# The Design and implementation of Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students : AEGIS

Mine, Tsunenori
Department of Intelligent Systems, Kyushu University

Suganuma, Akira
Department of Intelligent Systems, Kyushu University

Shoudai, Takayoshi
Department of Intelligent Systems, Kyushu University



出版情報：Proceedings of International Conference on Computers in Education, pp.651-658, 2000-11
バージョン：
権利関係：

# The Design and Implementation of Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students:AEGIS

**Tsunenori Mine  Akira Suganuma  Takayoshi Shoudai**
Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-kouen, Kasuga, 816-8580 JAPAN
E-mail:{ mine@is, suga@is, shoudai@i }.kyushu-u.ac.jp

**Abstract**

Many Internet technologies enable us to hold lectures with Web contents and even develop new lecture methods using the technologies. This paper proposes AEGIS (Automatic Exercise Generator based on the Intelligence of Students) that generates exercises of various levels according to each student's achievement level, marks his/her answers and returns them to him/her. In order to realize this feedback mechanism, we currently restrict the question-types which are generated to the following three types: multiple-choice question, fill-the-gap question, and error-correcting question. All question-types can be generated from the same tagged document. The aim of this system is to help the students understand the lecture with exploiting preexisting electronic documents.

**Keywords:** Artificial Intelligence in Education, Web-Based Learning, Exercise Generator

## 1 Introduction

As the Internet has come into wide use, WWW environments provide lots of opportunities to various fields. In the educational domain, Web data are being exploited as useful materials. We have been developing Web-based self-teaching systems and building the tools for helping students understand their subjects[1, 2, 3, 4].

We are currently focusing on the automatic student's achievement level evaluator that generates an exercise from tagged documents, presents it to students and marks their answer automatically. We call the system AEGIS (Automatic Exercise Generator based on the Intelligence of Students)[6, 7].

Creating exercises which are suitable for students is not easy. When we try to make some exercises for them in classes, we have to take at least their achievement level into considerations. The well-considered exercises are useful not only to measure the achievement level of students but also to improve their performance. It is not easy task for any teacher to make exercises of various difficulties according to their achievement level. Besides, it is very important to mark the students' answers and return the marked results to them for keeping their learning enthusiasms. This task becomes harder in proportion to the number of the students in a class[5].

This paper discusses AEGIS, which generates the three question-types from the same tagged data. Guessing the achievement level of each student from his/her trial history, AEGIS selects the most suitable question-type and exercise for him/her according to not only his/her achievement level but also the difficulty of the tagged data. After marking his/her answer, AEGIS returns it to him/her with its explanation.

The aim of this system is to exploit pre-existing electronic documents, in particular, our on-line documents shown at our Web site (http://cl.is.kyushu-u.ac.jp/Literacy) and to help students understand their lecture whose materials are set up as Web data so that they even at home can try exercises using AEGIS through the Internet.

The rest of this paper is constructed as follows: Section 2 shows related works to discuss the difference from AEGIS. Section 3 describes question-types that AEGIS deals with, considering both view points of students(answerers) and teachers(questioners) and Section 4 describes the exercise generating process by AEGIS. Section 5 shows the overview of AEGIS.

## 2 Related Works

A lot of automatic quiz generators have been proposed so far. Browning et. al. proposed Tutorial Mark-up Language(TML in short) to generate questions automatically[8, 9]. TML has a couple of tags to specify a question, a multiple-choice and a message. It requires a correct answer in a multiple-choice tag to mark a student's answer to the question. Carbone et. al. proposed CADAL Quiz[10], which generates a multiple-choice quiz from a question database. After marking a student's answer, CADAL Quiz returns the result to him/her and tutors. Both of them restrict the question type only to a multiple-choice quiz. On the other hand, ClassBuilder[11] generates many kinds of quizzes and grades a student's answer. However, all of them do not mention any effect of making the difficulty level of question-type change according to the students' achievement level. In order to improve their performance and keep their enthusiasm to challenge the quiz for a long time, it is indispensable to consider their performance level for generating their exercise. This point is the difference from other systems. AEGIS makes use of pre-existing electronic documents so as to embed tags into them, generates exercises automatically with tagged documents according to students' achievment levels, and reestimates both their levels and the difficulty level of the generated question through marking their answers.

## 3 Question-Types

There can be several types of a question in every subject. Since our aim is to get a computer generate an exercise and mark student's answer to it, we thus restrict to the following three question-types: multiple-choice question, fill-the-gap question, and error-correcting question.

**Multiple-choice question.** Students choose the correct answer from a given candidate list.

*Example.* Complete the sentence. Choose your answer from the following list.

Data structures need to be studied _____ order to understand the algorithms.

(1) an@@ (2) in@@ (3) on@@ (4) at@@ (5) by

**Fill-the-Gap question.** Students try to fill in the blank of a given sentence with the correct answer without any help.

*Example.* Fill in the blank with the right word.

Data structures need to be studied _____ order to understand the algorithms.

**Error-correcting question.** Students have to find the wrong expression in a given sentence and correct it.

*Example.* Right or wrong? Correct the sentence if it is wrong.

Data structures need to be studied an order to understand the algorithms.

All of these question-types can be constructed from a sentence by replacing one or more consecutive words with a blank or a wrong expression. We call the region replaced *hidden region*. We note that these three question-types have different difficulties even if they are constructed from the same *hidden region*. Figure 1 shows the tagged data to be used for generating the above three types of questions.

⟨QUESTION SUBJECT="idioms"⟩
Data structures need to be studied ⟨DEL CAND="an,on,at,by"⟩ in ⟨/DEL⟩ order to understand the algorithms.
⟨/QUESTION⟩

Figure 1: The tagged data to generate three question-types shown in Section 3

**Students' View Point**

Every multiple-choice question has surely the correct answer in its candidate list and contains the information that leads students to the correct answer. They can therefore make their choice with confidence from the list. In the case of a fill-the-gap question, they have to fill in the blank by themselves with their convinced answer without any information about the answer. Comparing both question-types, we can say

that a fill-the-gap question is more difficult than a multiple-choice one. In the case of an error-correcting question, it forces them to determine whether or not there is an error in the question sentences and to correct it if it is found. An error-correcting question gives no information leading them to its correct answer, and the wrong expression in the sentences is not clear for students. We can therefore say that an error-correcting question is the most difficult one for students among those question-types.

**Teachers' View Point**

Once teachers set a *hidden region*, the efforts that are required to make with the three question-types are similar. The process for making exercises is as follows: in the case of a fill-the-gap question, the teachers have nothing to do. There is no information that they have to add to the exercise paper. We can say that a fill-the-gap question is the easiest one which is made among these three question-types. In the case of an error-correcting question, teachers have to think of at least one wrong expression which can be replaced with the *hidden region*. In the case of a multiple-choice question, they have to prepare several distractors to construct a candidate list. We can say that a multiple-choice question requires more information than an error-correcting one. From their points of view, a fill-the-gap question is consequently the easiest one which is made, and an error-correcting question is easier than a multiple-choice one.

# 4 Automatic Exercise Generating

## 4.1 Exercise Generating Process

The exercise generating process from teaching documents is summarized as follows:

1. Setting a *hidden region*: teachers make clear their intention why they want to ask the question to their students, that is, they consider which of the *hidden region*s is the most suitable for their intention.

2. Selecting a paragraph or sentence(s) from teaching documents: the sentences before and after *hidden region*s are often of importance to ask their students the unique answer of the question. We call the paragraph or sentence(s) a *question region*. A *question region* may have more than one *hidden region*.

3. Constructing a candidate list: a multiple-choice question requires a couple of distractors to set up a list of answer candidates. Any distractor should be natural so as to be added to the list. This list depends on the teacher's intention.

These three steps are deeply related to the teachers' intentions. It is not easy to extract such intentions automatically from the teaching documents. AEGIS system thus deals with tagged documents that already have the information such as *hidden region*s and candidate lists.

## 4.2 Necessary Information for Generating Exercises

In order to embed the above three kinds of information into the teaching documents, we define the following three tags: QUESTION, DEL, and LABEL.

QUESTION surrounds a *question region*, that is, the statements between ⟨QUESTION⟩ and ⟨/QUESTION⟩ are a *question region*. In the region, there can possibly be some expressions that are related to a *hidden region*. They can be good hints to lead students to the correct answer.

SUBJECT is the unique attribute of QUESTION. Its value stands for the subject or topic of *question region*.

DEL indicates a *hidden region*, which is the word(s) or sentence(s) between ⟨DEL⟩ and ⟨/DEL⟩.

A fill-the-gap question can be generated only by replacing the *hidden region* with a blank.

CAND is one of DEL's attributes. It is used to specify a candidate list.

LABEL has an attribute NAME that specifies a dependency relation with a *hidden region*. The sentence/s surrounded by LABEL tags is/are presented as a reference for the answer of a question, which will be generated with the DEL tag whose REF's value is the same as that of the NAME of the LABEL.

| | | |
|---|---|---|
| ⟨QUESTION SUBJECT="W_S"⟩ *question region* ⟨/QUESTION⟩ | | |
| W_S | ::= | word or symbol, where a backslash (\) must be added just before the symbol if it is a comma (,), double quotes ("), or a backslash (\). |
| ⟨DEL CAND="CANDIDATE" LEVEL="PAIR" GROUP="ID" REF="ID"⟩ *hidden region* ⟨/DEL⟩ | | |
| CANDIDATE | ::= | W_S \| W_S,CANDIDATE |
| W_S | ::= | word or symbol, where a backslash (\) must be added just before the symbol if it is a comma (,), double quotes ("), or a backslash (\). |
| PAIR | ::= | LOW,HIGH |
| LOW | ::= | an integer between 1 and 10 |
| HIGH | ::= | an integer between 1 and 10 |
| ID | ::= | keyword |
| ⟨LABEL NAME="ID"⟩ sentences ⟨/LABEL⟩ | | |
| ID | ::= | keyword |

Figure 2: Tags for exercise generations

### 4.3 Necessary Information for Adjusting Difficulty Level of Question

The additional three attributes of DEL, which contain the information on the difficulty of solving the exercise, are LEVEL, GROUP, and REF. They specify the difficulty of each *hidden region*, and the connections to other *hidden region*.

LEVEL specifies the difficulty of the exercise to be generated from a *hidden region* itself. The value of this attribute is a pair of integers between 1 and 10. These integers specify the lowest and highest achievement level of the students who can try the exercise. AEGIS system determines whether or not the *hidden region* is worth being transformed into the exercise by comparing the student's achievement level from the both values of LEVEL.

GROUP specifies the dependency relation between *hidden region*s and holds the uniqueness of the correct answer. This GROUP is used to adjust the exercise level. If we want to generate more difficult exercises, all the *hidden region*s that have the same values in GROUP are replaced with blanks or wrong expressions at the same time. On the other hand, for generating easier ones, some of the *hidden region*s in the group are not transformed because those regions help students answer the question as hints.

REF specifies the dependency relation between a *hidden region* and other expressions than the *hidden region*. Both the region and expressions are specified with LABEL. If a *hidden region* is connected to an expression, the value of REF in the *hidden region* is the same as that of NAME in the expression with LABEL.

## 5 AEGIS system

### 5.1 Overview of AEGIS

The AEGIS system consists of three databases: *Exercise DB* (**EDB** in short), *User Profile DB* (**UPDB** in short) and *Level Management DB* (**LMDB** in short), and three main database managers: *Exercise Generator* (**EG** in short), *Answer Evaluator* (**AE** in short) and *Level Manager* (**LM** in short). The overview of AEGIS is shown in Fig. 3.

Teaching documents with the tags are compiled into the **EDB** and **LMDB**. All of the *question region*s are indexed sequentially and each *hidden region* is labeled with its own subindex of the index of each *question region*. The level of a *hidden region*, which is deeply related to the level of the question to be generated from the *hidden region*, is stored in the **LMDB** together with the index of the *hidden region*. The level of each *hidden region* in **LMDB** is reexamined regularly. **UPDB** keeps students' trial histories with their current achievement level.

**EG** and **AE** make communications with the users (students) through Web browsers after being invoked through CGI (Common Gateway Interface).

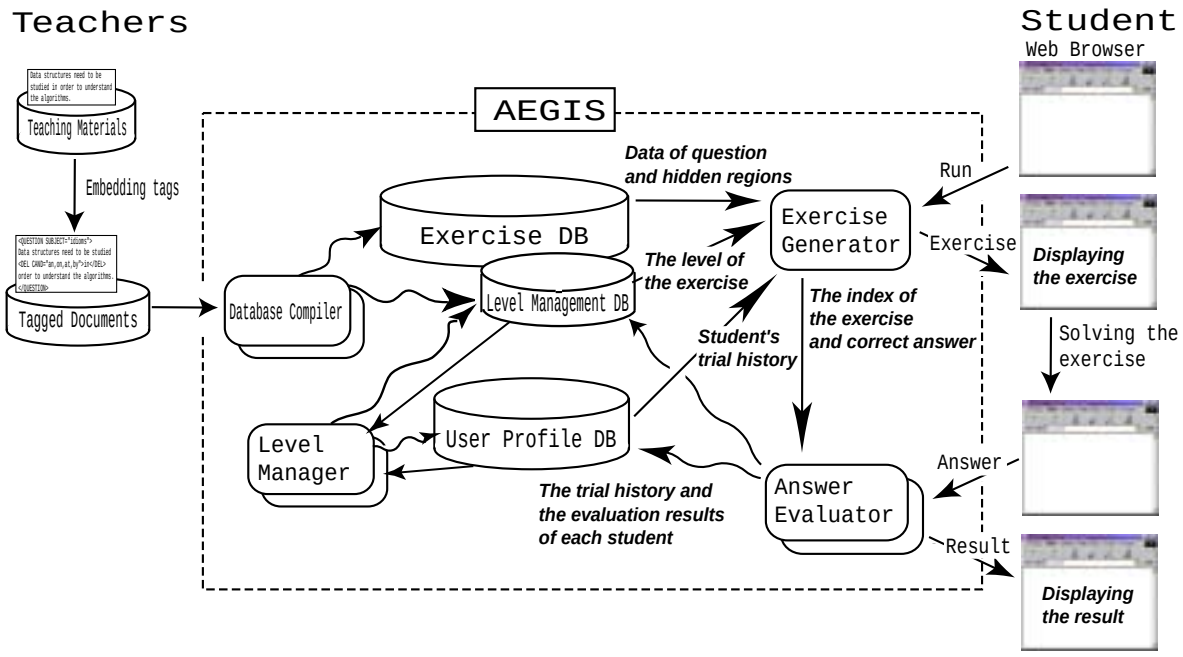


Figure 3: Overview of AEGIS

## 5.2 Exercise Generator(EG)

The exercise request from a student invokes **EG**. The **EG** searches the most suitable *hidden region* in **EDB** with looking over both the student's profile stored in **UPDB** and the level of the *hidden region* stored in **LMDB**, and determines the question-type of the *hidden region*. As mentioned in section 3, every question level has a relation to the question-type. **EG**'s decision process of the question-type thus employs the following strategy: If the student's achievement level is closer to the lowest number in LEVEL of the *hidden region*, **EG** selects a multiple-choice question as the question-type with high probability. On the other hand, if it is closer to the highest number in the LEVEL attribute, **EG** selects an error-correcting one.

Once **EG** determines the question-type of the *hidden region*, it is not difficult to generate the question. This is because the *hidden region* represents the correct answer of the question which is generated and teachers have already given the list of distracts explicitly with CAND attribute. Now, let's see how **EG** works when it generates the three kinds of questions:

- *Multiple-choice question*: **EG** randomly constructs one possible list for the multiple choice with both the correct answer and some distracts and outputs a question, which is generated by replacing the *hidden region* with a blank, with the list.
- *Fill-the-Gap question*: **EG** outputs a question which is generated only by replacing the *hidden region* with a blank.
- *Error-correcting question*: **EG** outputs a question which is generated by replacing the *hidden region* with one of the wrong answers specified in the CAND attribute.

Figure 4 shows an example of teaching documents with the tags. It is a piece of the teaching documents in the elementary course of Computer Literacy at our university. This course is taken by all first and second year students, about 2,300 students[5]. The teacher's intention in the example document is to teach how to use multiply and divide operations. Figure 5 shows the three question-types which are generated from the document.

## 5.3 Answer Evaluator(AE)

After outputting a question to the student, **EG** sends the following three kinds of information to ask **AE** to mark his/her answer: the index of a *hidden region*, the question-type, and the correct answer. After

In the previous section, we learned a program for adding two integers and showing the answer on the display. In the similar way, for all basic arithmetic operations including addition, subtraction, multiplication, and division, we can make a Pascal program in the following way.
⟨QUESTION SUBJECT="arithmetic operations"⟩
This program computes the multiplication and division for two input integers and shows the answer.

```
program enzan;
var x,y:integer;
    seki,shou:integer;
begin
    write('Input two integers : ');
    readln(x,y);
    seki:=⟨DEL CAND="x,xy,x×y,x mul y" LEVEL="1,5"⟩x*y⟨/DEL⟩;
    shou:=⟨DEL CAND="x/y,x÷y,xdivy,x mod y" LEVEL="1,5"⟩x div y⟨/DEL⟩;
    writeln('Seki:',seki);
    writeln('Shou:',shou)
end.
```

⟨/QUESTION⟩
The 7th statement multiplies x by y, and the 8th statement divides x by y. We note that the answer of "div" is an integer.

Figure 4: Example of teaching documents with the tags

(a) Multiple-Choice (b) Fill-the-Gap (c) Error-Correcting

Figure 5: Three questions generated from the document in Figure 4

marking his/her answer by matching with the correct answer, **AE** shows him/her the marked result and stores it with the index of the *hidden region* and the question-type into the **UPDB**.

## 5.4 Level Manager(LM)

Although the initial value of the level of each *hidden region* is specified by teachers, it continues to move up and down according to the students' achievement levels, which will change as time goes by. The supplement manager **LM** processes their achievement levels statistically, computes the revised level of each *hidden region*, and stores it into the **LMDB**. **LM** increases the difficulty level of a question if a student whose level is greater than the level of question answers it wrongly, and decreases if a student whose level is less than the level of question answers it correctly. The new difficulty level of a question is consequently determined as shown in Fig.6.

After updating **LMDB**, **LM** updates the student's achievement level according to the difficulty levels of all questions he/she correctly answered.

Now, we show the formal definition of calculating both the achievement level of a student and the difficulty level of a question. Let $s_{i,t}$ and $q_{j,t}$ be the achievement level of student $S_i$ and the difficulty level of question $Q_j$ at time $t$ respectively, where $1 \leq s_{i,t} \leq 10$ ,$1 \leq q_{j,t} \leq 10$. $s_{i,t}$ is recursively calculated with $q_{j,t}$ at stated periods and vice versa. They are defined as follows:

$$s_{i,t} = \begin{cases} 1 & \text{if } m_{s_i,t} = 0 \\ \frac{1}{m_{s_i,t}} \sum_{j=1}^{m_{s_i,t}} q_{j,t} \cdot \delta_{i,j} & \text{otherwise} \end{cases} \qquad \delta_{i,j} = \begin{cases} 1 & \text{if } S_i \text{ answered } Q_j \text{ correctly} \\ 0 & \text{otherwise} \end{cases}$$

$$q_{j,t} = \begin{cases} q_{j,t-1} + \frac{\sum_{i=1}^{m_{q_j,T}} |s_{i,\tau \in T} - q_{j,t-1}| \cdot \xi_{i,j}}{\sum_{i=1}^{m_{q_j,T}} |\xi_{i,j}|} & \text{if } \sum_{i=1}^{m_{q_j,T}} |\xi_{i,j}| \neq 0 \\ q_{j,t-1} & \text{otherwise} \end{cases}$$

$$\xi_{i,j} = \begin{cases} -1 & s_{i,\tau} \text{ is less than } q_{j,t-1} \text{ and } S_i \text{ answered } Q_j \text{ correctly} \\ 1 & s_{i,\tau} \text{ is greater than } q_{j,t-1} \text{ and } S_i \text{ answered } Q_j \text{ wrongly} \\ 0 & \text{Otherwise} \end{cases}$$

Where $m_{s_i,t}$ stands for the number of questions that $S_i$ tried by $t$ and $\tau$ is the latest time such that $S_i$ tried to answer $Q_j$ and $t-1 < \tau \leq t$. $T$ is the set of $\tau$. $m_{q_j,T}$ stands for the total number of students who tried $Q_j$ in $T$. $q_{j,0}$, which is the initial difficulty level of the question $Q_j$, is given with the attribute $LEVEL$ of $DEL$ tag by teachers.

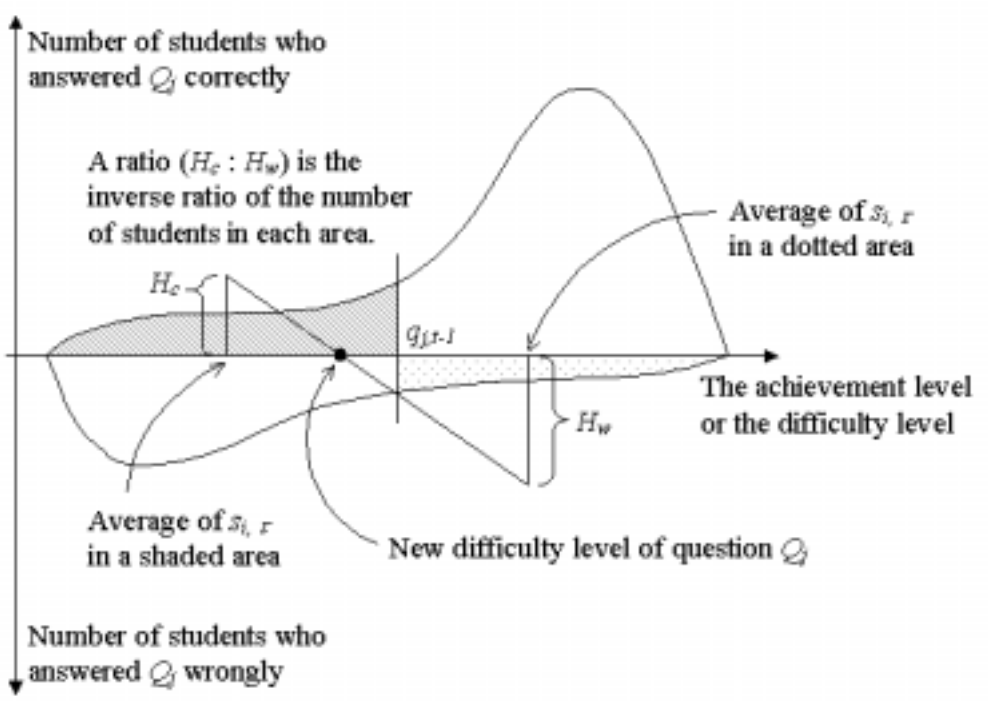


Figure 6: Renewing Difficulty level of Question based on Student's Achievement Level

# 6 Conclusions

We discussed our new Web-aided system AEGIS. The system is currently implemented in Perl scripts and CGI. We have a plan to evaluate this system by applying it to the real courses of Computer Literacy, which are taken by more than 2300 students at our university. We hope it will work fine as an educational tool for every student and help him/her to understand his/her subjects if teachers can make tags in their teaching documents. Also, we plan to implement a tagging tool and an algorithm to generate another kind of exercise that allows more than one correct answers.

# Acknowledgments

We thank Dr. Teruko Mitamura at Carnegie Mellon University for her helpful comments to improve this paper.

This research was partly supported by the Grant for Special Academic Research P&P from Kyushu University and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Encouragement of Young Scientists(A), No. 11780281, 2000.

## References

[1] H. Sato, T. Mine, T. Shoudai, H. Arimura, S. Hirokawa: On web visualizing how programs run for teaching 2300 students. In *Proc. Int. Conf. on Computers in Education, ICCE'97*, pages 952–954, 1997.

[2] T. Mine, D. Nagano, K. Baba, T. Shoudai, S. Hirokawa: On-web visualizing a mechanism of a single chip computer for computer literacy courses. In *Proc. Int. Conf. on Computers in Education, ICCE'98*, volume 2, pages 496–499, 1998.

[3] R. Fujimoto, A. Suganuma, T. Mine: Development of a classroom management system on the web. In *Proc. Int. Conf. on Computers in Education, ICCE'99*, volume 2, pages 756–759, 1999.

[4] R. Fujimoto, Y. Tsutsumi, A. Suganuma: Cacce: Computer aided cooperative classroom environment. In *World Conference on Educational Multimedia, Hypermedia & Telecommunications*, page 1686, 1999.

[5] S. Hirokawa, T. Miyahara, T. Mine, T. Shoudai, M. Mori: Teaching 2300 students with www – practice and experience at kyushu university. In *Proc. ERI'96*, pages 59–63, 1996.

[6] T. Shoudai, A. Suganuma, T. Mine: AEGIS: Automatic Exercise Generator with Tagged Documents based on the Intelligence of the Students, Joint Conference on Knowledge-Based Software Engineering, to appear. JCKBSE2000, Brno, Czech Republic, 2000

[7] T. Mine, T. Shoudai, A. Suganuma: Automatic Exercise Generator with Tagged Documents

Considering Learner's Performance Proceedings of WebNet2000, San Antonio, Texas, to appear as a short paper, 2000

[8] P. Browning, J. Williams, D. Brickley, H. Missou: Question Delivery over the Web using TML http://www.ilrt.bris.ac.uk/netquest/liveserver/qbanks/demos/paul/lboro/ohp1.html, CAA, 1997

[9] P. Browning: TUTORIAL MARKUP LANGUAGE - A CBA SYSTEM http://www.soton.ac.uk/~ukgec/workshop/5-cba/minutes.htm#TUTORIAL, 1998.

[10] A. Carbone, P. Schendzielorz: A Web-Based Quiz Generator for Use in Tutorials and Assessment Global J. of Engng. Educ., Vol.I, No.3, 1997 http://www.eng.monash.edu.au/usicee/gjee/vol1no3 /paper20.htm.

[11] ClassBuilder GradeBook And Exam Creation Software: http://www.classbuilder.com/