar to the context of the corresponding ability of the university policy, and
3) each ability is improved in the specialty of the department.

If *X* inherits *Y*, then we call *X* a child, and *Y* a parent. If nothing inherits *Z*, we call *Z* a terminal.

B. Lecture Rubric

A lecture rubric is basically corresponding to the learning goals described in the syllabus. A student is evaluated on the basis of the level of achievement for the learning goals. Thus, if we don't have to consider the department rubric, we can create a lecture rubric by creating criteria for each learning goal.

However, in order to put rubrics together into an e-portfolio to check how a student obtains the abilities for the diploma policy, we must create criteria that enable the e-portfolio to aggregate.

A department does not have only a diploma policy, but also a curriculum policy. That is, according to the curriculum policy, lectures are distributed. Thus, all of lectures distributed in a department must concern the diploma policy. Thus, at least one learning goal of each lecture must concern the diploma policy. Moreover, this can yield the fact that at least one criterion in the department rubric must concern at least one learning goal in each lecture. Let the number of criteria of a department rubric be *n*, and the number of learning goals of a lecture be *m*. Then, the number of combinations of them is *nm*. For *nm* combinations, at least one combination must concern. On the other hand, the maximum number of combinations to concern is *nm*. Thus, when we create criteria of a lecture rubric, we must probably examine all of combinations with each criterion in the department rubric and each learning goal.

IV. Algebra and Object-Oriented Analysis

We discuss conditions that a rubric satisfies where the rubrics form an e-portfolio.

A. Algebra on Rubrics for e-Portfolio

Let *S* denote a set of students, $L = \{0,1,...\}$ be the set of levels. Let 0 denote the fail level, and other elements denote success levels. For a criterion $c : S \to L$, when a student $s \in S$ is evaluated as $l \in L$, we denote $c(s) = l$. Let ${}^{S}L$ denote the set of whole criteria. We call a function $\varphi : S \to \{true, false\}$ a predicate.

For a set of elements *E*, we denote E^* as a set of whole series of elements of E with finite length. For $x \in E^*$, let $|x|$ denote the length of *x*. Let x_i denote the *i*-th element in *x*.

We also describe $x = (x_1,..., x_{|x|})$.

For predicates φ, and ψ, φ inherits ψ, if the following condition holds:

$$(\forall s \in S)[\varphi(s) \to \psi(s)]. \quad (1)$$

We denote this as $\varphi \rhd \psi$.

Proposition 1: $\rhd$ on predicates is a partial order.

For series of criteria $x, y \in ({}^{S}L)^{*}$, x inherits y, if the following condition holds:

$$(\forall s \in S)\left[\bigwedge_{i=1}^{|x|}(x_i(s) > 0) \to \bigwedge_{j=1}^{|y|}(y_j(s) > 0)\right] \quad (2)$$

We denote this as $x \rhd y$.

Proposition 2: $\rhd$ on series of criteria is a partial order.

We often consider a series as a set. That is, we denote the membership whether an element e belongs to a series s as $e \in s$. Moreover, $x \rhd y$ can be considered not only over series, but also over sets.

We assume a diploma policy *UDP* of a university consists of plural predicates as followings:

$$UDP = (\hat{\varphi}_1,..., \hat{\varphi}_n). \quad (3)$$

Let a university rubric *UR* be the following:

$$\begin{aligned} UR &= (uc_1,..., uc_n) \\ uc_1 &= (uce_{1,1},..., uce_{1,\hat{k}_1}) \\ &\cdots \\ uc_n &= (uce_{n,1},..., uce_{n,\hat{k}_n}) \end{aligned}$$

where $uce_{i,j} : S \to L$ is denoted as a criterion. The relationship between the diploma policy and the rubric as the following:

$$\hat{\varphi}_i(s) = \bigwedge_{j=1}^{\hat{k}_i}(uce_{i,j}(s) > 0). \quad (4)$$

Next, we focus on a department. We assume that the diploma policy *DP* of a department is based on the diploma policy of the university. Thus, *DP* can be described as follows:

$$DP = (\varphi_1,..., \varphi_n). \quad (5)$$

Here, we assume that the following relationships hold.

$$\varphi_i \rhd \hat{\varphi}_i \quad (i = 1,..., n). \quad (6)$$

On the other hand, let a department rubric *R* denote as follows:

$$\begin{aligned} R &= (c_1,..., c_n) \\ c_1 &= (ce_{1,1},..., ce_{1,\bar{k}_1}) \\ &\cdots \\ c_n &= (ce_{n,1},..., ce_{n,\bar{k}_n}) \end{aligned},$$

where each $ce_{i,j} : S \to L$ denotes a criterion.

Finally, we focus on a lecture. We can consider that a department consists of lectures. A lecture usually takes on the small responsibility of some part of the diploma policy of the department, partially. That is, a lecture gives a student small abilities for the department rubric. However, all of the required lectures and the lectures designated by the course plan finally satisfy both the diploma policy of the department and the department rubric. Thus, even if we define learning goals of *k*-th lecture as such a series of predicates as the following (7), the learning goals do not necessarily inherit the diploma policy, since learning goals are not the same as the diploma policy. Notice that learning goals are not independent from the diploma policy.

$$CP_k = (p_{k,1},..., p_{k,m_k}). \quad (7)$$

Let the lecture rubric of the *k*-th lecture be as follows:

$$\begin{aligned} CR_k &= (cc_{k,1},..., cc_{k,m_k}) \\ cc_{k,1} &= (cce_{k,1,1},..., cce_{k,1,l_{k,1}}) \\ &\cdots \\ cc_{k,m_k} &= (cce_{k,m_k,1},..., cce_{k,m_k,l_{k,m_k}}) \end{aligned}$$

where each $cce_{k,i,j} : S \to L$ is a criterion. Each series $cc_{k,i}$ of criteria is corresponding to a learning goal. On the other hand, the accumulation of criteria for all lectures in the department satisfies the department rubric. That is, for a criterion *ce* in the department rubric, there exists a set of criteria $C \subseteq {}^{S}L$ of the lecture rubrics such that $ce \rhd C$ holds.

B. Object-Oriented Analysis

In the previous subsection, we organize the relationships among policies, learning goals, and criteria of rubrics by introducing the operator for inheritance.

While we create rubrics, we have to pay attention not to yield a contradiction among them. After we grope the method how to create rubrics, we find that it is similar to the object-oriented analysis to create rubrics that have the relationship of inheritance. In this section, we discuss that we adopt principles of object-oriented analysis to create rubrics.

Here, we recall terminology of object-oriented development. In object-oriented development, we define a type of an object as a class, then classify classes by putting them into a package. A class consists of methods. A method consists of a signature and an implementation. If class *X* inherits class *Y*, then *X* inherits all of methods of *Y*. However,

X always inherits only the signature of a method. That is, we can override an implementation for an inherited method of *X*. On the other hand, if we define nothing for an inherited method of *X*, both the signature and the implementation of the method is automatically inherited.

At first, we reflect this discussion to the notion of rubrics. Then, we find the followings: The notion of packages is corresponding to the notion of rubrics and the notion of a series of criteria. On the other hand, the notion of classes is corresponding to the notion of criteria. When we create a rubric by inheriting another rubric, we should not revise its division greatly. On the other hand, when we create a department rubric from the university rubric, we revise so that inherited criteria improve its specialty.

Now, considering this analogy, we recall the package principles of object-oriented analysis as following [6]:

1) Principles of package cohesion
   Reuse-release equivalence principle (REP)
   Common-reuse principle (CRP)
   Common-closure principle (CCP)
2) Principles of package coupling
   Acyclic dependencies principle (ADP)
   Stable-dependencies principle (SDP)
   Stable-abstractions principle (SAP)

Let us consider these as the principles for rubrics and their series of criteria.

Moreover, object-oriented design principles (SOLID) are the followings [7]:

1) Single responsibility principle (SRP)
2) Open/closed principle (OCP)
3) Liskov substitution principle (LSP)
4) Interface segregation principle (ISP)
5) Dependency inversion principle (DIP)

Let us consider these as the principles for criteria in rubrics.

Now, we consider the principles for rubrics below.

C. Principles for Policies, Series of Criteria, and Rubrics

1) Principles of package cohesion

a) Reuse-release equivalence principle (REP)

In object-oriented analysis, REP essentially means that the package must be created with reusable classes — "Either all of the classes inside the package are reusable, or none of them are." The classes must also be of the same family. Classes that are unrelated to the purpose of the package should not be included. A package constructed as a family of reusable classes tends to be most useful and reusable.

By replacing terminology of rubrics, policies should be divided into units that are actually used in the inherited rubrics.

For example, suppose that a university rubric contains both the policy of differential and integral calculus and the policy of linear algebra separately where they are similar. However, they are usually inherited together. Thus, the separated policies only make the rubric complex. The original rubric should contain the single policy about both differential and integral calculus, and linear algebra.

On the other hand, if a rubric contains the policy about a fundamental subject to which mathematics and humanities are united, it is hard to deal with the rubric to inherit, since there might exist department that dealt with mathematics as a specified subject.

b) Common-reuse principle (CRP)

This principle is for making REP realize. This principle states that all of criteria in a series should be inherit totally.

For example, suppose that a university rubric contains a policy about whole lectures. Thus, department rubrics should deal with the policy so that the policy is divided into the policy of special subjects and the policy of fundamental subjects. The policy that would be divided when the rubric inherits should be divided in advance.

c) Common-closure principle (CCP)

This principle is also for making REP realize. This principle states that for one reason to change, changes should affect not to the plural policies, but to the single policy.

For example, suppose that a university rubric contains two policies that contain the same ability (e.g. the ability of communication), then, department rubrics had been created without specializing the ability. A student would be expected to obtain the ability of communication indirectly. Thus, there are more than one criterion about the ability in the department rubrics. Once a department added a lecture about the ability, then, the lecture would have to evaluate the ability. Thus, the rubric of the lecture should contain a criterion about the ability. However, the criterion about the ability in the rubric of the lecture would cover the corresponding criteria in the department rubric. Therefore, the common contents between policies should be reduced at the very least. For this example, in order to avoid this problem, we should describe the ability in the single policy, or standalone the policy about the ability.

2) Principles of package coupling

Since, we do not usually couple rubrics, we might not need the following principles.

1) Acyclic dependencies principle (ADP)
   A rubric should not refer any inherited rubrics.
2) Stable-dependencies principle (SDP)
   We should design parent rubrics to be stable. On the other hand, we should deal with terminal rubrics as if they are volatile.
3) Stable-abstractions principle (SAP)
   We should design so that a stable rubric is abstract, and a volatile rubric is concrete.

Therefore, by gathering SDP and SAP, we should design a university rubric so that it is abstract. On the other hand, we should design a lecture rubric so that it is concrete and is easy to change.

D. Object-Oriented Design Principles

For the principles of object-oriented class analysis, we discuss about designing criteria of rubrics.

1) Single responsibility principle (SRP)

A criterion should have only a single responsibility. Moreover, if a criterion should be changed, the number of reasons to change the criterion should be one.

For example, if there were a criterion that concern whole curriculum, it might be changed for every change such as changing a fundamental subject, addition of a specified subject, and so on. We should avoid designing a criterion covering over plural types of subjects.

2) Open/closed principle (OCP)

Criteria should be open for extension, but closed for modification.

For example, suppose that a university rubric contains a very concrete criterion such as "more than 60 points." Then, we cannot change any criteria in the inherited rubrics. Moreover, if a very concrete criterion of the university rubric were changed, every inherited rubric would have to be changed.

Thus, criteria of a parent rubric should be abstract so that they are merely changed, and we can override their criteria with more concrete criteria.

3) Liskov substitution principle (LSP)

For corresponding criteria between a parent rubric and a child rubric, the area of the evaluation of each criterion must be neither spread nor narrowed (be substitutionable).

For example, suppose that the university rubric contains a criterion about "having a good grounding in sciences and other," then by stretching our interpretation of "and other," we could replace any independent condition from the corresponding criterion about "having a good grounding in design" for "and other." However, we should avoid stretching our interpretation the meaning of criteria. On this principle, exchanging the criterion with the corresponding criterion of parent rubric causes changing the subject to evaluate. That is, stretching our interpretation is not specializing. Criteria should be abstract in parent rubrics. However, it should not contain words such as "and other."

4) Interface segregation principle (ISP)

This principle states that we should reduce supplementary items in the criteria. If a criterion had supplementary items, changing some of the supplementary items would cause changing every inherited rubric.

For example, suppose that a university rubric contains a criterion for experiment subjects such that "understanding the contents of Physics I through natural phenomena." Then, if the name of the subject were changed from "Physics I" to "Physics A," every inherited rubric should be changed. In this example, the criterion of the parent rubric should use more abstract key word such as "a fundamental natural science subject" instead of "Physics I."

5) Dependency inversion principle (DIP)

One should depend upon abstractions, [not] concretions. DIP states that criteria in parent rubrics should not depend on details.

For example, suppose that a university rubric contains a criterion about fundamental subjects, but concretely described the names of the subjects. Then, the criterion depends on changing of each described subject. This can be avoided by introducing an abstract key word such as "fundamental subject."

Note that in object-oriented analysis, increasing the number of classes and the number of packages are not bad. Thus, if we rashly obeyed these principles to create rubrics, this would cause increasing the number of policies and the number of criteria. In order to simplify a rubric, we also need another principle such as "small rubric principle."

V. Rubric Calculator for e-Portfolio

As discussed in the previous section, we can adopt principles of object-oriented analysis to create rubrics. Therefore, we expect to utilize an integrated development environment (IDE) for object-oriented programming to aid designing criteria of rubrics. In this section, we discuss about computer-aided rubric creation systems. Note that in lecture rubrics, we should examine the relationship with each criterion in the department rubric for each learning goal of the lecture. Thus, we discuss how a computer automatically aids this.

In integrated development environments for object-oriented programming such as Eclipse, when we create a new child class by extending the parent class, it supports to generate signatures of all required methods automatically. Then, the proposed rubric creation support system should also have such function. Note that since in programming, inheritance means recycling resources in a parent class, IDEs do not copy the implementations in the parent class to the child class. IDEs only copy signatures of methods.

Now, at first, we examine to utilize casual commonly used software such as Excel, databases, and UML tools to adopt to create rubrics. However, since it is required to increase many criteria when a policy is increased, the data structure used by the software such as table might be transformed greatly. This requirement is not impossible, but requires a complex macro. Moreover, it is said that using a macro causes not only vulnerability for operation, but also both difficulty for distribution and support. Therefore, we conclude that it is difficult to utilize any commonly used software to create rubrics. That is, we have to develop a specified software to create rubrics. Then, we design, and implement a rubric creation support system.

A. Design

The proposed software is to deal with a university rubric, a department rubric, and a lecture rubric.

According to the discussion in Section IV, a university rubric is highly abstract and stable. Moreover, it is actually described by a small-number of people, then is agreed by the university as a whole. Thus, we can assume that when we use the proposed software, we do not have to change the university rubric. Similarly, we assume that the diploma policy of the university is also defined in advance. Let each series of criteria in a university rubric have a description for the series. Let the name of series be A, B, ..., moreover, the name of criteria be A1, A2, ..., B1, B2, ... and so on.

We assume that the diploma policy is designed so that each policy is corresponding to a policy in the diploma policy of the university. We create a department rubric by inheriting the university rubric. Basically, each criterion of a department rubric is created by corresponding to the criterion in the university rubric one by one. However, for example, the department rubric for the department of information and the department rubric for the department of architecture are not the same. Each diploma policy expresses about specialty. Then, a criterion of a department rubric should be created by revising the corresponding criterion so that it increases specialty. Moreover, even though a department rubric is

enough abstract, it might be revised bottom-up while we create lecture rubrics. Thus, the proposed software copies the criteria of the university rubric at first, then enables us to change statement of the criteria.

Learning goals of a lecture can freely be added by itemizing, since a lecturer is allowed to set learning goals at their own discretion. Now, we consider a lecture rubric. Since we do not deal with the natural language process, our proposed system must be quite simple. By adding a learning goal, our system increase criteria as follows:

1) the proposed system copies all criteria of the department rubric;
2) then, connects the added learning goal to the bottom of the statement of each criterion.

This enables us to create statement of criteria easily since we can easily find how much each learning goal concerns with each criterion of the department rubric. Moreover, let the system be enabled to delete a criterion and to add an item independent from the department rubric.

Finally, in order to exchange the data to an office staff, the system should be able to save and load the data as the CSV file format.

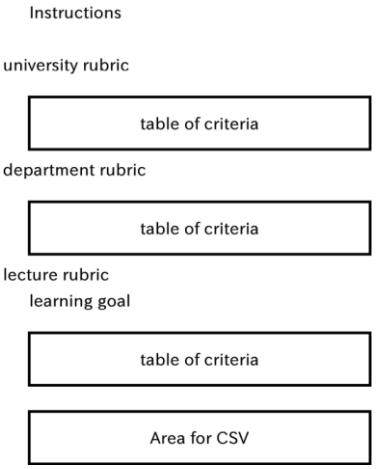

Fig. 2. Screen layout.

B. Implementation

We implement the proposed system by using HTML + JavaScript + CSS. This enables us to ease implementation, management, distribution, and maintenance. We display the layout of the screen in Fig. 2. For each rubric, the system consists of TEXTAREAs in each table, uses DOM to change the statements, and puts add buttons and delete buttons.

For the TEXTAREAs in the lecture rubric, in order to obey the change of department rubric, and for stability after creating the statement of criteria, when the statement is copied initially, "__edit_this__" is inserted between the statement of a criterion of the department rubric and the learning goals.

When the statement of a criterion of the department rubric is changed, the statement is substituted for only criteria containing "__edit_this__." On the other hand, when the delete button is pressed, the added criterion is simply deleted.

However, for the criterion corresponding to a criterion of the department rubric, only string "__nothing__" is inserted into the TEXTAREAa to ease to restore.

We display the screen shot in Fig. 3.

Fig. 3. Screen shot of the proposed system.

C. Demonstration

1) At first, we complete the department rubric. Revise the copied criteria in the department rubric so that each criterion increases the specialty according to the diploma policy of the department.
2) Next, we create the lecture rubric

Enter the name of the lecture.

Enter the learning goals. Once we enter a learning goal, the system generates a table that contains combinations with the learning goal and each criterion of the department rubric automatically.

For each combination, delete it if it is not necessary, or make a fare copy if it is effective.

Add other criteria if you have them to evaluate.

3) Generate the CSV format output data, copy them into notepad or other, then save them.

D. Evaluation

We can create a rubric actually by applying the proposed system. This system can help to create consistent rubrics by realizing the discussion about object-oriented analysis.

On the other hand, two professors in charge of developing academic education give comments to the system.

One points out that out system discloses problems about creating consistent rubrics.

Another points out as follows: object-oriented analysis is not universal. It can solve not so many actual models by applying simple inheritance relationship. More discussion is required as well as UML is extended to SYSML.

VI. Conclusions

In this study, in order to create rubrics for e-portfolio, we develop conditions that rubrics satisfy, then, discuss that we can adopt principles of object-oriented analysis to create a rubric. Moreover, we propose a rubric creation support system as a browser application.

Since our system is still a prototype, we have to improve our system.

Acknowledgment

The author thanks Prof. Tetsuo Shiotsuki and Prof. Kazuhiko Kudo for their useful comments.

REFERENCES

[1] T. S. Brophy, *Writing Effective Rubrics, University of Florida Institutional Assessment*.

[2] Civil and Environmental Engineering, College of Engineering, University of Delaware. ABET scoring rubrics for program outcomes. [Online]. Available: http://www.ce.udel.edu/ABET/Current%20Documentation/ABETscoring rubrics index.html

[3] Utah Education Network. The UEN rubric tool. [Online]. Available: http://www.uen.org/rubric/

[4] Data pacific (Japan) ltd. Web class. [Online]. Available: http://www.webclass.jp/

[5] M. Wesel and A. Prop, "The influence of portfolio media on student perceptions and learning outcomes," 2008.

[6] Wikipedia. Package principles — Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Package principles

[7] SOLID — Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/SOLID

Naoshi Sakamoto was born in Japan in 1964. He graduated from the University of Electro-Communication in 1987, and received his M.S. and Dr.S. degree from Tokyo Institute of Technology in 1989 and 2001, respectively.

From 1992 to 1997, he was a research assistant of Computer Center of Hitotsubashi University. From 1997 to 2001, he was a research assistant at Tokyo Institute of Technology. Since 2001, he has been an associate professor at Tokyo Denki University. Since 2014, he has been a professor. Then, since 2016, he is a deputy director of the office of educational development. He got the 1999 IEICE excellent paper award and the paper award of SNPD2015. His research interests are distributed algorithms, randomized algorithms, complexity theory, and educational technology.