Chapter 10
Using a Programming Exercise Support System as a Smart Educational Technology

Toshiyasu Kato, Yasushi Kambayashi and Yasushi Kodama

Abstract During the completion of programming exercises at higher educational institutions, students typically must complete the assigned exercise problems individually. While there are certain students who can easily solve the problems independently, many students require too much time to do so. For this reason, most institutions use teaching assistants, or TAs, to help teach programming classes. In this chapter, we propose support functions to assess the learning conditions of a programming practicum. The aim of these functions is to reduce the burden on instructors by supporting the assessment of learning conditions in order to improve the quality of instruction. In the current study, we have designed support functions to assess learning conditions for a support system. We also conducted experiments in actual classes to assess the results. We propose three functions for a programming exercise support system. The first function is to support teachers with the analysis of students with common problems. The second function is to support teachers with the analysis of students who are having difficulties. The third function is to provide TAs with the features of students' programming behaviors. We have developed smart educational environments through these three functions of our programming exercise support system. The system has successfully supported instructors and TAs in their provision of smart pedagogy for students.

Keywords Programming exercises · Learning conditions · Instructor assistance · Face-to-face class · Smart education · Learning analytics

T. Kato (✉) · Y. Kambayashi
Nippon Institute of Technology, Saitama, Japan
e-mail: katoto@nit.ac.jp

Y. Kambayashi
e-mail: yasushi@nit.ac.jp

Y. Kodama
Hosei University, Tokyo, Japan
e-mail: yass@hosei.ac.jp

V.L. Uskov et al. (eds.), *Smart Universities*, Smart Innovation, Systems and Technologies 70, DOI 10.1007/978-3-319-59454-5_10

10.1 Introduction

Higher education institutions use various learning management systems for laboratory-style lessons [1, 2]. In addition, researchers have conducted studies to analyze the learning histories stored in such learning management systems [3, 4]. When doing programming exercises, students independently complete the assigned problems. While there are certain students who can easily solve the problems independently, there are many students who require much time to solve the same problems [5]. In order for an instructor to effectively support students' problem solving, the instructor is expected to not only answer the questions from the students, but also understand which particular students really need help [6]. The instructor needs to understand which students need assistance, what kind of assistance is needed, and in what situations it will be most effective [7].

As for the current state of programming exercises for entry-level classes, the number of instructors and the number of TAs are limited [8]. The instructor can recognize the learning situations of students from observing their computer screens; however, accurately assessing the work progress of the class as a whole as well as each student's individual learning needs are two completely different tasks. Assessing student's individual learning needs is difficult to achieve. Instructors can encourage students to raise their hands to indicate their need for help or check student submissions of the assigned problems. This method is inefficient, though, because it takes a lot of time when the number of students in the class is large.

This chapter presents the learning situation awareness functions of the Web-based learning management system for the implemented programming exercises. This function enables instructors to assess the learning situation of each student. This cannot be understood by simply viewing a student's computer screen. The proposed functions present the information necessary for the instructors to guide each student individually. The instructors and TAs can assess the learning needs of each student and give appropriate guidance according to the information that the function provides.

In order to implement the given function, we performed a requirements analysis of the programming exercise designated for learning situation awareness. Next, we executed the design and implementation of the function on the basis of previous investigations. The assessment of student learning situations is generally performed in a face-to-face manner. Therefore, the present study extracted the functions from the requirements analysis that assesses the student learning situations during a regular, face-to-face class. In addition, the authors clarified that the problem of the present study is the techniques used in previous studies. The present study requires the use of a learning management system that can collect student learning history data in real time.

10.2 Literature Review

Kurasawa [9] developed a support system for assessing learning conditions that provides instructors with information regarding students with common problems. It collects the compiler history and, from the analysis of trends in past compiler errors, it estimates and aggregates the locations and causes of errors. By supplying the instructor with error messages, the number of errors, their causes, and error syntax, the instructor can receive information on students' common problems. This, though, requires analyzing and preparing error cause candidates beforehand and, as the instructor must analyze error causes, it adds to the workload of the instructor. Moreover, it is limited to whole-class instruction and does not provide information regarding individual students. Individual learning is fundamental in programming practicums and information necessary to individual instruction, such as student logs and repeated mistakes, is indispensable. In the present study's proposed method, error analysis takes place automatically. There is no need for the professor to do any work. Furthermore, it assesses and displays assessments of common student problems as well as individual solution histories to allow for both class and individual instruction.

Higher education institutions could harness the predictive power of CMS data to develop reporting tools that identify at-risk students and allow for more timely pedagogical interventions, as well as which student online activities accurately predict academic achievement [1].

Colthorpe [10] investigated the relation between the presence of self-reflection based on material access and the report date of Learning Management System (LMS) submission. Students that reported reviewing lectures as a learning strategy were more likely to access the online lecture recordings, but higher access was actually associated with poorer academic performance. Cluster analysis of all available data showed high academic performance was positively associated with early submission of intra-semester assessment tasks but negatively associated with both the use of, and the reported of use of, lecture recordings by students. Therefore, using an online test enables more realistic feedback.

Research on the assessment of programming behaviors includes debugging training, programming tutorials, and error analysis. Ryan [11] studied how to improve student debugging skills. He constructed a model from debugging and development logs. He also found that students could improve their debugging skills using the model. The present study also analyzes debugging logs and other programming records.

Alex [12] proposed employing tutors to support students in developing programs step by step. The tutors examined students' processing and monitored whether they were on the right track. Students received advice when their programs were incorrect. For example, the tutor provided hints to students on how to refactor their programs. The present study builds upon this work by using TAs to provide students with problem-solving techniques to solve their own problems.

Serral presented a synthesis and update of a long-term project that addressed this challenge in the context of conceptual modeling by developing SAiLE@CoMo, a smart and adaptive learning environment [13]. By crafting innovative process analytics techniques and expert knowledge on feedback automation, SAiLE@CoMo automatically provides personalized and immediate feedback to learners. This research will deal with the problem of feedback deficits reported by students and will significantly alleviate teacher effort. This leaves the instructor more time for in-depth discussions with students.

Truong's paper described a "fill in the gap" programming analysis framework that tests students' solutions and gives feedback on their answers, and detects logic errors and provides hints as to how to fix them [14]. The framework makes use of client-server communication architecture. This is where the execution of students' programs takes place on their own machines while the evaluation is carried out on the server. With this framework, teachers can immediately confirm changes to program sources.

In light of the abovementioned research, we are able to assess student work progress from the start of the exercise to the submission of their work. The purpose of this is to find students with ongoing errors at an early stage. In addition, we will be able to grasp the compilation errors occurring throughout the entire class without requiring faculty error analysis. The reason for this is that the teacher must analyze all of the errors if a single student generates multiple errors at once. Furthermore, we are able to notice the students who are behind in their work in relation to the whole class. This can help find and instruct students who are not progressing even if no error has occurred.

Smart education is rapidly gaining popularity among the world's best universities because modern, sophisticated smart technologies, smart systems, and smart devices create unique and unprecedented opportunities for academic and training organizations to improve their educational standards [15].

The smartness level here is the discovery of students experiencing difficulties with a focus on assessing individual learning situations in detail.

10.3 Research Design and Research Objectives

A teacher can more accurately assess student progress on learning tasks and provide feedback when using smart educational technology. This creates a smart learning environment [16] for students.

10.3.1 The Problems of Programming Exercise Support

In programming exercises, each student must individually tackle the assigned problems. The progress of each student is very different and depends on their skill

levels [5]. An inefficient programming mode is generally the main reason a particular student cannot progress. This student occasionally comes to a standstill and does not know what to do next. Such students can often become unmotivated [7]. In programming exercises, it is important to identify these students during the early stages of the exercise.

In the programming exercise, the instructor and TAs go around to students' consoles. Even in this situation, it is not easy for the instructor and TAs to determine which students need extra assistance [17]. The instructor and TAs check students' progress through submitted programs and the students' screens; however, the number of TAs is limited. They are always too busy to check simple errors. We have to rely on the instructor's teaching experience and intuition to solve the problems of individual students [7]. The instructor and TAs answer students' questions as needed. The instructor teaches the class based on the information they have collected from individual student cases. The instructor must advise students when it comes to challenge topics and how to understand the learning method [17].

The objective of the learning management system for programming exercises is to reduce the workloads of the instructor and TAs [18, 19]. The function of the system is composed of the functions of the instructor and students. Figure 10.1 shows the basic functions of the programming exercise support system.

- Functions of the instructor:
  - Exercise making: the question contents and the date of setting questions are input and registered.
  - Exercise demonstration: the specified exercise at the date of setting questions is presented.
- Functions of the students
  - Exercise receipt: the exercise is selected and the development of the answer begins.
  - Program edit: the answer program to the exercise is edited.

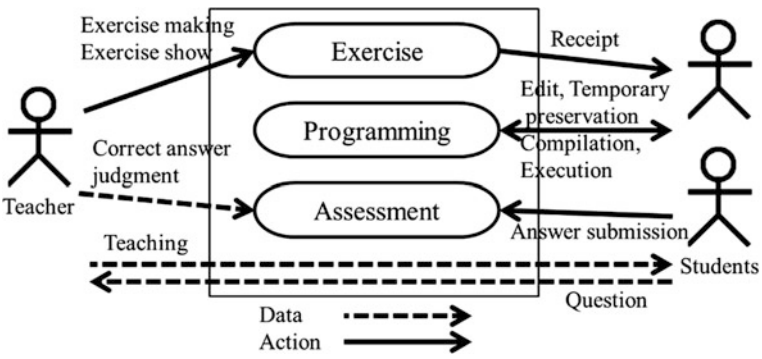


Fig. 10.1 Functions of the programming exercise support system

- Temporary preservation: the answer program is temporarily preserved.
- Compilation: the answer program is compiled and it presents the results.
- Execution: the answer program is executed and presents the results.
- Answer submission: the answer program is submitted and the exercise is complete.

10.3.2 Request Analysis for Smart Classroom Realization

In order to utilize a smart classroom for programming exercises, the following three principles of instruction should be followed [20, 21]:

(1) Assess each individual student's progress toward the learning target. The objective is to consider subsequent lesson plans for each student based on his or her current problems.
(2) Assess each individual student's understanding of the learning content. The objective is for the instructor to recognize the exact area where the student is having difficulty so that he or she can arrange the guidance contents.
(3) Assess each individual student's difficulties. The objective is for the instructor to recognize where each student is having problems so that appropriate instruction can be provided.

As for the proposal of the present study, we set three functions listed below on the basis of the principles of pedagogy.

Work progress sum function. The work progress sum function presents the achievement situation of the learning target as work progresses in the exercise. The learning target of the programming exercise is to solve the exercise. The achievement situation refers to the advancement towards execution and compilation of the exercise. The work of the programming exercises is as follows:

1. The answer begins
2. Input and compilation of the program
3. Confirmation of compile errors
4. Execution
5. Confirmation of execution results
6. Submission of exercise

The reason for this function is to understand the work progress of students on exercises in class. Moreover, the function should be suitable for the intention of the exercise. This function checks the answers and detects mistakes in the execution results.

Error classification sum function. The error classification sum function assesses student understanding of the learning contents as an error status of the class. The reason for this is that the error of understanding of the shortage and the

programming is the same. The goal of this function is to identify any common errors occurring in the class.

Work delay detecting function. The work delay detecting function shows delays in the completion of students' work. The purpose of this function is to inform instructors of which students are slow or late in completing the exercise.

10.4 The Function of Assessing the Learning Situation of the Class

This function is a work progress sum function and an error classification sum function. We first designed the function, then defined the algorithm, and then completed the mounting.

Work Progress Sum Function

Function design

The objective of the work progress sum function is to present the number of students that have completed the exercise at any time. The previous work is specified for one work [12]. Therefore, the technique was not applicable to the entire exercise. The present study was designed to assess student progress at each step of the exercise. The proposal function presents the sum result at each work completion time. The instructor modifies the instructions based on the display contents. Therefore, the contents include work completion time, matriculation number, name, and seat number. This information is input at the login of the learning management system.

Work completion with correct answers is defined by the presence of the correct answer according to the correct answer judgment. The correct answer judges the student's execution result, the instructor's execution result, and the keyword. This is done according to the timing of the students' execution of the program. The function then presents the presence of the correct answer or the incorrect answer and the keyword. The work progress sum function calculates the number of incorrect answers and the number of correct answers. The instructor only prepares the example answer program and the keyword. As shown in the following example of correct answer judgment, the correct answer is "for" in the answer program and 55 of the execution result. "The total from 1 to 10 is output by the use of the 'for' sentence." This correct answer judgment can be applied to the exercises that obtain the output result and ask for the grammar.

Algorithm

1. The function initiates the processing by the instructor's access.
2. Each work completion time of the class set is input.
3. The number at each work completion time is totaled.
4. The total number of each work is output.

User Interface

Learning situation screen (displayed in Fig. 10.2): This shows data for each student's work at the beginning, compilation, execution, correct answer, and answer submission stages of the exercise. When the number of presented items is clicked, it shifts to the learning situation screen according to the classification.

Learning situation screen according to classification (displayed in Fig. 10.3): This shows students' matriculation numbers, names, seat numbers, and work completion times that come under each item of the learning situation screen.

Error Classification Sum Function.

Function design

The objective of the error-classification sum function is to present the result of the error classification from the student's program and the error message of the compilation. The previous work will prepare the error factor and the error pattern [9, 11]. Therefore, the present study was designed to identify student compilation errors without requiring these analyses. The proposal function presents the sum result of the error classification. The error classification presumes the place of a common compile error. The number at the head of the error corresponds to the line number of the example answer program (hereafter, correspondence line number). Because the first error leads to other error factors, the error of the head is targeted. The object language of the error classification uses the same Java language as the lesson of the assessment experiment. The analysis object of the error classification is a compile error of the student who does not arrive at execution. This is because it acquires the error that occurs when performing the function. The reason to assume the analysis object to be a compile error is that there is a necessity for the guidance of the instructor in the programming exercise [7].

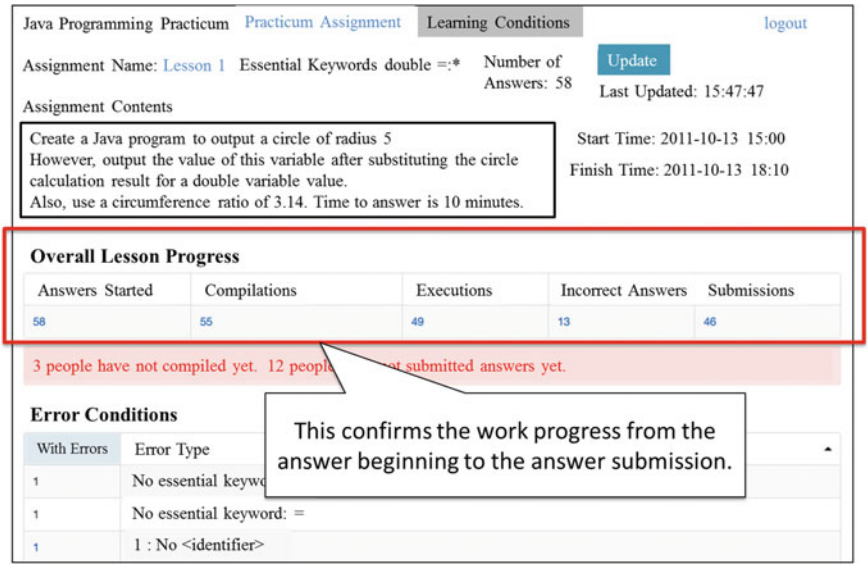
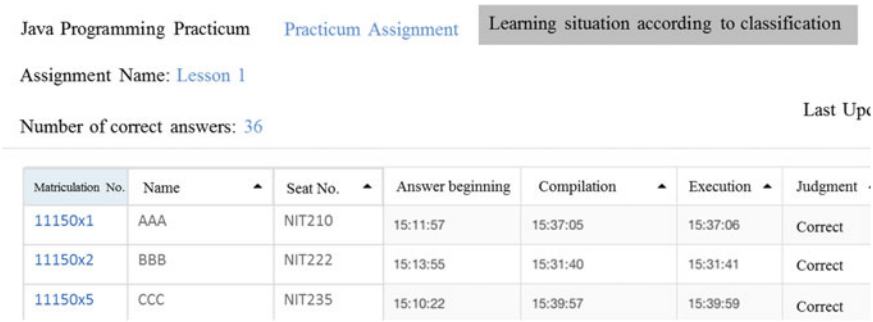


Fig. 10.2 Work progress displayed on the learning situation screen


Java Programming Practicum Practicum Assignment Learning situation according to classification

Assignment Name: Lesson 1

Number of correct answers: 36 Last Upc

Matriculation No.	Name	Seat No.	Answer beginning	Compilation	Execution	Judgment
11150x1	AAA	NIT210	15:11:57	15:37:05	15:37:06	Correct
11150x2	BBB	NIT222	15:13:55	15:31:40	15:31:41	Correct
11150x5	CCC	NIT235	15:10:22	15:39:57	15:39:59	Correct

Fig. 10.3 Learning situation screen according to classification

The error classification method uses the difference of the text by diff [22] in UNIX. The line can correspond. "Diff-b answer program example answer program" of the command in the diff is space and a tab by one and counted optional-b. This method can be classified as an error to which the error of a different line is common in two or more programs. Moreover, the correspondence line number is "Line of error line c example answer program of the answer program" in the result of diff. Therefore, the proposal function can identify the error in the line number and types of classes regardless of the way the answer program is written. This error classification can be done according to the line number and the error type in the error message. Therefore, this can also be applied to C language, C ++ language, etc.

Algorithm

1. When the instructor accesses the function, it initiates the processing.
2. Input of answer program, error message, and example answer program.
3. The first error line number and the error kind are extracted from the error message.
4. The correspondence line number of the example answer program corresponding to the error line is extracted.
5. If the correspondence line number does not exist, the correspondence line number is assumed to be unclear.
6. The correspondence line number and the error type of pair are output.

An example of executing the error classification follows. Figures 10.4, 10.5, and 10.6 show examples of the answer program, the error message, and the example answer program, respectively.

1. The instructor accesses the error classification sum function.
2. Input of answer program, error message, and example answer program.
3. The "11" of the error line number and error kind of "';' expected" are extracted from the error message.
4. Correspondence line number "7" is extracted from the answer program and the example answer program.

Fig. 10.4 Answer program

```
1 class ForExample {
2  public static void main (String[] args) {
3   int sum = 0;
4   int i;
5
6    for (i = 1; i <= 10; i++) {
8     sum += i;
9   }
10
11  System.out.println("The total from 1 to 10 is " + sum)
12  }
13 }
```

Fig. 10.5 Error message

```
ForExample.java:11:error: ';' expected
  System.out.println("The total from 1 to 10 is " + sum)
                                                      ^
1 error
```

Fig. 10.6 Example answer program

```
1 class ForExample {
2  public static void main (String[] args) {
3   int sum = 0;
4   for (int i = 1; i <= 10; i++) {
5    sum += i;
6   }
7   System.out.println("The total from 1 to 10 is " + sum);
8  }
9 }
```

5. Because the correspondence line number exists, nothing is done.
6. Correspondence line number and error kind of "7 ';' expected" are output.

The line numbers of Figs. 10.4 and 10.6 provide additional explanation. This technique does not target an irrelevant character string in the error line. Therefore, this can display the tendency for compile errors.

User Interface

The learning situation screen (shown in Fig. 10.7): The correspondence line number and the kind of error pair are displayed in the order of the sum number. When the number of presented items is clicked, it shifts to the learning situation screen according to the classification. Refer to Fig. 10.3.

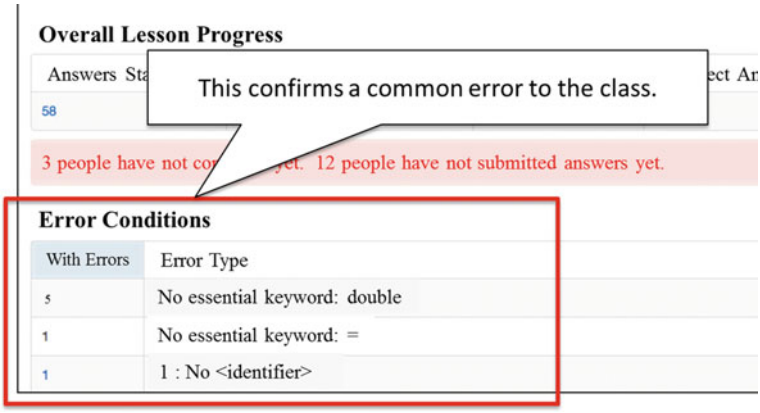


Fig. 10.7 Error classification sum on learning situation screen

10.5 Function to Assess Slow or Late Students

The method of achieving the function designs the function of the work delay detecting function. Afterward, the algorithm is defined and mounted.

Work Delay Detecting Function

Function design

The objective of the work delay detecting function is to detect the students who are behind. The pattern of the work delay is prepared during previous work [8]. Therefore, the present study detects the student who submitted their work late without requiring the pattern of the work delay. The proposal function detects students who have not finished working when the time expires. The function performed statistically labels these as outliers for the distribution of the class at the work completion time. The reason at a time now is that there is no work time data for the students who have not worked yet. Therefore, the student for whom work is late is shown at the time of the outlier. The detection by the outlier function also used the threshold method. This method is problematic, though, as the delay of continuous work is detected without fail according to the work time. The instructor should instruct only the students who are working slowly [23].

Analysis methods of the outlier use the Smirnov-Grubbs test to consider the delay of work to be an outlier by the elapsed time of work [24]. The Smirnov-Grubbs test is a technique for giving official approval of the maximum or minimum value. Figure 10.8 shows an actual class distribution of the answer beginning, compilation, execution, and submission during the laboratory class. The present study assumed a normal distribution of added time at the completion time of work, as shown in Fig. 10.8. The outlier at a time now can define delayed students who have not yet begun working. This detection can present students who are behind in their work progress because it changes the amount of detection due to an increase in the problem presenter.

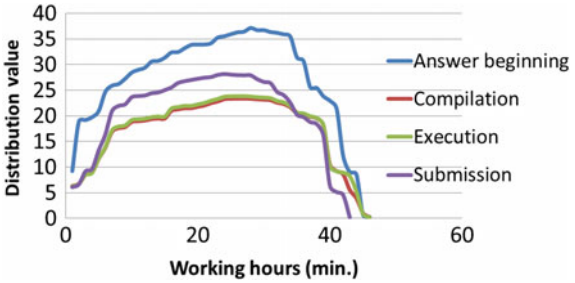


Fig. 10.8 Distribution of work time

Algorithm

1. When the instructor accesses the function, it initiates the processing.
2. Input of work time set and significance level and time now.
3. The time now is added to the work time set and the outlier is analyzed.
4. The coming off standard value is calculated from the number of the Smirnov dismissal authorization table and work time set.
5. The test statistic of each work time is calculated.
6. The maximum value of the test statistic is extracted.
7. The maximum value of the test statistic comes off and, if it is larger than the standard value, the work time is assumed to be an outlier.
8. This algorithm is ended if the outlier is not time now.
9. Information on students that the work time does not exist in the work time set is output.

An example of executing the work delay detection is as follows. The submission time set is shown in Table 10.1 as an example of input data.

1. The instructor accesses the work delay detecting function.
2. It is assumed "17:38:25" of time now and the work time set. The submission time set and the significance level "5%" are input.
3. The time now is added to the work time set.

Table 10.1 Submission time set

Students	Submission time
A	17:09:00
B	17:12:00
C	17:12:40
D	17:15:10
E	17:16:13
F	17:21:40
G	17:25:10
H	–
I	–
J	–

Table 10.2 Test statistic of work time set

Students	Test statistics
A	1.029
B	0.714
C	0.643
D	0.381
E	0.270
F	0.303
G	0.671
Time now	2.064

4. The coming off standard value is from a Smirnov dismissal authorization table to "2.032" when the number of work time sets is 8.
5. The test statistic of each work time is requested. Table 10.2 shows the test statistic of the work time set.
6. The maximum value of the test statistic is "2.064" at the time now.
7. The maximum value of the test statistic comes off and the time now is assumed to be an outlier because it is larger than the standard value.
8. The outlier is next at time now.
9. The matriculation number and the name of student "H, I, J," who does not have the work time, are output.

Tables 10.1 and 10.2 arrange the time in ascending order for explanation. Students who have not submitted their work receive a "-". Moreover, expression (10.1) shows the test statistic using the Smirnov-Grubbs test. t of the coming off the standard value of Smirnov-Grubbs test is the number n of specimens, significance level α, and $\alpha/n \times 100$ of t distribution of the degree of freedom n-2.

$$\tau = \frac{(n-1)t}{\sqrt{(n(n-2)+nt^2)}} \tag{10.1}$$

User interface

Learning situation screen (shown in Fig. 10.9): This shows the students who are late completing their work, their working names, and how many there are. When the number of people is clicked, it shifts to the learning situation screen according to the classification. Refer to Fig. 10.3.

10.6 Evaluation of the Smart Classroom by Instructors

10.6.1 Objective

The objective of the experiment was to evaluate the utility and effectiveness, in an actual lesson, of the achieved learning situation assessment function (hereafter, the

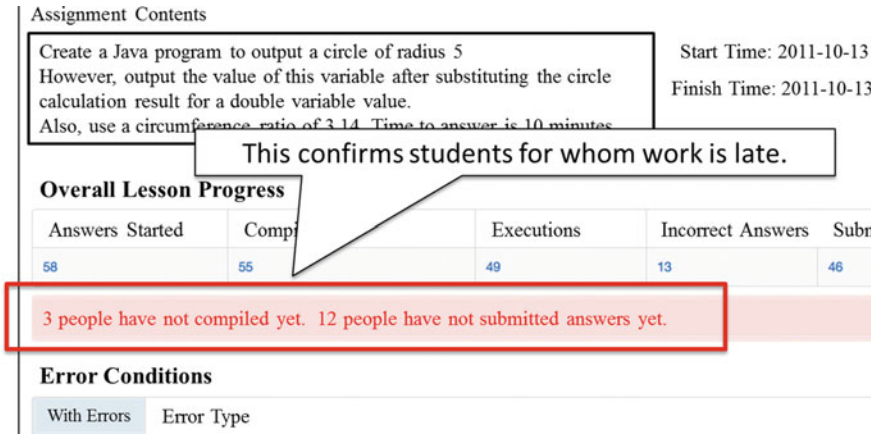


Fig. 10.9 Display of students whose work is late on the learning situation screen

achievement function). The method of evaluating the utility measures was to use the achievement function in an actual lesson. The method of evaluating the effectiveness measures the presence of guidance by the achievement function. Moreover, we questioned the lesson instructor.

10.6.2 Method

Outline of the lesson. The course used was "Basic Programming and Exercises" based on the console application in Java. Table 10.3 shows the outline of the experiment subjects. This subject has two classes, each of one instructor, and TA of two people and four people. The classes have 38 people and 73 people respectively.

Experiment system. Figure 10.10 shows the composition of the experiment system. This experiment system is a client-server method of the Web-base that can,

Table 10.3 Outline of subjects in the experiment

Learning contents	Number of exercises	Performance target
Arithmetic operator and expression	3	Do the learning of something as the expression
Condition branching	4	Do the learning of the method of switching processing on the condition
Boolean	4	Do the learning of expressible of the combination of two or more conditions by the use of Boolean
Repetition	4	When the loop construct is used, do the learning of can the description of the repetition instruction of the same processing

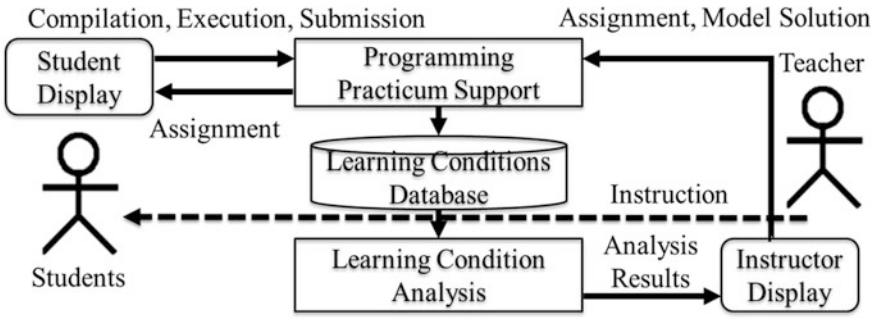


Fig. 10.10 Composition of the experiment system

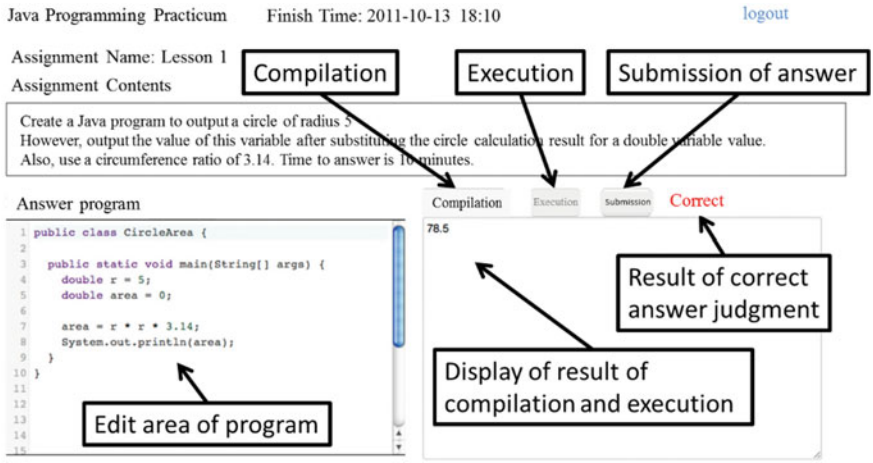


Fig. 10.11 Student screen

in real time, collect students' learning history data in the programming exercises. The student screen of the client requests preservation, compilation, execution, temporary submission, etc. from the server for the input answer program. Figure 10.11 shows the student screen. The instructor screen shows the exercise and the example answer program. These are transmitted to the server and the result is received. The server does the compilation and execution of the received program and returns the result to the client afterward. The programming languages used in the experiment were PHP, HTML, Ajax, JavaScript, and MySQL.

Experimental conditions. Two experiments were conducted. In the first, the achievement function was used twice. In the second, the achievement function was not used twice. This looks at the user's experience through the achievement function and presence of the effect. The environment of the experiment is composed

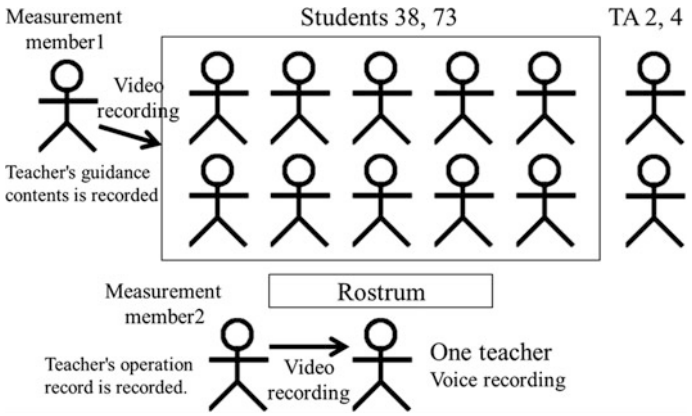


Fig. 10.12 Measurement environment of two classes

of the learning management system that can be accessed from all PCs on the network.

The significance level in the work delay detecting function was fixed at 25%. This setting was an average value used to detect outliers. Not using the work delay detecting function to evaluate the examination.

Measurement method. Measurement items for usefulness shall be the guidance frequency that is the cause of guidance on the presentation screen of realized functions in order to measure whether or not the realization function is used for instructors. Moreover, the items that measured effectiveness were the guidance frequency for each realized function based on the guidance content provided. The method of measuring these included a voice recording of the instructor, taking a picture of the lesson scenery, and taking a picture of the operation record on the instructor screen. Figure 10.12 shows the measurement environment. The learning situation screen, instructors and TAs walking around, and students raising their hands all lead to the assessment that a student requires assistance. The result of guidance was a pair of the entire guidance or an individual counseling and specific guidance content. We recorded the guidance that the instructor directed the TA to provide, but we did not record the guidance given by the TA to the student. Guidance from the TA is described in Chap. 7 of the following description. Moreover, we did not record the contents of student questions following the instructions. Finally, we interviewed the instructors at the end of the experiment.

10.6.3 Results

There were 30 instances of guidance based on the screen of the achievement function. Table 10.4 shows the guidance contents shown on the screen of the

Table 10.4 Outline of the experiment

Result	Classification	Class	Individual
	The main guidance contents	The demand of the support is pressed from the entire work progress to late students[a] 1 (5) Use the Boolean[a] 2 (2) Confirm the method of connecting character strings[a] 2 (2)	Student with a long interval time of the compilation is urged[a] 3 (2)
Guidance frequency		10	20
Total		30	

where: [a] 1, 2, and 3 show guidance given unconventionally
() the frequency of common guidance contents is shown

Table 10.5 Guidance frequency by instructor rounds and students raising hands

Achievement function		None	Used
Case	**Rounds**	42	23
	Raising hands	22	10
Total		64	33

achievement function. Guidance was given five times regarding the work progress of the class based on the work progress sum function ([a]1 in Table 10.4). Guidance was given 4 times to correct common errors of the class using the error classification sum function ([a]2 in Table 10.4). Guidance was given 2 times to slow students using the work delay detecting function ([a]3 in Table 10.4).

The numerical value in () in Table 10.4 is the frequency of common guidance contents. Moreover, guidance was given 64 times by the instructors and TAs walking around and from the students raising their hands. The achievement function was not used in these cases. The use case was 33 times. Table 10.5 shows the guidance frequency by the instructors and TAs walking around and students raising their hands.

The results of the interview with the instructor are seen below. The first question was "Please tell us a good point and a bad point about the proposal function." Their answers to this question were as follows:

Answer of Instructor A:

Good points:

- Everything from the beginning of the exercises to submission is automated as a system. Therefore, the instructional workload is decreased during the exercise and the confirmation of the problem submission.
- The error status when compiling can be understood. Therefore, I can understand the students' standstill situations in detail. This can be used as prior information before guidance is provided.

Bad points:

- Nothing in particular.

Answers of Instructor B:
Good points:

- It is possible to provide comments about compile errors to many students at once.
- The learning situation can be assessed remotely from the rostrum. Therefore, I can issue instructions to the TA based on the contents.
- Moreover, I can understand the causes for which student work is late based on the information of the function.

Request:

- Please let me know (send me the alert message) about those students who need guidance.

The second question was as follows: "Please tell us about the effect of the display contents of the learning situation assessment function on guidance." The instructors' answers were as follows:

Answers of instructor A:

- Information on student work progress can provide understanding of the learning situation and needs of the class.
- Information on error classification sum can be provided for the entire class or individual students for guidance.

Answer of instructor B:

- Information on students for whom work is late can identify students who are slower than others and need guidance.

10.6.4 Consideration of the Results

Guidance based on the proposal function was given 30 out of 63 times. Additionally, the instructor interviews provided useful feedback regarding the contents of the achievement function. The achievement function presents useful information regarding the areas in which students require guidance. This is effective for the assessment of student learning situations during programming exercises.

The guidance given when only the achievement function was used is detailed in [a]1, [a]2, and [a]3 in Table 10.4. Additionally, the interview results provided answers that confirmed that the contents led to guidance. Moreover, the ratio of guidance from the instructor was high in the lessons that used the achievement function. Table 10.6 shows the ratio of guidance from the instructor. The numerical value of

Table 10.6 Ratio of active guidance by the instructor

Achievement function		None		Used
Case	**Observing screen and making rounds**	The instructor's active guidance	65.6% (42)	84.1% (53)
	Raising hand	The student requests guidance	34.4% (22)	15.9% (10)
Total			64	33

where the guidance frequency to the cause of guidance is shown in ()

() in Table 10.6 is the frequency of guidance for each cause of guidance. The ratio of active guidance increased because the instructor was able to understand the learning situation from the screen display of the achievement function. The ratio of guidance given at the students' request decreased because the instructor was able to identify the question from the display screen before the question came from the students. The ratio of this guidance is intentionally high at a significance level of 1% by the chi-square test. The instructor was able to assess and instruct a difficult learning situation. Moreover, the instructor can give guidance before help is asked for by the student. Therefore, the achievement function is effective for guidance during programming exercises.

According to the results of the interview, the instructor should return to the rostrum during the achievement function. Instructors can not check their own computer while teaching at student's desk. Instructors can direct and instruct more students without returning to the rostrum if they use the accomplishment function. A method to overcome this problem is the use of the tablet terminal. Figure 10.13 shows instruction using a tablet terminal in the lesson in the following year of the assessment experiment.

Students can request instructor guidance remotely through the use of tablets. Therefore, tablet use is more effective than the instructor making rounds when time efficiency must be taken into consideration.

Fig. 10.13 Assessment of learning situation using a tablet terminal

When the achievement function is developed with other languages and courses, it raises the issue of what should be how much time can be given to each learning task. Moreover, the error classification sum function can correspond in the case of the console application. There is a problem, though, in the method of acquiring the error for the GUI application.

10.7 Function to Understand Programming Behavior for TAs

10.7.1 Analysis for TA

Data mining, as a smart educational technology, can support the TA and help in the realization of a smart learning environment.

During programming exercises, the main role of TAs is to assist students in correcting their errors. The programming behavior of a particular student, which includes his or her programming style, is different from other students' behaviors. For this reason, it is hard for TAs to provide appropriate guidance other than error handling for each student [25]. The lecturers understand students' learning contexts due to their prior teaching experiences. TAs have greater difficulty understanding students' learning contexts because they do not have enough teaching experience to do so. Therefore, the present study proposes a function to help TAs understand student programming behavior and common student difficulties. The factors of problem solving in programming include how much a particular student follows the programming codes, understands the grammar of the programming language, and uses the compiler. This section reports on the results of our data mining, which focused on the programming mode. The programming mode is the basic attitude that represents how much a particular student follows the programming codes that they are encouraged to follow at our institute. Furthermore, by focusing on the programming mode, we can collect students' behavioral data in real-time. This is done through the programming exercise support system that we have developed [19].

The present study classifies the features of the students' programming behaviors in order to infer the characteristics of each student. Through the classification, we have examined the relationship between the results of the questionnaire toward the programming mode and the programming behaviors. The behaviors include the number of compilations, the number of trial executions, the number of errors, the number of repetitions of the same errors, the average interval of the compilations, and the average intervals of the executions. The programming codes are shown in Table 10.7. We found that each student had a particular programming trait. We also found that we can measure that trait by observing how much a particular student follows the programming codes.

Table 10.7 The programming codes

Code #	Programming code details
1	When the grammar is ambiguous, examine it in the texts or manuals
2	Add one line of sentence and compile
3	When compile errors appear, deal with the first error
4	Construct programs from the skeleton
5	When the execution result is not correct, trace the execution process
6	Insert spaces after keywords so that they are highlighted
7	Write output sentences first so the behaviors of the program can be observed
8	Make and try several solutions to solve the errors
9	Insert spaces after commas, so that they are easily seen
10	When inserting an opening parenthesis, insert the corresponding closing parenthesis immediately
11	Write meaningful comments so that the semantics of the program can be understood
12	Choose meaningful variable names
13	When the usage of the instruction is ambiguous, refer to the samples
14	Insert space lines so that blocks in the program are clearly seen
15	Indent the codes so that the structure of the program can be clearly seen
16	Insert spaces before and after operators so that they are highlighted
17	When modifying the program, leave the old source codes as comments
18	When the program behaves strangely, print out intermediate variables
19	Write many more programs until you can write them comfortably
20	Use the patterns of program codes

10.7.2 Classification of Programming Behaviors

In this section, we classify the features of programming behaviors. In order to do so, we have performed a cluster analysis of the records of the programming behaviors. Clustering is a process of grouping objects into classes of similar objects [26]. It is an unsupervised classification or partitioning of patterns (observations, data items, or feature vectors) into groups or subsets/clusters based on their locality and connectivity within an n-dimensional space. In the present study, we have performed a cluster analysis over the submitted programs that solve the assignments. The total number of the subjects was 80 and we employed seven feature variables. Figures 10.14 and 10.15 show the results of the cluster analysis.

We employed Ward's method of hierarchical clustering in the cluster analysis [27]. We also employed k-means for the non-hierarchical clustering. Ward's method is a criterion applied in hierarchical cluster analysis [28]. K-means is an algorithm that clusters objects based on attributes in k partitions [29]. The result of the k-means analysis depends on the initial values. Therefore, we have chosen initial values as the best values produced by Pseudo-F, where the number of clusters is 4, in our preliminary experiments. Pseudo-F is an evaluation criterion in cluster

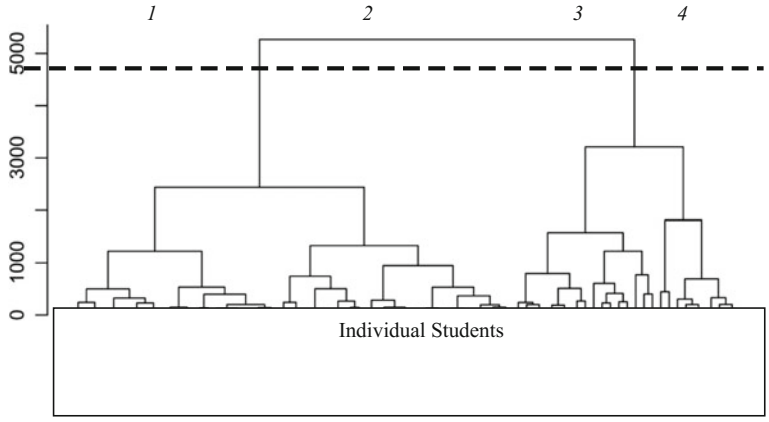


Fig. 10.14 Cluster analysis using Ward's method

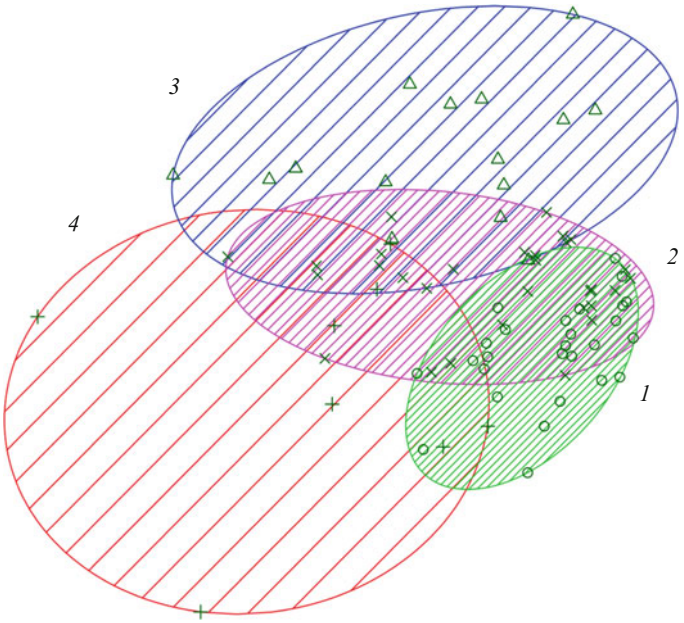


Fig. 10.15 Cluster analysis using k-means

analysis [30]. A good Pseudo-F value means the resultant clusters have little overlapping. In addition, the density of each cluster is high.

The present study examined the relationship between the students' programming behaviors and their programming modes. Tables 10.8 and 10.9 show the results. The numerical values in Tables 10.8 and 10.9 are mean values of the number of

Table 10.8 Breakdown of dotted line upper floor layer in Fig. 10.1

Cluster	Solvingtime	Compilation interval	Execution interval	Compilation frequency	Execution frequency	Number of errors	Number of same errors[a]	Codes of programming	Number of students
1	528	209	267	7	5	1	1	10	25
2	812	379	606	6	3	3	3	6	28
3	1405	590	1160	7	3	4	3	5	17
4	1506	228	337	18	13	5	4	4	10

Table 10.9 Cluster breakdown of Fig. 10.2

Cluster	Solving time	Compilation interval	Execution interval	Compilation frequency	Execution frequency	Number of errors	Number of same errors[a]	Codes of programming	Number of students
1	514	239	289	6	5	1	1	9	27
2	884	364	589	7	4	3	3	8	30
3	1398	599	1216	7	3	4	3	7	15
4	1694	225	378	20	14	6	5	5	8

where [a] the same person making the same error multiple times

individuals in each cluster. The numerical values of the programming modes are the answers of four-stage evaluation (+1 done and −1 not done). Moreover, we have performed the correlation analysis with the duration time for problem-solving and programming mode. We have observed positive correlations (0.24) in 18 of the programming codes.

The dotted line of Fig. 10.14 indicates the middle of the dendrogram, that is, 2500. We can observe that Tables 10.8 and 10.9 are similar. We can conclude that there were four clusters, as follows:

- **Cluster 1**: Duration time of problem-solving is short. The score of the programming mode is high. The intervals of the compilation and the intervals of execution are short. The compilation frequency is few. The students understand the contents of the errors and what they are doing in the program.
- **Cluster 2**: Duration time of problem-solving is shorter than cluster 3. The score of the programming mode is low. A lot of errors exist and many are similar errors. The intervals of the compilation and the intervals of execution are shorter than those of the cluster 3. The students are doing the programming without understanding the contents of the errors.
- **Cluster 3**: Duration time of problem-solving is long. The score of the programming mode is low. The students are repeating the same error. The students are compiling without understanding the contents of the errors.
- **Cluster 4**: Duration time of problem-solving is long. The score of the programming mode is low. In Table 10.8, there are many compilation frequencies and execution frequencies. The students compile frequently and are committing many errors. In Table 10.9, the compilation frequency and the execution frequency are low. Surprisingly, these students are submitting the correct solutions to the problems. They have likely copied the correct answers from cluster 1 students.

10.8 Evaluation of the Smart Classroom by TAs

We set up a hypothesis based on the considerations of the previous section. The hypothesis is that the programming behavior appears in the programming mode.

Experiments. In this section, we verify the hypothesis about the programming mode and programming behavior. This was to what degree the understanding level in the programming codes corresponds to the programming behavior. Moreover, the present study verifies the differences in the results of the questionnaires about the subjective understanding level of the students before and after the experiment. The TA instructs the students based on the results of the questionnaires completed before the experiment. The TA guides the basic attitudes within the low numerical value for understanding level based on the questionnaires.

The verification method is to evaluate the duration time of the problem solving of both the experimental group and the control group. The group consists of forty

students who all submitted their solutions late. These students were split into two groups at random to create the control group and the experimental group. The procedures of the experiment were as follows:

1. Perform understanding level inquiry of the programming mode the first time. Acquire the duration time for problem solving before the experiments.
2. First experiment: guidance from TA.
3. Second experiment: guidance from TA.
4. Third experiment: guidance from TA.
5. Fourth experiment: no guidance from TA. Perform understanding level inquiry, the second time, of the programming mode. Acquire the duration time for problem solving after the experiments.

Figure 10.16 shows the change in the problem solving time. We can observe the improvement of the duration time of the problem solving in the experimental group; however, there was no significant difference between the two groups.

Table 10.10 shows the result of the understanding level inquiry. The understanding frequency value of the experimental group rose by 5% compared with the control group. The result of the questionnaire, according to the cluster, is shown in Table 10.11. In Table 10.11, the gray background shows where the improvements in student understanding levels are remarkable.

Discussion. The problem solving time may have been shortened due to the understanding level of the programming codes corresponding to the programming behavior. Data mining of students' behaviors enabled the TAs to provide effective guidance. Moreover, the results of the data mining provided the TAs with the features of the students' programming behaviors.

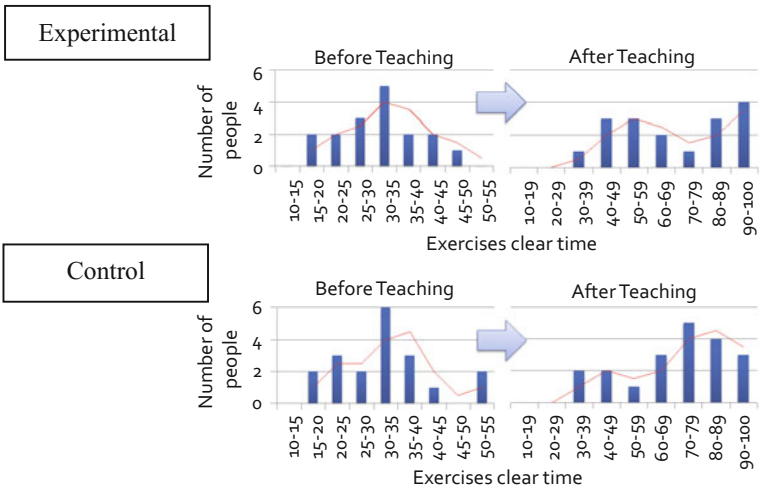


Fig. 10.16 Changes in problem-solving time

Table 10.10 Results of the questionnaire of understanding level by student subjectivity

Group	Understanding frequency value			
	Before	After	Effective	None
Experimental	6.66	9.7446% UP	13 students	4 students
Control	5.84	8.241% UP	13 students	6 students

Table 10.11 Results of the questionnaire in each cluster

Cluster	Matter	1	2	3	4	5	6	7	8	9	10	11
1	Before	0.1	0.8	0.7	0.0	0.3	0.1	0.6	0.4	−0.2	0.1	0.7
	After	−0.2	1.0	0.9	0.1	0.4	0.1	0.5	0.6	0.0	0.1	0.9
2	Before	−0.1	0.3	0.7	0.2	0.3	0.3	0.8	0.4	−0.4	0.6	0.7
	After	0.2	0.7	0.7	0.2	0.3	0.7	0.9	0.9	0.1	0.6	0.9
3	Before	0.1	0.2	0.8	−0.2	0.5	0.0	0.7	0.5	0.0	0.3	0.7
	After	−0.2	0.6	0.9	0.1	0.3	0.6	0.8	0.8	0.1	0.5	0.7
4	Before	−0.3	0.3	0.7	−0.1	−0.2	0.5	0.6	0.4	−0.3	0.6	0.9
	After	0.1	0.6	0.6	−0.3	0.1	0.6	0.8	0.3	−0.1	0.6	0.7
Cluster	Matter	12	13	14	15	16	17	18	19	20	Students	
1	Before	0.8	0.8	0.8	0.7	0.8	−0.3	0.5	−0.4	1.0	5	
	After	0.5	0.9	1.0	0.3	0.8	−0.1	0.5	0.4	0.8		
2	Before	0.1	1.0	0.9	0.3	0.6	−0.2	0.3	−0.3	0.8	13	
	After	0.2	0.7	0.8	0.3	0.7	0.0	0.5	−0.2	0.9		
3	Before	0.3	0.3	0.8	−0.2	0.3	−0.4	0.1	−0.4	0.7	11	
	After	0.3	0.6	0.8	0.2	0.6	0.0	0.4	−0.4	0.9		
4	Before	0.0	0.8	0.7	0.4	0.9	−0.5	−0.2	−0.6	1.0	7	
	After	0.3	0.5	0.9	0.5	0.5	0.2	−0.2	−0.3	1.0		

We possibly did not observe significant difference between the two groups because of the close connections between friends. For example, students may have shared the advice of the TA with their friends.

The guidance effects on programming mode are shown in Table 10.12. The data mining of students' behaviors enables TAs to give advice beyond just simple error correction.

Table 10.12 Effects of programming mode

Cluster	Effective	Ineffective
1	Examine the grammar Write the comment Make a lot of programs	None
2	Adjust the appearance Write the comment	Make a lot of programs
3	Adjust the appearance	Compile for each line
4	Compile for each line Deal with the first error Copy the sentences that work	Examine the grammar Write the output sentence first

10.9 Scaling-up to Smart University

We have experimented on subjects in programming exercises. We would like to apply it fully to other subjects.

We are currently trying the proposed method in language classes [31]. We are studying how the students can reflect even in the face-to-face class. Typically, we have implemented Web exercises using Google Forms for continual self-reflection. We have performed the text mining to "Devised it" of the Web exercises. We have observed the transformation from the mentally passive word "Do" to the active words "Examine it." We found that Google Forms motivates students' self-regulatory learning. Based on these findings, we also present a prospect for a lesson that draws out the subjectivity of the students.

Furthermore, we are studying which tasks effectively motivate students. Currently, it is not very smart, but further study of deep learning in programming classes will make it possible to extract issues that increase motivation. The smartness level can be improved by developing these areas.

There is manual work in realizing smartness and it would be a restriction when considering large-scale deployment. By using more sophisticated machine learning, we can realize a smart learning environment. Moreover, we have analyzed the relation between programming behavior and programming mode. The results were related to the behavioral features and the programming mode. The authors have reached the hypothesis that the duration time of problem solving could be reduced as a result of the effective guidance of the TAs concerning programming mode. Then, the authors performed the assessment experiments of guidance by TAs. As a result, the duration time of the problem solving of students who were taught has been shortened. Therefore, the proposed technique enables TAs to provide effective support for students because they can better obtain a deep understanding of the learning situation of each student. Therefore, we can conclude that the proposed function enables TAs to effectively support students when learning and programming.

10.10 Conclusions

This chapter described the achievement of the learning situation assessment function in the programming exercises for entry-level programming classes. This function provides an assessment of the learning situation of individual students as well as the class as a whole. The function for students indicates who is behind in their work and who needs help. The function for the class displays the sum of the work progress from the beginning of the exercise until submission. Moreover, the function provides the error classification summary by identifying compile error lines based on the example answer program. We found that the function applied to the Java programming exercises actually produced accurate output. Moreover, the

function identifies students that are behind in their work via outlier analysis of class work progress. Because the instructor appropriately sets the standard value of the outlier, this function can identify students who are behind in order to provide them with assistance before it is too late. In the assessment experiment in an actual lesson, we observed appropriate guidance being presented using information from the proposed function. We, thus, can conclude that the proposed function is effective for assessing student learning situations in programming exercises for beginners.

In summary, we have achieved a smart educational environment through the use of this function. The environment supports instructors and TAs so that they can provide smart pedagogy to students.

Acknowledgements This work was supported by Japan Society for Promotion of Science (JSPS), with the basic research program (C) (No. 15K01094 and 26240008), Grant-in-Aid for Scientific Research.

References

1. Macfadyen, L.P., Dawson, S.: Mining LMS data to develop an "early warning system" for educators: a proof of concept. Comput. Educ. **54**(2), 588–599 (2010)
2. Open University of Japan: H21-22 Survey on the Promotion of ICT Use Education. Center of ICT and Distance Education (2011)
3. Klosgen, W., Zytkow, J.: Handbook of data mining and knowledge discovery. Oxford University Press, New York (2002)
4. Romero, C., Ventura, S., Garcia, E.: Data mining in course management systems: moodle case study and tutorial. Comput. Educ. **51**, 368–384 (2007)
5. Horiguchi, S., Igaki, H., Inoue, A., et al.: Progress management metrics for programming education of HTML-based learning material. J. Inf. Process. Soc. Jpn. **53**(1), 61–71 (2012). (In Japanese)
6. McCartney, R., Eckerdal, A., Mostrom, J.E., Sanders, K., Zander, C.: Successful students' strategies for getting unstuck. SIGCSE Bull. **39**(3), 156–160 (2007)
7. Sagisaka, T., Watanabe, S.: Investigations of beginners in programming course based on learning strategies and gradual level test, and development of support-rules. J. Japan. Soc. Inf. Syst. Educ. **26**(1), 5–15 (2009). (In Japanese)
8. Igaki, H., Saito, S., Inoue, A., et al.: Programming process visualization for supporting students in programming exercise. J. Inf. Process. Soc. Jpn. **54**, 1 (2013). (In Japanese)
9. Kurasawa, K., Suzuki, K., Iijima, M., Yokoyama, S., Miyadera, K.: Development of learning situation understanding support system for class instruction in programming exercises. Inst. Electron. Inf. Commun. Eng. Technol. Rep. ET Educ. Eng. **104**(703), 19–24 (2005). (In Japanese)
10. Colthorpe, K., Zimbardi, K., Ainscough, L., Anderson, S.: Know thy student! Combining learning analytics and critical reflections to develop a targeted intervention for promoting self-regulated learning. J. Learn. Analytics **2**(1), 134–155 (2015)
11. Ryan, C., Michael, C.L.: Debugging: from novice to expert. ACM SIGCSE Bull. **36**(1), 17–21 (2004)
12. Alex, G., Johan, J., Bastiaan, H.: An interactive functional programming tutor. In: Proceedings of the 17th ITiCSE 2012, pp. 250–255. ACM (2012)

13. Serral, E., De Weerdt, J., Sedrakyan, G., Snoeck, M.: Automating immediate and personalized feedback taking conceptual modelling education to a next level, In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), pp. 1–6. IEEE (2016)
14. Truong, N., Roe, P., Bancroft, P.: Automated feedback for fill in the gap programming exercises, In Proceedings of the 7th Australasian conference on Computing education, vol. 42, pp. 117–126. Australian Computer Society, Inc. (2005)
15. Uskov, V.L., Bakken, J.P., Pandey, A., Singh, U., Yalamanchili, M., Penumatsa, A.: Smart university taxonomy: features, components, systems. In: 2016 Smart Education and e-Learning, pp. 3–14. Springer, Cham (2016)
16. Hwang, G.J.: Definition, framework and research issues of smart learning environments–a context-aware ubiquitous learning perspective. Smart Learn. Environ. a Springer Open Journal, 1:4, Springer (2014)
17. Friend, M., Bursuck, W.: Including students with special needs: a practical guide for classroom instructors, Chap. 5, pp. 164–165. Prentice Hall, Saddle River (2006)
18. Watanabe, H., Arai, N., Takei, S.: Case-based evaluation support system of novice programs written in assembly language. J. Inf. Process. Soc. Jpn. **42**(1), 99–109 (2001). (In Japanese)
19. Kato, T., Ishikawa, T.: Design and evaluation of support functions of course management systems for assessing learning conditions in programming practicums. Int. Conf. Adv. Learn. Technol. **2012**, 205–207 (2012)
20. Tanaka, K.: Yokuwakaru Jyugyouron. Minervashobo (2007) (In Japanese)
21. Japan Society for Educational Technology: Kyouiku Kougaku Jiten. Jikkyo Shuppan (2000) (In Japanese)
22. Diffutls. http://www.gnu.org/software/diffutils/diffutils.html. Accessed 14 Sept 2016
23. Ueno, M.: Online outlier detection for e-learning time data. J. Inst. Electron. Inf. Commun. Eng. J90-D **1**, 40–51 (2007). (In Japanese)
24. Bull, C.R., Bull, R.M., Rastin, B.C.: On the Sensitivity of the chi-square test and its con-sequences. Meas. Sci. Technol. **3**, 789–795 (1992)
25. Yasuda, K., Inoue, A., Ichimura, S.: Programming education system that can share problem-solving processes between students and teaching assistants. J. Inf. Process. Soc. Japan **53**(1), 81–89 (2012). (In Japanese)
26. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
27. Kamishima, T.: A survey of recent clustering methods for data mining (Part 1): try clustering! J. Japan. Soc. Artif. Intell. **18**(1), 59–65 (2003). (In Japanese)
28. Michael, R.A.: Cluster Analysis for Applications. Academic Press, New York (1973)
29. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, California, USA, vol. 1, pp. 281–297 (1967)
30. Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. **3**, 1–27 (1974)
31. Kato, T., Kambayashi, Y., Kodama, Y.: Practice for self-regulatory learning using google forms: report and perspectives. Inf. Eng. Express Int. Inst. Appl. Inf. 2016 **2**(4), 11–20 (2016)