Categories of Attitude and Student Determined by Cluster Analysis of the Attitudes toward Programming Abilities in a Blended Class

Isao Miyaji[1], Kouji Yoshida[2]
Okayama University of Science[1], Shounan Institute of Technology[2]

Abstract

In a programming course, lectures were given using a slideshow, and syntax and example programs from a textbook were explained. Afterward, students received worksheets with example programs and problems for practicing syntactic elements, and the professor explained the worksheets. The students then performed an exercise where they created a program based on example programs as an assignment. They were instructed to finish as much of their program as possible during class and to submit their program file and a report file over an e-learning site. On the seventh and fourteenth weeks, students were assigned to create programs of their own design as an independent project, then to peer review each other's programs, use the results of the review to revise, and resubmit the programs on the eighth and fifteenth weeks. They could learn either in class or through lecture slides uploaded to an e-learning site. Students' attitudes and their familiarity with terminology were assessed before and after the course. This educational information was analyzed with significance tests and cluster analysis, the results of which are reported in this paper. Attitude and students were classified in five and four clusters respectively by cluster analysis for the attitude about the ability related to the programming. The characteristic of each cluster is considered and discussed by comparing average rating values.

1. Introduction

Blended learning is currently being used to make classes more effective, more efficient, and more attractive to students, particularly at institutions of higher education [1][2]. The author of this paper promotes a university education that includes creating things and evaluating them in order to build problem-solving skills [3]. It is advocated that in addition to lectures, learning opportunities for a variety of students should be created through classes that take individual students' situations into account and allow them to prepare for class and review "anytime and anywhere."

One way of doing this is blended classes that combine methods such as lecture organizing notebooks, e-learning (learning with lecture slides, learning with exercise problems, collaborative learning and peer review of student-generated learning materials), and quizzes, which have been demonstrated effective in a previous report by the same author of conducting such a course [4] [5]. The author also found that using comprehension surveys and increasing interactions between students and faculty can further enhance results [6].

Several methods to deepen students' understanding in programming class have been proposed [7]. One method that has been reported to be effective is blended learning classes [8]. There are also reports of students collaborating on projects and then evaluating them [9].

In this study, a professor conducted blended classes that utilized e-learning while considering what media are required for a programming class. The format of the class was as follows. Problems and answers from the previous class were explained, and then a lecture was given with slides based on the day's syntax elements and processing details. Next, students were given a worksheet with example problems and assigned problems that included the information taught that day, and the professor explained the worksheet with slides. Afterward, they performed an exercise where they created an assigned program while referring to syntax, processing details, and example programs. There was also a collaborative learning element to this. At the midpoint and end of the course, students created a program as an independent project, peer reviewed each other's programs, and revised their program based on the review. Students were surveyed about their thoughts regarding this system and results were reported. Students' scope of knowledge has been measured using a metric such as familiarity with terminology. This paper reports the results of conducting and analyzing pre- and post-course surveys of students' attitudes and their familiarity with terminology. Attitude and students were

classified in five and four clusters respectively by cluster analysis for the attitude about the ability related to the programming. The characteristic of each cluster is considered and discussed by comparing average rating values.

2. Course design and content

The blended course was a programming elective for second-year students in the Faculty of Information Sciences at A University. Each class was 90 minutes long and 15 classes were held. The contents of the lectures and lecture plans are shown in Table 1. A final examination was held after the fifteenth class to motivate students to learn and to assess their understanding. Twenty-seven students took the course. Exercises were led by the instructor and a TA.

2.1. Course objectives and goals

In web services that run on the current Internet, programs such as CGI dynamically run on the web server and change web pages. The objectives of this course are to learn PHP, which is a language often used in CGI, as well as to learn how to execute basic programs and to be able to create a dynamic website. The achievement goals are as follows: (1) understand the relationship between a server and a client, (2) understand web services, (3) learn how to use PHP, and (4) learn how to generate CGI.

Students will also engage in researching, thinking, creating, evaluating, and revising activities during the course and will build problem-solving skills that they will need as members of society.

2.2. Class format

The format of each class was as follows. First, answers to problems from the previous class were explained (approximately 10 minutes). Next, a lecture based on the day's syntax elements and processing details from the textbook [10] was given using slides (approximately 30 minutes). Students were then given a worksheet with example problems and practice problems that included the content from that day. An explanation of this worksheet was given using slides (approximately 10 minutes). Afterward, students were instructed to perform an exercise where they created a program while referring to syntax, processing details, and example problems (approximately 40 minutes). Students were allowed to download example programs, run the programs, and observe the processing flow as well as the result of running the program. Students who finished their practice program were instructed to submit the program with a report file.

2.3. Description of assignments

As assignments, students were instructed to create one related PHP program for each chapter discussed in the lecture. After they finished their program, they were instructed to paste it into a report form outline on A4 paper and to submit it along with the program file. The items on the report form were a program list, result of execution, and observations. Grades were determined comprehensively from submitted work such as exercises and assigned problems as well as from the final examination.

On the seventh and eighth weeks as well as the fourteenth and fifteenth weeks, students were assigned to independently design and create a program for another person to use, for example, a card game, a fortune-telling program, or a math learning program using elements such as control statements and arrays. The process for completing this project was as follows. On the first week of the project, students (1) created a program, (2) ran their

Table 1. Design of the programming course

Week	Contents	Lesson							e-learning				
		No. of slides	Distributed documents	Textbook	Examples and assignments	Self-imposed assignment	Survey of term recognition	Survey of attitude	Learning by lesson slides	Downloading	Program	Reports	Evaluation sheet
1	Before beginning PHP	36	Document of lesson plan				Pre	Pre		How to create PHP program			
2	Basic program	25	How to create PHP program	Chapter 1	Example 1				Chapter 1	Reprt			
3	Variable	28		Chapter 2	Example 2				Chapter 2	Evaluation sheet	Assignment 1	Assignment 1	
4	Condition sentence	42		Chapter 3	Example 3				Chapter 3		Assignment 2	Assignment 2	
5	Repetition sentence	40		Chapter 4	Example 4				Chapter 4		Assignment 3	Assignment 3	
6	Array and control sentence	27		Chapter 2	Example 5	Specification 1			Chapter 2	pendent proj	Assignment 4	Assignment 4	Self assessment
7	Mutual use of self-imposed assignment 1, Evaluation, Correction					Program					Assignment 5	Assignment 5	Peer assessment
8	Mutual use of self-imposed assignment 1, Evaluation				Example 6	Correction							Peer assessment
9	Function	32		Chapter 5	Example 7				Chapter 5		Assignment 6	Assignment 6	
10	Use of the regular expression	27		Chapter 6	Example 8				Chapter 6		Assignment 7	Assignment 7	
11	Use of the character string function	23		Chapter 6	Example 9						Assignment 8	Assignment 8	
12	Use of the file	22		Chapter 8	Example 10				Chapter 8		Assignment 9	Assignment 9	
13	Access to a database	30		Chapter 8	Example 11	Specification 2					Assignment 10	Assignment 10	Self assessment
14	Mutual use of self-imposed assignment 1, Evaluation, Correction					Program		Independen t project			Assignment 11	Assignment 11	Peer assessment
15	Mutual use of self-imposed assignment 1, Evaluation					Correction	Post	Post					Peer assessment

created program, (3) underwent peer review, and (4) revised their program based on peer review. On the following week, they (5) ran their revised program, (6) did another peer review, (7) assessed whether they had revised the program properly, and (8) filled out a report. Report forms for submitted independent projects were uploaded so that others could view them.

2.4. Contents of e-learning

To intensify the effect of this lecture, e-learning functions were added as follows: (1) learning with lecture slides; (2) browsing the assignment; (3) downloading documents and one template; (4) submitting and uploading the exercise as report.

3. Results of analysis

Students' familiarity with terminology was assessed to understand how their knowledge changed after they took the programming course and their attitudes toward their abilities were assessed to understand how their attitudes changed. Data from these surveys were analyzed with significance tests and the results of this analysis are explained below. Attitude and students were classified in some clusters by cluster analysis for the attitude about the ability related to the programming. The characteristic of each cluster is considered and discussed by comparing average rating values.

In the following results, a significance level of 5% was considered to indicate a significant difference. The symbols m, SD, t, and p indicate the mean, the standard deviation, the test statistic, and the p-value, respectively. Significance levels of 0.1%, 1%, 5%, and 10% are indicated with ***, **, *, and +, respectively.

3.1. Results from the survey of familiarity with terminology

As shown in Table 2, pre-course (Week 1) and post-course (Week 15) surveys [6] were given to assess students' familiarity with 60 terms. These 60 terms were important terms that appeared in the textbook, were related to the content of the programming classes, and were selected from the index.

Three levels of familiarity with terminology were used: 1. "Do not know," 2. "Do not know well but have heard of it," and 3. "Know it." The mean level of familiarity with terminology was 1.7 before the course and 2.4 after the course. Twenty-five students each responded to the pre-course and the post-course surveys.

Analysis with paired t-tests revealed a significant difference between the overall pre-course and post-course levels of familiarity with the 60 terms (last line of Table 2). Students' overall level of familiarity was significantly higher after the course, which showed that students' overall knowledge of programming increased after the course.

Analysis with paired t-tests revealed a significant difference between pre-course and post-course levels

Table 2. Significance tests for familiarity with terms relating to the course

No	Technical term	Pre		Post		t-test	
		m	SD	m	SD	t	p
1	Apache	1.2	0.5	1.9	0.8	4.6	***
2	array	1.4	0.7	2.6	0.6	7.6	***
3	break	2.0	0.9	2.6	0.6	3.8	***
4	case	1.8	0.8	2.4	0.6	3.9	***
5	CGI	1.7	0.8	2.1	0.6	2.4	*
6	CHECKBOX	1.4	0.8	2.5	0.5	7.2	***
7	chop	1.2	0.4	1.8	0.7	4.5	***
8	continue	1.6	0.7	2.2	0.7	3.5	***
9	date	2.0	0.9	2.5	0.7	2.4	*
10	define	2.4	0.7	2.6	0.6	1.7	+
11	do～while	2.6	0.6	2.7	0.6	0.8	
12	else	2.8	0.5	2.8	0.6	0.0	
13	eiseif	2.1	0.9	2.7	0.6	3.3	***
14	endforeach	1.1	0.3	1.8	0.7	5.3	***
15	endif	1.2	0.5	2.0	0.8	4.7	***
16	exit	1.7	0.7	2.2	0.8	3.0	**
17	fclose	1.0	0.2	2.1	0.9	6.7	***
18	feof	1.0	0.2	1.9	0.8	5.9	***
19	fgets	1.0	0.2	1.9	0.9	5.6	***
20	file_exits	1.2	0.5	1.8	0.8	3.9	***
21	filesize	1.5	0.7	2.1	0.8	3.2	***
22	float	1.4	0.7	2.2	0.8	4.6	***
23	fopen	1.1	0.4	2.0	0.8	6.2	***
24	for	2.6	0.6	2.8	0.5	1.8	+
25	foreach	1.0	0.2	2.2	0.8	8.9	***
26	form	1.6	0.7	2.4	0.7	4.9	***
27	fputs	1.0	0.2	2.2	0.8	8.1	***
28	FTP	1.8	0.8	2.4	0.7	3.2	***
29	function	1.9	0.8	2.6	0.6	3.9	***
30	GET	1.6	0.6	2.2	0.7	3.8	***
31	global	1.5	0.6	2.1	0.7	4.0	***
32	GUI	1.5	0.7	2.1	0.7	3.6	***
33	HTML	2.6	0.6	2.8	0.4	1.8	+
34	HTTP	2.4	0.7	2.7	0.4	2.5	*
35	if	2.7	0.5	2.8	0.4	1.5	
36	include	2.3	0.8	2.5	0.7	1.1	
37	MySQL	1.3	0.7	2.1	0.8	4.7	***
38	NULL	2.0	0.8	2.6	0.6	3.5	***
39	print	2.4	0.8	2.8	0.4	2.9	**
40	require	1.1	0.3	2.1	0.7	7.9	***
41	return	2.6	0.7	2.6	0.6	0.0	
42	round	1.4	0.6	2.2	0.8	5.0	***
43	SELECT	1.8	0.8	2.5	0.6	4.2	***
44	STDIN	1.0	0.2	1.8	0.6	7.3	***
45	STDOUT	1.1	0.3	1.8	0.6	6.7	***
46	SUBMIT	1.3	0.6	2.1	0.7	5.0	***
47	switch	1.8	0.7	2.5	0.6	4.8	***
48	TEXT	1.8	0.8	2.3	0.7	2.9	**
49	URL	2.7	0.7	2.8	0.5	0.9	
50	VALUE	1.8	0.8	2.5	0.6	4.6	***
51	Web server	2.4	0.8	2.8	0.4	2.7	**
52	WWW	2.8	0.5	2.8	0.5	0.7	
53	XAMPP	1.1	0.3	2.8	0.4	22.8	***
54	Regular expression	1.2	0.4	2.5	0.6	11.4	***
55	Session	1.4	0.6	2.4	0.6	7.1	***
56	Transmission button	2.0	0.7	2.7	0.5	4.7	***
57	Here document	1.2	0.4	1.9	0.7	5.7	***
58	File handle	1.2	0.4	2.0	0.8	6.0	***
59	Radio button	1.4	0.7	2.8	0.4	11.1	***
60	Associative array	1.1	0.3	2.0	0.8	6.4	***
	Average	1.7	0.8	2.4	0.7	3.6	***

*** p<.001, ** p<.01, * p<.05, + p<.1

of familiarity with 50 individual terms (Table 2). These terms were as follows: 1-9, 13-23, 25-32, 34, 37-40, 42-48, 50, 51, 53-60. Knowledge of these 50 terms was found to have increased after the course. In addition, there was a trend toward significance for three terms (10, 24, and 33). This indicates that knowledge of these three terms tended to increase after the course. There was a significant difference or a trend toward significance for 88.3% of terms, indicating that familiarity with almost all terms had improved.

No significant difference was found for the following seven terms: 11, 12, 35, 36, 41, 49, and 52. Knowledge of these 7 terms was found not to have increased after the course.

3.2. Results from survey of students' attitudes toward their abilities

As shown in Table 3, pre-course (Class 1) and post-course (Class 15) surveys including 55 items relating to students' attitudes toward their abilities were also conducted [4]. The following nine-point scale was used to evaluate attitude: not at all confident/interested (1 point), not very confident/interested (3 points), somewhat confident/interested (5 points), quite confident/interested (7 points), and very confident/interested (9 points). Twenty-four students each responded to the pre-course and the post-course surveys.

The mean overall score for the 55 items was 4.3 before the course and 4.9 after the course. Analysis with paired t-tests revealed a significant difference between overall pre-course and post-course scores for the 55 items (last line of Table 3). This showed that overall, students' attitudes toward their abilities improved after the course.

Results of paired t-test analyses of pre-course and post-course attitude scores for each ability are shown in Table 3. Analysis of each item with paired t-tests revealed a significant difference between students' pre-course and post-course attitudes toward the following 20 items: 2, 3, 16, 27, 32, 36, 38–44, 46–48, and 52–55. This showed that students' attitudes toward these 20 items improved after the course. In addition, there was a trend toward significance for five items (5, 8, 11, 37, and 45). This showed that students' attitudes toward these five abilities tended to improve after the course.

A significant difference in attitude or a trend toward significance was observed for 25 abilities, which indicates that students' attitudes improved after the course for 45% of the 55 items. No significant difference in attitude or trend toward significance was observed for 30 items.

3.3. Categories of attitude determined by cluster analysis of attitudes toward programming abilities

Twenty-five items relating to programming were used as the rows and 24 students as the columns. A 25 row x 24 column spreadsheet was created to assess increases in attitude scores for programming. This table was analyzed by cluster analysis with Ward's method using attitudes as cases and students as variables. Based on the obtained dendrogram, attitudes were classified into five clusters as shown in Figure 1. These groups are numbered I–V. The x axis of Figure 1 shows non-similarity and the y axis shows attitudes.

Group I comprised attitudes toward seven items: "37. Ability to think about algorithms," "38. Ability to review the flow of an algorithm," "36. Ability to express an idea as an algorithm," "32. Knowledge of programming," "35. Ability to think about a problem in stages," "43. Ability to work to improve a program," and "51. Ability to keep working on a problem until it is finished." For each item, significance levels are indicated with a symbol after the result of the t test (*** p<.001, ** p<.01, * p<.05, + p<.1). The mean attitude score for these seven items was 0.91, which is somewhat lower than the overall mean score but still a moderate score. Of these, scores for items 36 and 38 were relatively higher. Based on its constituent items, Group I can be characterized as "I. Abilities relating to expression and flow of algorithms."

Group II comprised attitudes toward five items: "49. Ability to collaborate on problems," "50. Desire to learn about programming through problems," "33. Desire to learn about programming," "34. Desire to try problems," and "31. Interest in programming."

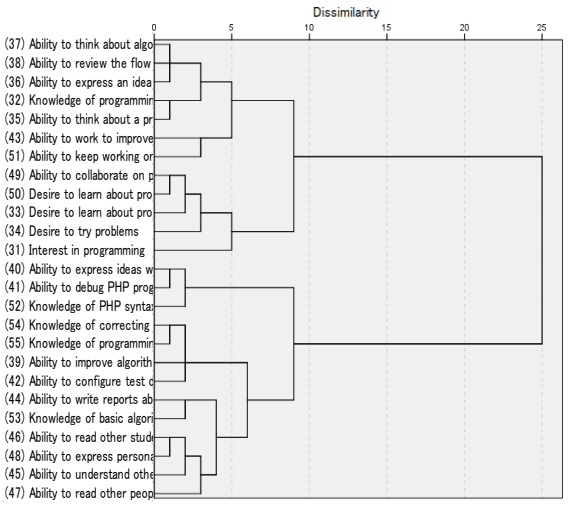

Figure 1. Dendrogram showing attitude clusters obtained through cluster analysis

The mean score for these five items was 0.06, and this was the lowest-scored group. There was no

significant difference between attitude scores for any of these five items before and after the course, indicating that attitudes did not improve. Based on its constituent items, Group II can be characterized as "II. Ability to work on problems."

Group III comprised attitudes toward three items: "40. Ability to express ideas with PHP," "41. Ability to debug PHP programs," and "52. Knowledge of PHP syntax." The mean attitude score for these three items was 2.5, and this group had the highest score. Attitudes toward each of the three items significantly increased, indicating that attitude improved. Based on its constituent items, Group III can be characterized as "III. Ability to express ideas with PHP."

Table 3. Significance tests for attitudes toward abilities

Attitude items	Pre		Post		Difference		t-test	
	m	SD	m	SD	m	SD	t	p
(1) Interest in and curiosity about computers	7.0	2.0	6.0	2.2	−0.58	2.9	0.9	
(2) Understanding of computers	4.4	1.6	5.1	1.7	1.04	2.2	2.1	*
(3) Computer operation skills	4.5	1.7	5.5	1.8	1.29	2.3	2.5	*
(4) Computer usage methods and broadening of situations	5.1	1.9	5.8	2.1	0.96	2.5	1.7	
(5) Ability to set challenges, ability to discover problems	3.9	2.2	4.8	1.7	1.12	2.5	2.0	+
(6) Ability to plan, to do things in a planned manner	3.9	1.8	4.5	2.1	0.89	2.5	1.6	
(7) Cultivation of understanding of knowledge learned	4.2	1.4	4.6	2.0	0.64	2.3	1.2	
(8) Ability to study by oneself, ability to learn	4.3	1.8	4.9	1.7	0.87	2.1	1.8	+
(9) Ability to gather information, ability to conduct research	4.7	2.0	5.0	2.0	0.64	2.4	1.2	
(10) Ability to sort through related information or data	4.5	1.5	4.8	1.7	0.67	2.1	1.4	
(11) Ability to analyse information	4.2	1.9	4.8	1.7	0.79	2.0	1.7	+
(12) Ability to express thoughts in writing	4.2	2.1	4.6	2.1	0.65	2.5	1.2	
(13) Ability to express thoughts through media other than writing	4.7	1.9	4.9	1.9	0.47	2.5	0.8	
(14) Ability to speak and explain things to others in an easy-to-understand manner	4.4	2.0	4.2	1.8	0.11	2.1	0.2	
(15) Ability to make presentations	4.0	1.9	4.2	1.8	0.47	2.7	0.8	
(16) Ability to listen to what people are saying and ability to ask people questions	4.3	1.8	5.3	2.1	1.25	2.5	2.2	*
(17) Communication ability	4.3	2.4	4.7	2.1	0.64	3.3	0.9	
(18) Ability to appropriately self-evaluate one's thoughts	4.2	1.7	4.7	2.1	0.71	2.2	1.4	
(19) Ability to appropriately evaluate other people's thoughts	5.1	1.9	4.9	1.6	0.17	2.4	0.3	
(20) Ability to correct and improve on one's own thoughts	4.6	1.8	4.5	1.9	0.23	2.5	0.4	
(21) Ability to pursue matters deeply, ability to explore matters	4.5	1.5	4.9	1.9	0.71	2.1	1.5	
(22) Ability to execute, ability to practice, ability to put into action	4.6	1.5	4.8	1.9	0.45	2.4	0.8	
(23) Ability to cooperate with others, ability to study in cooperation with others	5.2	1.4	4.7	1.6	−0.15	2.2	0.3	
(24) Sense of accomplishment, sense of satisfaction	5.2	2.0	4.8	2.0	−0.09	3.3	0.1	
(25) Sense of fulfilment, sense of achievement	5.2	1.9	5.0	2.0	0.21	3.2	0.3	
(26) Ability to solve problems	4.5	1.8	4.9	1.7	0.70	2.0	1.6	
(27) Ability to construct and create knowledge	4.0	1.7	5.0	1.9	1.22	2.5	2.2	*
(28) Ability to think, consider and come up with ideas by oneself	5.0	2.1	5.1	1.9	0.46	3.0	0.7	
(29) Creativity/ability to create	4.6	1.8	4.9	2.1	0.57	2.5	1.0	
(30) Interest in and curiosity about this field	5.7	1.8	5.2	1.8	−0.06	2.5	0.1	
(31) Interest in programming	6.2	1.8	5.5	2.2	−0.25	2.9	0.4	
(32) Knowledge of programming	4.2	1.6	4.7	1.7	0.80	1.6	2.2	*
(33) Desire to learn about programming	5.8	1.5	5.3	2.2	−0.09	2.6	0.2	
(34) Desire to try problems	5.6	1.9	5.0	1.7	−0.27	2.8	0.4	
(35) Ability to think about a problem in stages	4.5	1.7	4.8	1.7	0.58	2.3	1.1	
(36) Ability to express an idea as an algorithm	3.6	1.9	4.6	1.8	1.23	2.2	2.5	*
(37) Ability to think about algorithms	3.7	1.9	4.5	1.6	1.06	2.4	2.0	+
(38) Ability to review the flow of an algorithm	3.6	1.8	4.5	1.6	1.15	2.1	2.4	*
(39) Ability to improve algorithms	3.3	1.9	4.8	1.8	1.74	2.3	3.3	**
(40) Ability to express ideas with PHP	2.6	1.9	5.1	1.9	2.60	2.7	4.2	***
(41) Ability to debug PHP programs	2.2	1.5	4.8	1.9	2.64	2.5	4.8	***
(42) Ability to configure test data	2.5	1.9	5.0	1.8	2.65	2.2	5.3	***
(43) Ability to work to improve a program	4.0	2.2	5.2	1.7	1.41	2.7	2.3	*
(44) Ability to write reports about programs	3.3	2.2	4.5	1.6	1.35	2.3	2.6	*
(45) Ability to understand other people's ideas	4.2	1.6	4.8	1.7	0.88	2.3	1.7	+
(46) Ability to read other student's programs	3.5	1.9	4.9	1.7	1.56	2.4	2.9	**
(47) Ability to read other people's reports	3.5	2.2	5.2	1.8	1.93	2.8	3.0	**
(48) Ability to express personal ideas using a computer	3.4	1.8	4.9	1.5	1.70	2.2	3.5	**
(49) Ability to collaborate on problems	5.0	1.7	5.0	1.8	0.35	2.5	0.6	
(50) Desire to learn about programming through problems positively	4.9	1.7	5.1	1.6	0.56	2.0	1.2	
(51) Ability to keep working on a problem until it is finished	5.4	1.9	5.1	1.8	0.11	2.6	0.2	
(52) Knowledge of PHP syntax	2.9	1.8	5.0	1.9	2.27	2.6	3.9	***
(53) Knowledge of basic algorithms	3.6	1.6	4.7	1.4	1.33	2.1	2.8	*
(54) Knowledge of correcting program errors	3.4	1.9	5.1	1.6	1.86	2.4	3.5	**
(55) Knowledge of programming techniques	3.0	1.9	4.8	1.6	1.96	2.1	4.3	***
Total	4.3	2.0	4.9	1.9	0.87	1.8	2.2	*

*** p<.001, ** p<.01, * p<.05, + p<.1

Group IV comprised attitudes toward four items: "54. Knowledge of correcting program errors," "55. Knowledge of programming techniques," "39.configure test data." The mean attitude score for these three items was 2.05, and this group had the second-highest score. Attitudes toward each of the four items significantly increased, indicating that attitudes improved. The score for item 42 was relatively higher. Based on its constituent items, Group IV can be characterized as "IV. Knowledge of programming techniques."

Group V comprised attitudes toward six items: "44. Ability to write reports about programs," "53. Knowledge of basic algorithms," "46. Ability to read other people's programs," "48. Ability to express personal ideas using a computer," "45. Ability to understand other people's ideas," and "47. Ability to read other people's reports." The mean attitude score for these six items was 1.46, and this group had the third-highest score. Attitudes toward five items significantly increased, indicating that attitudes improved. There was a trend toward significance for attitude toward the remaining item (45), which tended to improve. The attitude scores for items 47 and 46 were relatively higher. Based on its constituent items, Group V can be characterized as "V. Ability to read other people's programs and reports."

3.4. Categories of student determined by cluster analysis of attitudes toward programming abilities

Attitude scores for programming items were analyzed in the 25 row by 24 column table from Section 3.3 with Ward's method using students as cases and the attitudes as variables. Based on the obtained dendrogram, students were classified into four clusters as shown in Figure 2. These groups were numbered 1–4. The x axis of Figure 2 shows dissimilarity and the y axis shows students.

Group 1 comprises three students whose attitude score decreased and is called "1. Worsened attitude." Group 2 comprises three students whose attitude scores greatly improved and is called "2. Greatly improved attitude." Group 3 comprises ten students whose attitude scores slightly improved and is called "3. Slightly improved attitude." Group 4 comprises eight students whose attitude scores did not change and is called "4. No change in attitude."

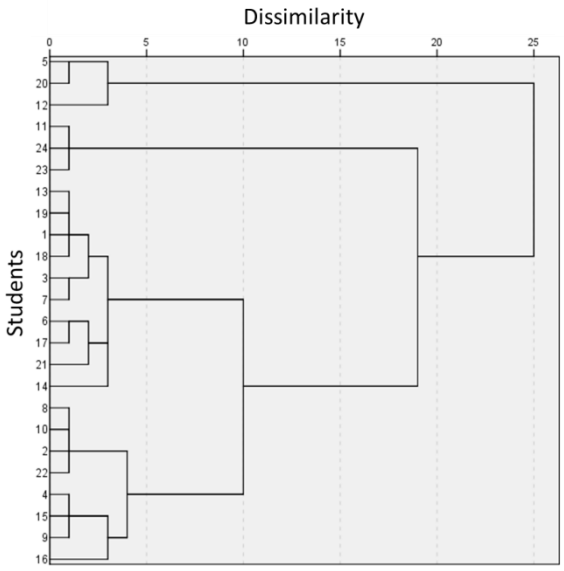

Figure 2. Dendrogram showing student clusters obtained through cluster analysis

4. Discussion

4.1. Changes in familiarity with terminology and reasons for these changes

Analysis with paired t-tests revealed that students' overall familiarity with the 60 terms was significantly higher after the course, indicating that students' overall knowledge of programming increased after the course. In addition, analysis of individual terms with paired t-tests revealed that there was either a significant difference or a trend toward significance for 88.3% of terms, indicating that familiarity with almost all terms had improved. It appears that students learned many terms and understood them after listening to lectures, listening to explanations of example programs, and actually creating programs.

No significant difference was found for the following seven terms: "do...while" (11), "else" (12), "if" (35), "include" (36), "return" (41), "URL" (49), and "WWW" (52). The mean familiarity with these terms before the course was 2.6, 2.8, 2.7, 2.3, 2.6, 2.7, and 2.8, respectively. Students already learned five of these terms (11, 12, 35, 36, and 41) in the C language and thus knew them relatively well. They had also learned terms (49) and (52) in classes related to the internet. Therefore, the reason why there was no significant difference is that their familiarity with these terms before the course was close to 3 and only increased slightly.

4.2 Changes in students' attitudes toward their abilities and reasons for these changes

Attitudes can be categorized into attitudes toward general abilities (1–30) and attitudes toward programming abilities (31–55). In the category of attitudes toward general abilities, a significant difference was observed for four items and a trend toward significance for three items. The ratio of items that improved was 7/30=0.23. Significant differences were observed for the following items: "2. Understanding of computers,", "3. Computer operating skills," "16. Ability to listen to others," and "27. Ability to organize knowledge." Trends toward significance were observed for the following items: "5. Ability to design a problem," "8. Ability to learn," and "11. Ability to analyze information." Items 2 and 3 likely improved because students created programs on a computer in class. Items 8 and 16 likely improved because students had to carefully listen to explanations of example programs and syntactic elements in the lectures in order to complete assignments. Items 5, 11, and 16 likely improved because students performed exercises where they created programs based on example programs and assigned programs. As just described, abilities related to participating in the class appeared to improve.

In the category of attitudes toward programming abilities, a significant difference was observed for 16 items and a trend toward significance for two items. The ratio of items that improved was 18/25=0.72. The mean increase in score was 0.57 for attitudes toward general abilities and 1.24 for attitudes toward programming abilities. Analysis of the mean increase in these two scores with paired t-tests revealed a significant difference. This indicates that attitudes toward programming abilities improved more than attitudes toward general abilities ($t(53)=3.6$, $p<0.001$). Therefore, a greater proportion of students showed improvement in their attitude toward their programming abilities and the mean increase in score was also greater.

It can be concluded that although the course methods described in Section 2 do not really improve students' attitudes toward general abilities, they do improve their attitudes toward programming abilities. Strategies to get students actively involved in class will be necessary to improve attitudes toward general abilities in the future. One thing that the author of this paper wishes to do is to include easy assignments for students with poor understanding to inspire them to create programs.

4.3 Attitude clusters as classified by cluster analysis of students' attitudes toward their programming abilities

The number of items composing the five clusters obtained in Section 3.3 for which attitude scores significantly improved, as well as the mean score and standard deviation for each cluster, is shown in Table 4. The groups "III. Ability to express ideas with PHP," "IV. Knowledge of programming techniques," and "V. Ability to read other people's programs and reports" had scores higher than the overall mean. Significant differences were observed for all items in these clusters. In addition, the mean scores of all items in these clusters improved.

There was no increase for any item in Group II ("Ability to work on problems") or items 35 ("Ability to think about a problem in stages") and 51 ("Ability to keep working on a problem until it is finished") in Group I ("Abilities relating to expression and flow of algorithms"). Although overall attitudes toward programming improved, students' attitudes toward their ability to keep working on problems until they are finished did not improve. Their attitudes did not improve because of assignments. Specifically, considering that the percent of assignments submitted declined in the second half of the course, it is possible that students' attitudes toward this ability did not improve because they found assignments too difficult to finish and lost motivation to try assigned problems. Therefore, strategies such as including easy assignments among the programs students are to create will be necessary.

Table 4. Mean scores and standard deviations for attitude clusters

Cluster name	Item numbers	No. of items	No. of significant difference & tendency	m	SD
I. Abilities relating to expression and flow of algorithms	32,35,36,37,38,43,51	7	5	0.91	2.34
II. Ability to work on problems	31,33,34,49,50	5	0	0.06	2.61
III. Ability to express ideas with PHP	40,41,52	3	3	2.50	2.62
IV. Knowledge of programming techniques	39,42,54,55	4	4	2.05	2.28
V. Ability to read other people's programs and reports	44,45,46,47,48,53	6	6	1.46	2.39
Total	-----	25	18	1.24	2.56

4.4. Student clusters as classified by cluster analysis of attitudes toward programming abilities

The mean scores for attitudes toward programming abilities and mean scores for attitudes toward general abilities of the four student clusters obtained in Section 3.4, along with mean familiarity with terminology, are shown in Table 5.

The correlation coefficient of the mean increases in score for attitudes toward general abilities and attitudes toward programming abilities was r=0.90***, indicating a significantly strong correlation.

However, the correlation coefficient of the mean score for attitudes toward programming abilities and mean familiarity with terminology was -0.12, indicating no significant correlation. This signifies that an increase in attitude score is not associated with an increase in familiarity with terminology, or, in other words, an increase in knowledge. Furthermore, this signifies that knowledge does not necessarily increase when attitudes improve. This indicates that although both knowledge and attitudes generally improved, this increase in knowledge was not associated with an improved attitude for some individual students even though it was for other students.

The same relationships can be observed in Table 8; even within the four student groups, it can be inferred that there is a relationship between mean increases in attitude scores for general abilities and programming abilities but not between mean attitude scores for programming abilities and mean familiarity with terminology.

The range of the mean familiarity with terminology of the group is from 0.61 to 0.83. As a result of having done an analysis of variance about four student groups; it was admitted that the mean familiarity with terminology of "2. Greatly improved attitude G" and "4. No change in attitude G" was significantly higher than "3. Slightly improved attitude G". Significant difference is not recognized between others. Student group G4 which had strongest consciousness was the highest mean familiarity with terminology. However, student group G3 which had secondly strongest consciousness was the lowest mean familiarity with terminology.

Table 5. Mean scores and standard deviations for student clusters

Cluster name	No.of Students	Rating of terms 31 to 55		Rating of terms 1 to 30		Familiarity with terminology	
		m	SD	m	SD	m	SD
1. Worsened attitude	3	−2.1	1.6	−1.8	2.0	0.71	0.88
2. Greatly improved attitude	3	4.5	1.1	4.2	1.4	0.83	0.85
3. Slightly improved attitude	10	2.0	2.0	0.9	2.0	0.61	0.95
4. No change in attitude	8	0.3	1.9	−0.5	1.9	0.79	0.94
Average	—	1.24	0.86	0.57	0.44	0.68	0.35

5. Conclusions

Students were taught with lectures and exercises, reviewed concepts with lecture slides on an e-learning site, and submitted assignments as part of programming education at a university. Students' attitudes and their familiarity with terminology were assessed with surveys conducted before and after the course. Data from these surveys were analyzed with significance tests and cluster analysis.

The following was found after conducting the course. These findings could also be a useful resource for other courses.

(1) Students' overall familiarity with programming terms was significantly higher after the course, indicating that their overall knowledge increased after the course.

(2) Students' familiarity with approximately 83% of the 60 terms increased after the course.

(3) In general, students' attitudes toward their abilities improved.

(4) Students' attitudes toward about approximately 45% of the 55 items improved after the course.

(5) A significant difference or a trend toward significance was observed for 23% of items in the category of attitudes toward general abilities and 72% of items in the category of attitudes toward programming abilities. The reason for this is likely that the increase in attitude scores was greater for programming abilities.

(6) An increase in attitude score is not associated with an increase in familiarity with terminology.

(7) In the four student groups obtained from cluster analysis, mean attitude scores for programming abilities were not related to mean familiarity with terminology.

(8) In the five clusters for attitudes toward programming, mean scores for Group III ("Ability to express ideas with PHP"), Group IV ("Knowledge of

programming techniques"), and Group V ("Ability to read other people's programs and reports") were higher than the overall mean, and the mean scores for all items in these clusters had improved after the course.

As a future challenge, the authors of this paper would like to study how to apply the findings of this study to their teaching.

The authors appreciates the support of the Grant-in-Aid for Scientific Research, foundation study (C25350364) provided by the Ministry of Education, Culture, Sports, Science and Technology, Japan for this research.

References

[1] Bersin, J. (2004) *The Blended Learning Book: Best Practices, Proven Methodologies, and Lessons Learned*, John Wiley & Sons, San Francisco, USA,.

[2] Miyaji, I. (2011) 'Comparison between Effects in Two Blended Classes Which E-learning is Used inside and outside Classroom', *US-China Education Review*, 8 (4), pp.468-481.

[3] Miyaji, I. (Ed.) (2009) *Toward Blended Learning from E-learning*, Kyouritu-Shuppan, Tokyo, Japan.

[4] Miyaji, I. and Yoshida, K. (2005) 'The Practice and Learning Effect of Education by Blending of Lecture and E-learning', *Transactions of Japanese Society for Information and Systems in Education*, 22 (4), pp.230-239.

[5] Miyaji, I. (2009) 'Effects on Blended Class Which Incorporates E-learning Inside the Classroom', in Proceeding of *the 20th World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education, E-learn 2009*, Vancouver, pp.1818-1826.

[6] Miyaji, I., Yoshida, K., and Naruse, Y. (2007) 'The Effects of Blending E-learning and Lectures Utilizing a Structured Notebook', *Transactions of Japanese Society for Information and Systems in Education*, 4 (3), pp.208-215.

[7] Takaoka, E. and Ishii, W. (2008) 'Fully E-learning Java Programming Course: Design, Development and Assessment', *Transactions of Japanese Society for Information and Systems in Education*, 25 (2), pp.214-225.

[8] Shinkai, J., Miyaji, I. (2011) 'Effects of Blended Instruction on C Programming Education', *Transactions of Japanese Society for Information and Systems in Education*, 28 (2), pp.151-162,.

[9] Taniguchi, R. (2011) 'Collaborative Learning through Sharing Students' Work Information and Evaluation of This Learning Environment', *Transactions of Japanese Society for Information and Systems in Education*, 28 (4), pp.283-291.

[10] Anku (2011) *Illustrated Book of PHP*, Shoeisha, Tokyo, Japan.

