Useful Activities for Improving the Attitudes and Characteristic of Student Groups in Programming Course

Isao Miyaji

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. They could learn either in class or through lecture slides uploaded to an e-learning site. Students' attitudes were assessed before and after the course. The attitudes and activities were analyzed with cluster analysis. Useful activities for improving the attitudes in a programming course were found by chi-square analysis of cross-tabulation of attitude and activity clusters. Principal component analysis of attitudes was conducted. Students were classified based on principal components of attitude scores. Characteristic of student groups is explained. The findings are reported in this paper.**

Keywords- programming course; useful activities; attitude; blended learning; exercise, assignment

I. 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 [2] [6]. The author of this paper promotes a university education that includes creating things and evaluating them in order to build problem-solving skills [6]. 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 [3] [5]. The author also found that using comprehension surveys and increasing interactions between students and faculty can further enhance results [4].

Several methods to deepen students' understanding in programming class have been proposed [11]. One method that has been reported to be effective is blended learning classes [12]. There are also reports of students collaborating on projects and then evaluating them [11].

In this study, a professor conducted blended classes that utilized e-learning while considering what media are required for a programming class [7]. 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 and revised their program based on the peer assessment after reviewing each other's programs. Students were surveyed about their thoughts regarding this system and results were reported [8]. Students' scope of knowledge has been measured using a metric such as familiarity with terminology. The results of conducting and analyzing pre- and post-course surveys of students' attitudes and their familiarity with terminology were reported [9] [10].

In this paper, useful activities for improving the attitudes in this programming course were found by chi-square analysis of cross-tabulation of attitude and activity clusters. Principal component analysis of attitudes was conducted. Students were classified based on principal components of attitude scores. Characteristic of student groups is explained.

DOI: 10.5176/2345-7163_3.1.66

II. 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 I. 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.

A. 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.

B. 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 [1] 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.

C. Class Format 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 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.

TABLE I. 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	ependent 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

D. E-learning Contents

The e-learning system that was prepared and made available to students offered the following capabilities: (1) viewing of lecture slides for studying, (2) viewing of independent project reports, (3) download of example programs and problems, (4) download of practice problems to review syntactic elements, (5) materials for download, (6) an assignment upload function, and (7) a message board.

E. Types of Media Used

The course was conducted using the following media: (1) written explanations of lecture content, (2) lecture slides, (3) instructions for writing PHP programs, (4) sheets explaining example programs and assignments, (5) practice problems to review syntactic elements, (6) explanation of how to design independent projects, (7) an evaluation sheet file, (8) a report outline file, (9) slideshow files explaining example programs and assignments, and (10) e-learning.

III. Results of Analysis

Students' attitudes toward their abilities were assessed to understand how their attitudes changed after they took the programming course. The number of attitudes is 55. They were asked to select helpful activities and write them to the right of their attitude score on the post survey. The number of activities is 33. Data from these surveys were analyzed with significance tests and the results of this analysis are explained below. Changes in attitude and activities for improving attitude were analyzed using multivariate analysis.

In the following results, a significance level of 5% was considered to indicate a significant difference. Significance levels of 0.1%, 1%, 5%, and 10% are indicated with ***, **, *, and +, respectively.

A. Classification of Attitudes and Helpful Activities with Cluster Analysis

In a follow-up study, students were asked to select helpful activities from a list of 33 items as shown in TABLE III and write them to the right of their attitude score. The number of activities listed was then cross-tabulated with attitudes and activities. The total number of activities listed was 2224 (92.7 per person). The cross-tabulation table of attitude and activities was 55 rows by 33 columns. The number of activities in each cell of the cross-tabulation table was small (approximately 1.2) which made it difficult to determine helpful activities that way. Then cluster analysis is conducted to classify attitude and activities respectively. The results are explained in the following. Attitude items are denoted with a number put in parentheses (e.g., "(1)") and activities are denoted with the number alone (e.g., "1.").

1) Attitude Categories

The 55 × 33 cross-tabulation table was analyzed by using attitudes as cases and activities as variables. Based on the obtained dendrogram, attitudes were classified into four clusters as shown in TABLE II. Group I comprised attitudes toward nine abilities: "(5) Ability to design a program," "(6) Ability to do things systematically," "(9) Ability to gather information," "(1) Interest in computers," "(2) Knowledge of computers," "(3) Computer operating skills," "(4) Expanding how to use computers and usage of computers," "(7) Broadening depth of knowledge," and "(8) Ability to learn." These are only general abilities. The number of activities listed ranged from 44 to 54. Among these, the most activities were listed for (2), (3), (5), and (6). Based on its constituent items, Group I can be characterized as "I. Understanding of computers and ability to systematically set tasks."

Group II comprised attitudes toward 21 items: "(36) Ability to express an idea as an algorithm," "(38) Ability to review the flow of an algorithm," "(35) Ability to think about things in stages," "(37) Ability to think about algorithms," "(31) Interest in programming," "(32) Knowledge of programming," "(54) Knowledge of correcting program errors," "(52) Knowledge of PHP syntax," "(18) Ability to self-evaluate," "(24) Level of satisfaction," "(11) Ability to analyze information," "(10) Ability to organize and summarize information," "(39) Ability to improve algorithms," "(17) Ability to communicate," "(25) Feeling of accomplishment," "(41) Ability to debug PHP programs," "(55) Knowledge of programming techniques," "(26) Problem-solving abilities," "(14) Ability to explain things," "(40) Ability to express ideas with PHP," and "(30) Interest in the field." Twelve of these items were programming abilities. The number of activities listed ranged from 37 to 44, and almost the same number of activities was listed for each item. Based on these 21 items, Group II can be characterized as "II. Attitudes toward programming techniques."

Group III comprised attitudes toward five items: "(19) Ability to evaluate others," "(20) Ability to revise and improve," "(46) Ability to read other people's programs," "(47) Ability to read other people's reports," and "(45) Ability to understand other people's ideas." Three of these items were programming abilities. The number of activities listed ranged from 37 to 40 and almost the same number of activities was listed for each item. Based on these five items, Group III can be characterized as "III. Attitudes toward evaluating other people's work."

Group IV comprised attitudes toward 20 items: "(21) Ability to do deep research," "(48) Ability to express ideas using a computer," ""(29) Ability to create things," "(53) Knowledge of basic algorithms," "(27) Ability to organize knowledge," "(28) Ability to think independently," "(34) Desire to try problems," "(49) Ability to collaborate," "(33) Desire to learn about programming," "(43) Ability to improve a program to make it good," "(23) Ability to learn collaboratively," "(42) Ability to configure test data," "(15) Ability to give a presentation," "(16) Ability to listen to others," "(22) Ability to see things through," "(51) Ability to keep working on something until it is finished," "(12) Ability to express ideas in writing," "(44) Ability to write reports about programs," "(13) Ability to express ideas in non-written form," and "(50) Ability to proactively work on problems." Ten of these items were attitudes about programming. The number of activities listed ranged from 34 to 42, and almost the same number of activities was listed for each item. Based on these 20 items, Group IV can be characterized as "IV. Attitudes toward working on assignments."

TABLE II. NUMBER OF HELPFUL ACTIVITIES LISTED FOR EACH ATTITUDE ITEM

Groups	Attitudes toward abilities	Frequence	Sum
I. Understanding of computers and ability to systematically set tasks	(5) Ability to set challenges, ability to discover problems	48	425
	(6) Ability to plan, to do things in a planned manner	48	
	(9) Ability to gather information, ability to conduct research	44	
	(1) Interest in and curiosity about computers	46	
	(2) Understanding of computers	54	
	(3) Computer operation skills	48	
	(4) Computer usage methods and broadening of situations	47	
	(7) Cultivation of understanding of knowledge learned	45	
	(8) Ability to study by oneself, ability to learn	45	
II. Attitudes toward programming techniques	(36) Ability to express an idea as an algorithm	40	842
	(38) Ability to review the flow of an algorithm	41	
	(35) Ability to think about a problem in stages	40	
	(37) Ability to think about algorithms	41	
	(31) Interest in programming	43	
	(32) Knowledge of programming	44	
	(54) Knowledge of correcting program errors	41	
	(52) Knowledge of PHP syntax	41	
	(18) Ability to appropriately self-evaluate one's thoughts	39	
	(24) Sense of accomplishment, sense of satisfaction	37	
	(11) Ability to analyse information	39	
	(10) Ability to sort through related information or data	41	
	(39) Ability to improve algorithms	40	
	(17) Communication ability	37	
	(25) Sense of fulfilment, sense of achievement	36	
	(41) Ability to debug PHP programs	39	
	(55) Knowledge of programming techniques	38	
	(26) Ability to solve problems	40	
	(14) Ability to speak and explain things to others in an easy-to-understand manner	41	
	(40) Ability to express ideas with PHP	44	
	(30) Interest in and curiosity about this field	40	
III. Attitudes toward evaluating other people's work	(19) Ability to appropriately evaluate other people's thoughts	39	194
	(20) Ability to correct and improve on one's own thoughts	40	
	(46) Ability to read other student's programs	37	
	(47) Ability to read other people's reports	39	
	(45) Ability to understand other people's ideas	39	
IV. Attitudes toward working on assignments	(21) Ability to pursue matters deeply, ability to explore matters	34	763
	(48) Ability to express personal ideas using a computer	39	
	(29) Creativity/ability to create	35	
	(53) Knowledge of basic algorithms	38	
	(27) Ability to construct and create knowledge	41	
	(28) Ability to think, consider and come up with ideas by oneself	38	
	(34) Desire to try problems	41	
	(49) Ability to collaborate on problems	39	
	(33) Desire to learn about programming	38	
	(43) Ability to work to improve a program	39	
	(23) Ability to cooperate with others, ability to study in cooperation with others	35	
	(42) Ability to configure test data	37	
	(15) Ability to make presentations	37	
	(16) Ability to listen to what people are saying and ability to ask people questions	39	
	(22) Ability to execute, ability to practice, ability to put into action	42	
	(51) Ability to keep working on a problem until it is finished	40	
	(12) Ability to express thoughts in writing	42	
	(44) Ability to write reports about programs	36	
	(13) Ability to express thoughts through media other than writing	37	
	(50) Desire to learn about programming through problems positively	36	
Total		2224	2224

2) Activity Categories

The same cross-tabulation table was also analyzed by cluster analysis with Ward's method in the opposite direction from the previous section, using activities as cases and attitudes as variables, and the obtained dendrogram was then used to classify activities into Clusters 1 through 4 as shown in TABLE III. Group 1 comprised ten activities: "19. Asking the professor questions about program creation assignments," "20. Asking the TA questions about program creation assignments," "33. Other," "30. Revising programs based on peer review," "26. Creating programs based on research," "17. Reading the message board," "24. Thinking about specifications for independent projects," "23. Doing research about what to include in programs to be created," "18. Asking about program creation assignments," and "22. Deciding what to include in programs to be created." The number of times an activity was listed ranged from 32 to 84 times. The most frequently listed activities were 23, 26, and 30. Based on its constituent items, Group 1 can be characterized as "1. Activities related to research, revision, and asking questions." These activities were listed 560 times.

Group 2 comprised six activities: "05. Preparing," "06. Reviewing," "10. Studying using practice problems," "21. Asking friends questions about program creation assignments," "03. Asking friends questions about lecture topics," and "07. Studying using the textbook." The number of times an activity was listed ranged from 98 to 210 times. The most frequently listed activities included 7, 21, 3, 5, and 6. Based on its constituent items, Group 2 can be characterized as "2. Activities that involve studying with the textbook." These activities were listed 728 times. These six activities, when put together, were about as helpful as "1. Listening to lectures" in Group 4.

TABLE III. NUMBER OF TIMES EACH ACTIVITY WAS LISTED

Groups	Activities	Frequence	Sum
1. Activities related to research, revision, and asking questions	19. Asking the professor questions about program creation assignments	60	560
	20. Asking the TA questions about program creation assignments	62	
	33. Other	57	
	30. Revising programs based on peer review	64	
	26. Creating programs based on research	73	
	17. Reading the message board	46	
	24. Thinking about specifications for independent projects	46	
	23. Doing research about what to include in programs to be created	84	
	18. Asking about program creation assignments	32	
	22. Deciding what to include in programs to be created	36	
2. Activities that involve studying with the textbook	05. Preparing	104	728
	06. Reviewing	102	
	10. Studying using practice problems	98	
	21. Asking friends questions about program creation assignments	108	
	03. Asking friends questions about lecture topics	106	
	07. Studying using the textbook	210	
3. Activities that involve using applications as well as studying and evaluating other people's programs	14. Using Excel	32	231
	15. Using Word	15	
	16. Writing on the message board	17	
	27. Summarizing created programs with reports	12	
	04. Asking the professor about lecture topics	12	
	29. Running other people's programs and evaluating them	8	
	25. Revising design specifications for independent projects	6	
	32. Evaluating abilities and attitudes	9	
	02. Getting a broad understanding of the lecture	17	
	09. Evaluating in-class slides as a learning method	5	
	11. Evaluating practice problems as a learning method	10	
	13. Asking questions using e-mail	8	
	08. Learning using class slides	12	
	31. Revising programs while referencing other people's programs	24	
	12. Studying for the final exam	25	
	28. Viewing and evaluating other people's programs	19	
4.Listening to lectures	01. Listening to lectures	705	705
Total		2224	2224

Group 3 comprised 16 activities: "14. Using Excel," "15. Using Word," "16. Writing on the message board," "27. Summarizing created programs with reports," "04. Asking the professor about lecture topics," "29. Running other people's programs and evaluating them," "25. Revising design specifications for independent projects," "32. Evaluating abilities and attitudes," "02. Getting a broad understanding of the lecture," "09. Evaluating in-class slides as a learning method," "11. Evaluating practice problems as a learning method," "13. Asking questions using e-learning," "08. Learning using class slides," "31. Revising programs while referencing other people's programs," "12. Studying for the final exam," and "28. Viewing and evaluating other people's programs." The number of times an activity was listed ranged from 5 to 32 times. The most frequently listed activities were 14, 12, and 31. Based on its constituent items, Group 3 can becharacterized as "3. Activities that involve using applications as well as studying and evaluating other people's programs." These activities were listed 231 times. Although Group 3 contained the largest number of activities, these activities were listed the least frequently, indicating that these activities had little influence on improving attitudes.

Group 4 consisted of one activity: "1. Listening to lectures." Therefore, Group 4 can be characterized as "4. Listening to lectures." This activity was listed 705 times. "1. Listening to lectures" was the most listed item of all 33 items by a considerable amount, indicating that it was the most helpful activity.

B. Results from Analysis of Activities that Help to Improve Attitude

The cross-tabulation table of attitude and activities was 55 rows by 33 columns. The number of times activities were listed in each cluster obtained in the above section was totaled, and results are shown on the upper-left side of TABLE IV. Chi-square tests were performed using this table as a 4 × 4 contingency table. This showed that there was a significant bias regarding the number of times activities were listed (χ^2 (9) = 62.5, p < .001). Results from residual analysis are shown on the lower-left side of TABLE IV. Cells that contained statistically significant values with positive residuals are indicated with an asterisk (*) on the bottom-right side of TABLE IV. This analysis revealed that the activity "1. Studying and asking friend questions" was helpful in improving "IV. Attitude toward working on problems."

"2. Activities that involve studying with the textbook" were helpful in improving "I. Understanding of computers and ability to systematically set tasks."

"3. Activities that involve using applications as well as studying and evaluating other people's programs" were helpful in improving "III. Attitudes toward evaluating other people's work."

"4. Listening to lectures" was helpful in improving "II. Attitudes toward programming techniques."

TABLE IV. CHI-SQUARE ANALYSIS OF CROSS-TABULATION OF ATTITUDE AND ACTIVITY CLUSTERS

Groups of attitudes and activities	Observed frequency					Expected frequency			
	1. Activities related to research, revision, and asking questions	2. Activities that involve studying with the textbook	3. Activities that involve using applications as well as studying and evaluating other people's programs	4. Listening to lectures	Total	1. Activities related to research, revision, and asking questions	2. Activities that involve studying with the textbook	3. Activities that involve using applications as well as studying and evaluating other people's programs	4. Listening to lectures
I. Understanding of computers and ability to systematically set tasks	90	174	43	118	425	107.0	139.1	44.1	134.7
II. Attitudes toward programming techniques	216	259	71	296	842	212.0	275.6	87.5	266.9
III. Attitudes toward evaluating other people's work	34	57	45	58	194	48.8	63.5	20.2	61.5
IV. Attitudes toward working on assignments	220	238	72	233	763	192.1	249.8	79.3	241.9
Total	560	728	231	705	2224	560.0	728.0	231.0	705.0
	Adjusted residual					Significance probability			
I. Understanding of computers and ability to systematically set tasks	−2.1	4.0	−0.2	−1.9			***		
II. Attitudes toward programming techniques	0.4	−1.5	−2.4	2.7					**
III. Attitudes toward evaluating other people's work	−2.6	−1.0	6.1	−0.6				***	
IV. Attitudes toward working on assignments	2.9	−1.1	−1.1	−0.9		**			

*** p<.001, ** p<.01

C. *Principal Component Analysis of Attitudes*

Students were grouped based on improvement in attitude scores. In order to determine the characteristics of each factor and each group, attitude improvement as calculated from the difference between pre- and post-course scores for the 25 items (31) to (55) related to programming as listed in TABLE V was used to perform principal component analysis using a variance-covariance matrix. The loadings of principal components 1 and 2 were 63.4% and 8.8%, respectively. Their combined loading was 72.2%. TABLE V shows each evaluated item for Component 1 of the component matrix in descending order by the size of the coefficient.

The coefficients for evaluated items for Component 1 ranged from 0.89 to 0.59, indicating that this variable integrates all items. Based on this, Component 1 was named "general abilities required for programming." For Component 2, items such as "(44) Ability to write reports about programs" "(47) Ability to read other people's reports," "(48) Ability to express personal ideas using a computer," and "(54) Knowledge of correcting program errors" had large positive coefficients, whereas items such as "(31) Interest in programming," "(33) Desire to learn about programming," "(32) Knowledge of programming," "(35) Ability to think about a problem in stages," and "(49) Ability to collaborate on problems" had large negative coefficients. Therefore, the items with a positive tendency were named "ability to read reports" and the items with a negative tendency were named "interest in and motivation for programming." Attitude scores for programming-related items can be explained by these two components.

TABLE V. FACTOR MATRIX COEFFICIENTS FOR ATTITUDES TOWARD PROGRAMMING

Attitude items	Compo-nent 1	Compo-nent 2
(50) Desire to learn about programming through problems positively	0.89	−0.16
(38) Ability to review the flow of an algorithm	0.88	0.11
(35) Ability to think about a problem in stages	0.87	−0.36
(36) Ability to express an idea as an algorithm	0.87	−0.09
(49) Ability to collaborate on problems	0.87	−0.33
(46) Ability to read other student's programs	0.86	0.10
(54) Knowledge of correcting program errors	0.85	0.29
(37) Ability to think about algorithms	0.85	0.05
(40) Ability to express ideas with PHP	0.84	0.00
(34) Desire to try problems	0.81	−0.09
(55) Knowledge of programming techniques	0.79	0.05
(39) Ability to improve algorithms	0.79	0.10
(47) Ability to read other people's reports	0.78	0.39
(41) Ability to debug PHP programs	0.78	0.05
(53) Knowledge of basic algorithms	0.78	0.28
(52) Knowledge of PHP syntax	0.78	−0.03
(43) Ability to work to improve a program	0.77	0.06
(51) Ability to keep working on a problem until it is finished	0.77	−0.23
(33) Desire to learn about programming	0.77	−0.42
(45) Ability to understand other people's ideas	0.76	0.21
(48) Ability to express personal ideas using a computer	0.76	0.29
(32) Knowledge of programming	0.74	−0.37
(44) Ability to write reports about programs	0.73	0.50
(42) Ability to configure test data	0.71	0.48
(31) Interest in programming	0.59	−0.67

D. *Classification of Students Based on Principal Components of Attitude Scores*

The principal component scores calculated in Section IIIC for attitude score improvement were used as variables in cluster analysis. The 24 students included in the analysis were categorized into four clusters (groups). The four groups are listed in Figure 1 along with their scores for principal components 1 and 2.

Group 1 (G1, indicated with unfilled circles) comprised the four students on the far right. Their mean score was 4.1, which was the highest score of all four groups. They had the strongest general abilities required for programming. This group had the highest mean scores for all items and their abilities related to programming improved the most overall.

Group 2 (G2, indicated with squares) comprised the seven students to the right of the middle. Their mean overall score was 2.1 and their mean scores for individual items ranged from -0.1 to 4.7. They had the second highest mean score of all four groups. In this group, improvement of 2 points or more was seen for the following 15 items: "(36) Ability to express an idea as an algorithm," "(37) Ability to think about algorithms," "(38) Ability to review the flow of an algorithm," "(39) Ability to improve algorithms," "(40) Ability to express ideas with PHP," "(41) Ability to debug PHP programs," "(42) Ability to configure test data," "(43) Ability to work to improve a program," "(44) Ability to write reports about programs," "(46) Ability to read other people's programs," "(47) Ability to read other people's reports," "(48) Ability to express personal ideas using a computer," "(52) Knowledge of PHP syntax," "(54) Knowledge of correcting program errors," and "(55) Knowledge of programming techniques." An additional four items had scores higher than the overall mean score of 1.2.

Group 3 (G3, indicated with triangles) comprised the nine students to the left of the middle. Their mean overall score was 0.63 and their mean scores for individual items ranged from -1.5 to 2.7. The mean score of this group was lower than the overall mean and they did not improve very much. Improvement of 2 points or more was seen for the following five items: "42. Ability to configure test data" (m=2.7), "40. Ability to express ideas with PHP" (m=2.5), "(52) Knowledge of PHP syntax" (m=2.3), "(41) Ability to debug PHP programs" (m=2.3), and "(47) Ability to read other people's reports" (m=2.3). This indicates that their abilities related to PHP programs improved considerably. An additional five items had scores higher than the overall mean score of 1.2.

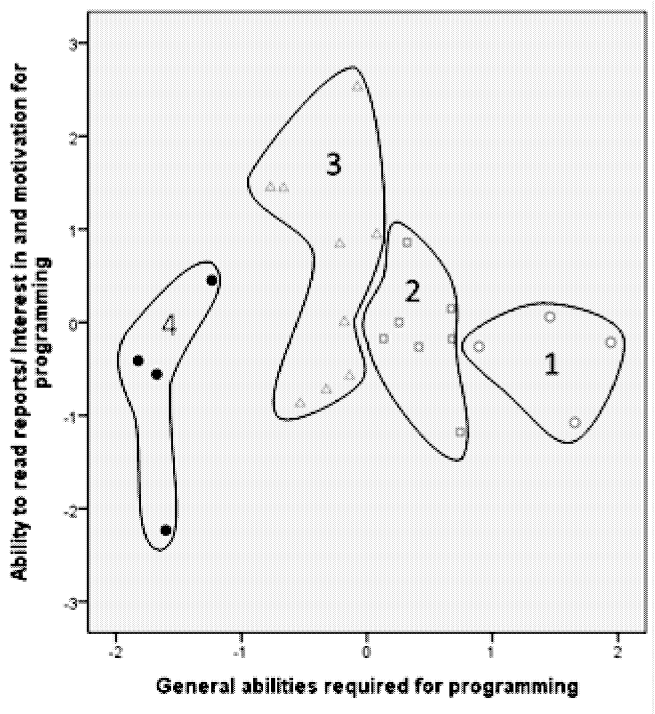


Figure 1. The four student groups shown level with the two components.

Group 4 (G4, indicated with filled circles) comprised the four students on the far left. Their mean overall score was 1.8 and their mean scores for individual items ranged from -3.3 to -1.5. This group had negative mean scores for all items and their programming abilities declined the most overall after the course. Their "(34) Desire to try problems" was particularly low and it appears that these students may have lost enthusiasm because problems were difficult for them.

E. Identification of Important Items in Each Group with Discriminant Analysis

Discriminant analysis was performed using improvement in scores to identify important items in each group.

1) Attitude Items that Contributed to Discrimination

Discriminant analysis of improvements in attitude scores for each group revealed three discriminant functions with respective eigenvalues of 62.6, 28.2, and 3.0. Two of these discriminant functions contained 96.8% of the information volume and thus standardized canonical discriminant function coefficients were based off of these two functions. For Discriminant Function 1, items with a large positive standardized canonical discriminant function coefficient were attitudes toward "(32) Knowledge of programming," "(40) Ability to express ideas with PHP," "(34) Desire to try problems," "(37) Ability to think about algorithms," "(50) Ability to proactively work on problems," "(43) Ability to work to improve a program," and "(45) Ability to understand other people's ideas." These abilities are directly related to creating a program. Items with a large negative canonical discriminant function coefficient were attitudes toward "(31) Interest in programming," "(35) Ability to think about a problem in stages," "(38) Ability to review the flow of an algorithm," "(44) Ability to write reports about programs," "(41) Ability to debug PHP programs," and "(33) Desire to learn about programming." These abilities are important for creating a program. Furthermore, attitudes toward these items greatly contributed to discrimination.

For Discriminant Function 2, items with a large positive standardized canonical discriminant function coefficient were attitudes toward "(35) Ability to think about a problem in stages," "(39) Ability to improve algorithms," "(44) Ability to write reports about programs," "(46) Ability to read other people's programs," "(31) Interest in programming," "(41) Ability to debug PHP programs," and "(33) Desire to learn about programming." These are ability necessary back and forth of programming. Items with a large negative coefficient were attitudes toward "(32) Knowledge of programming," "(34) Desire to try problems," "(47) Ability to read other people's reports," "(50) Ability to proactively work on problems," "(40) Ability to express ideas with PHP," "(49) Ability to collaborate on problems," "(38) Ability to review the flow of an algorithm," and "(42) Ability to configure test data." These are ability to be necessary when programming. Attitudes toward these items greatly contributed to discrimination.

2) Important Attitude Items in Each Group

For Group G1, coefficients for Fischer's linear discriminant function for all 25 items as calculated in the previous section were larger than the overall mean. This indicates that all abilities required for programming were important to the students in Group G1. Thus, it is clear that the students in Group G1 were enthusiastic about programming in general.

The coefficients of five items (42, 44, 47, 48, and 52) in Group G2 were higher than the overall mean. This indicates that abilities such as writing and reading reports, understanding syntax, and expressing ideas on a computer were important to the students in Group G2 and that these abilities improved in this group.

The coefficients of all 24 items in Group G3 excluding "45. Ability to understand other people's ideas" were higher than the overall mean. This indicates that all abilities required for programming were important to the students in Group G3, similarly to the students in Group G1. Thus, it is clear that the students in Group G3 were also enthusiastic about programming in general.

No items in Group G4 had coefficients higher than the overall mean, indicating that there were no important items. Thus, it is clear that the students in Group G4 were not very enthusiastic about programming.

IV. Discussion

A. Helpful Activities for Improving Attitude

In Section A in Chapter III, which outlined helpful activities, activities in "1. Studying and asking friends questions" were reported as helpful in improving "IV. Attitudes toward working on problems." This first group of activities consisted of items such as "23. Doing research about what to include in programs to be created," "26. Creating programs based on research," and "30. Revising programs based on peer review." This signifies that activities such as doing research about assigned problems, asking the professor or TA about problems, and creating and revising programs were helpful for solving problems.

"2. Activities that involve studying with the textbook" were helpful in improving "I. Understanding of computers and ability to systematically set tasks." This second group of activities consisted of "07. Studying using the textbook," "21. Asking friends questions about program creation assignments," "03. Asking friends questions about lecture topics," "05. Preparing," "06. Reviewing," and "10. Studying using practice problems." This signifies that activities such as studying the textbook and practice problems, asking friends about assignments, preparing and reviewing, and deepening programming knowledge were helpful for improving students' ability to systematically set tasks.

"3. Activities that involve using applications as well as studying and evaluating other people's programs" were helpful in improving "III. Attitudes toward evaluating other people's work." This third group of activities consisted of items such as "14. Using Excel," "31. Revising programs while referencing other people's programs," and "12. Studying for the final exam." This signifies that activities such as using Excel or Word, revising and evaluating programs and specifications, and studying for the final exam were helpful in improving students' ability to evaluate and revise programs and reports.

"4. Listening to lectures" was helpful in improving "II. Attitudes toward programming techniques." Essentially, this means that listening to lectures was helpful in improving students' knowledge of algorithms, their knowledge of and interest in programming, their ability to express ideas with PHP, and their knowledge of correcting errors.

B. Classification of Students Based on Principal Component Analysis of Attitudes

The principal component scores calculated in Section D in Chapter III were used as variables in cluster analysis of attitude score improvement and the 24 students included in the analysis were categorized into four clusters (groups). In order to identify the characteristics of each group, the mean and standard deviation of attitude scores for programming abilities and general abilities were calculated as shown in TABLE VI.

Analysis with ANOVA showed that attitude scores for the four groups were significant ($F(599) = 207.9^{***}$, $p < .001$). Multiple comparison with the Bonferroni method revealed significant differences for all comparisons. Based on these analyzes, it was found that the mean scores for groups 1 through 4 (m1, m2, m3, m4) were related as follows: m1 > m3 > m2 > m4. Thus, there were differences in programming abilities between the four groups, and, as also shown in Figure 1, differences between general abilities and abilities required for programming. These results suggest that in order to improve students' abilities further, there is a need to consider adjusting teaching methods for each group.

Principal component 1, "General abilities required for programming," is taken on the horizontal axis. "Ability to write a report" is taken on the plus direction of the vertical axis. "Interest in and motivation for programming" is taken on the minus direction of the vertical axis. The student was able to be classified in four groups as shown in Figure 1. Particularly, it is visually recognized that four groups are classified by the principal component 1 with predominantly big eigenvalue from Figure 1. A rating level of attitude relating to the programming and the general attitude is in order of student group G1, G2, G3, and G4 as shown in Table 10. It is revealed that the four groups were different in attitude.

In addition, the difference in four groups is recognized by the result of the following discriminant analysis. Student group G1 and G2 are generally highly motivated to programming. Student group G3 is the group which "ability to write a report, grammar knowledge, and ability to realize their ideas in the computer" have improved. Student group G4 has not items with greater rating value than the average of the total and is not very ambitious.

TABLE VI. MEAN ATTITUDE SCORES FOR THE TWO TYPES OF ABILITY IN EACH GROUP

Group	No. of students	Attitude scores for programming abilities		Attitude scores for general abilities	
		m	SD	m	SD
1	4	4.1	1.4	3.5	1.7
2	7	2.1	1.0	1.2	1.0
3	9	0.6	2.0	−0.3	2.0
4	4	−1.8	1.5	−1.5	2.0
m		1.2	0.9	0.6	0.4

V. 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 activities useful for improving them were assessed with surveys conducted before and after the course. Attitudes and their activities useful were analyzed with Chi-square tests and residual analysis. Attitudes related to programming were analyzed with principal component analysis and discriminant analysis.

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

(1) The activity "1. Studying and asking friend questions" was helpful in improving "IV. Attitude toward working on problems." The activity "2. Activities that involve studying with the textbook" were helpful in improving "I. Understanding of computers and ability to systematically set tasks." The activity "3. Activities that involve using applications as well as studying and evaluating other people's programs" were helpful in improving "III. Attitudes toward evaluating other people's work." The activity "4. Listening to lectures" was helpful in improving "II. Attitudes toward programming techniques."

(2) The students were categorized into four groups using the principal component scores as variables. The following four groups were found: group G1 who had the highest mean scores for all items and their abilities related to programming improved the most overall; group G2 who had the second highest mean score of all four groups; group G3 who had lower mean score than the overall mean and did not improve very much; group G4 whose programming abilities declined the most overall after the course.

(3) The following was lead from important attitude items in each student group: group G1 and G2 were enthusiastic about programming in general; group G3 improved only some abilities such as writing and reading reports; group G4 were not very enthusiastic about programming.

(4) There were differences in the general abilities required for programming between the four student groups obtained from cluster analysis with principal component scores as the variables, indicating the need to consider ways to adjust teaching methods to fit each type of student.

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

ACKNOWLEDGMENT

The author 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. The author would like to express my appreciation to the students who were surveyed and who helped me collect educational information.

REFERENCES

[1] Anku, Illustrated Book of PHP, Shoeisha, Tokyo, Japan, 2011.

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

[3] I. Miyaji, and K. Yoshida, "The practice and learning effect of education by blending of lecture and e-learning," Transactions of Japanese Society for Information and Systems in Education, Vol.22, No.4, pp.230-239, 2005.

[4] I. Miyaji, K. Yoshida, and Y. Naruse, "The effects of blending e-learning and lectures utilizing a structured notebook," Transactions of Japanese Society for Information and Systems in Education, Vol.4, No.3, pp.208-215, 2007.

[5] I. Miyaji, "Effects on blended class which incorporates e-learning inside the classroom," E-learn2009, The 20th World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education, pp.1818-1826, Vancouver, Canada, 2009.

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

[7] I. Miyaji, "Comparison between effects in two blended classes which e-learning is used inside and outside classroom," US-China Education Review, USA, Vol.8, No.4, pp.468-481, 2011.

[8] I. Miyaji, "Consciousness and recognition degree of terms in programming through blended classes," Proceedings of Japanese Society for Science Education, Chugoku Branch Symposium "Toward Blended Learning from E-learning (part 5), pp.24-28, 2013.

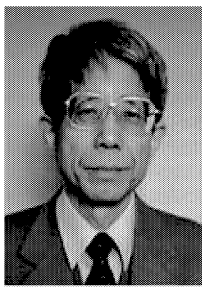
[9] I. Miyaji, and K. Yoshida, "Improvement of the attitudes and their familiarity with terminology of a programming course with a blended learning structure," Proceedings of the Canada International Conference on Education, CICE 2014, pp.301-307, Sydney, Canada, 2014.

[10] I. Miyaji and K. Yoshida, "Categories of attitude and student determined by cluster analysis of the attitudes toward programming abilities in a blended class," International Journal Cross-Disciplinary Subjects in Education (IJCDSE), Vol.5, Issue 4, pp.1845-1853, 2014.

[11] J. Shinkai, and I. Miyaji, "Effects of blended instruction on C programming education," Transactions of Japanese Society for Information and Systems in Education, Vol.28, No.2, pp.151-162, 2011.

[12] E. Takaoka, and W. Ishii, "Fully e-learning Java programming course: Design, development and assessment," Transactions of Japanese Society for Information and Systems in Education, Vol.25, No.2, pp.214-225, 2008.

AUTHOR'S PROFILE



Dr. Isao Miyaji received the B.S. and M.S. degrees from Okayama University in 1969 and 1971, and the Ph.D. in Engineering from Kyoto University in 1984. Since 1993 he has been a Professor with Faculty and Graduate School of Informatics, Okayama University of Science, Japan. He had a visiting Research Associate at Kyoto University at 1975, and at University of Minnesota and University of California Barkley, USA at 1981. His current interests are in blended learning and improvement of the instruction technique by using ICT. He received Best Paper Award from LICE 2013 and Achievement Award from Chugoku Branch of JSSE.