A Team Learning Approach to Motivate Students who are New Learners to Programming

Chikako Morimoto[1]

[1] Tokyo University of Science, Fujimi 1-11-2, Chiyoda-ku, Tokyo, 1020071, Japan
mrmt@rs.tus.ac.jp

Abstract. This study presents three approaches (first: team learning, second: difficulty adjustment, and third: pinch analysis) implemented to motivate students to learn in an IT system development class for students in the College of Business Administration. In particular, we supported to design a small team building. As a result, 80% of the students were anxious about programming before the class started, but after 14 weeks, the number of anxious students decreased to 30%. The number of students who found programming fun increased to 64%.

Keywords: Programing, Team Learning, Motivation.

1 Introduction

In recent years, information systems have become indispensable for social activities as the use of IT has advanced in various fields, including education and society. As a result, even universities specializing in fields other than Computer Science are increasingly offering classes on information system development and programming education using IT services.

In this study, we present three approaches (first, team learning; second, adjusting the level of difficulty; and third, pinch analysis) implemented to motivate students to learn in an IT systems development class for students in the School of Management.

The structure of this paper is as follows: in Chapter 2, we present a previous case study related to IT learning; in Chapter 3, we introduce the lesson content we worked on in this study; in Chapter 4, we describe our approach and its evaluation and the results. In Chapter 5, we summarize the conclusion.

2 Related Research and Practical Programing Education

It goes without saying that STEAM education has become increasingly important in recent years. Even before information systems became so important to society, there was a great need to train people in computer science. This is because being able to handle computers and program them is necessary to create new industries and advance society.

J. Caro et al. (eds.), *Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2023)*, Atlantis Highlights in Computer Sciences 20,
https://doi.org/10.2991/978-94-6463-388-7_14

In the UK, "Computing Education" has been offered to elementary school students since 2014. Before that, the curriculum was called "ICT". "ICT" was a curriculum that used computers as tools, but "Computing Education" has changed its content to develop creativity and thinking skills. It is reported that this education develops the four skills of Decomposition, Pattern Recognition, Abstraction, and Algorithms[1].

In Finland, programming has been a compulsory subject for elementary school students since 2016. This is based on the new National Core Curriculum, which aims at the continuous development of the population. It has been reported that this new curriculum has led to growth in logical thinking skills, particularly in mathematics [2].

In the U.S., programming education is also flourishing as former President Barack Obama has focused on promoting STEM education. For example, the non-profit organization Code.org offers free education, and the entire country is raising the bar for programming education.

In Japan, programming education in elementary schools became compulsory in 2020. Its content focuses on learning logical thinking rather than writing programs. In reality, there are a number of issues such as a shortage of teachers, teaching materials, and computing environments, and the spread of programming education has only just begun. In Japan, the shortage of IT personnel has been a social problem since around 2000. In particular, a shortage of software engineers has been pointed out, but the problem has not been resolved in more than 20 years. On the contrary, as new technologies are created one after another, the problem of shortage of software engineers is growing. Companies are also promoting the construction of low-code and no-code information systems, but this has not solved the problem. Therefore, an increasing number of universities in non-information fields are providing programming education. The number of practical courses that not only cultivate programming thinking but also involve actual programming is increasing.

3 The Problem and Our Approach

This chapter describes how we provided programming education to non-informatics students. First, we report the results of a survey of the characteristics of students who have difficulty with programming, and then we describe the teaching we conducted as a team approach.

3.1 Programming Education Issues

In general, students outside of informatics are not good at programming. They often have a sense of wanting to avoid it if possible. Okamoto classified first-time students' difficulties in learning programming into three patterns: (a) difficulty with complex tasks, (b) difficulty with complex concepts, and (c) losing sight of the objective because they do not understand the essence of the subject [3].

Based on this classification, we conducted an experiment with students who self-identified as poor at programming and observed how they stumbled. The subjects were four university students.

Prior to the experiment, we conducted a survey on why they felt uncomfortable with programming and received the following three responses.

(i) There is a lot of knowledge to learn.

(ii) It is difficult to maintain motivation.

(iii) Unable to be sure of the legitimacy of the programming.

In this experiment, the process of solving programming problems was observed by analyzing logs of computer operations.

Subjects were given two programming problems and asked to type the answers into the computer. If they did not understand something, they were allowed to search the Internet. The programming language was Java, which the four participants knew. The problem was a computation to find the product and quotient of integers. The experiment took place in October 2020, one participant at a time.

Figure 1 shows the active window of Student A, who answered correctly the fastest. Figure 2 shows the active window of Student D, who could not answer correctly.

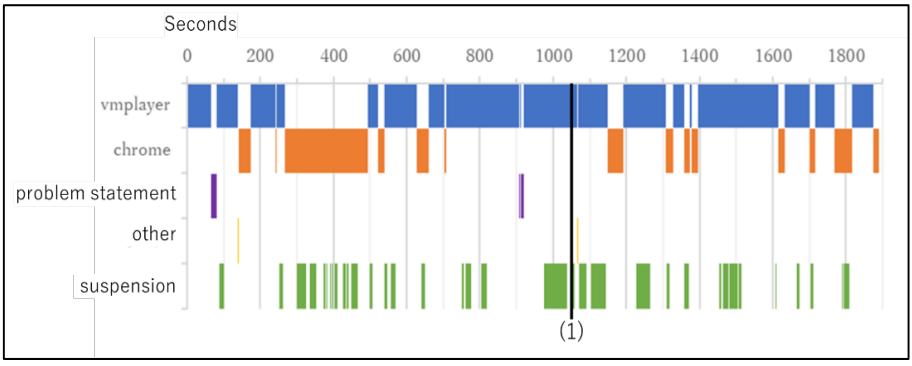

Fig. 1. Changes in Student A's active window

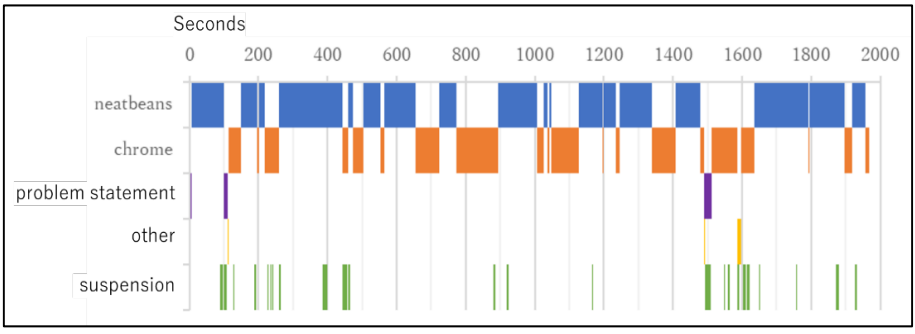

Fig. 2. Changes in Student D's active window

Student A answered the first question in (1). He also checked the question carefully at the beginning and answered it correctly. There was no duplication of search keywords, and he knew what he needed to look up. On the other hand, Student D had many

duplicate search words and did not know what to look up. It also took time for him to understand the error text. He did not use the error support function of the development tool.

From the results of this experiment, it can be concluded that Student A did not fit any of the stumbling blocks and been motivated problems, whereas Student D fit all of a, b, and c.

It is thought that encouraging each other with peers is effective in dealing with motivation problems. In addition, giving advice to a, b, and c at appropriate times is effective. Therefore, we decided to introduce a team approach to programming learning.

3.2 Our Team Approach

Many studies have shown that pair programming is effective for learning programming [4]. Pair programming has also become a necessary practice in the agile development process.

The benefits of pair programming include improved quality through mutual checking, reduced review costs, and the possibility of improving the development process[6].

However, there are some points that should be noted when introducing this practice for programming learning purposes. If there is a large difference in knowledge between pairs, it becomes a one-way action, and the person teaching cannot feel growth; if the knowledge between pairs is low, learning costs increase; and the Ringelmann effect may cause slacking off. Therefore, we decided to implement the following approach.

Team learning

- Create teams instead of pairs
- Teams of up to 4 people
- Teams should consist of people who know each other

Difficulty adjustment and pinch analysis

- Each team is free to set the level of difficulty from 3 levels.
- Each person writes a weekly reflection in class.
- TAs will analyze the pinches written in the reflections and provide support.
- Pinch points in the development environment are published as FAQs to the class.

The difficulty levels are as follows.
Low: HTML and CSS only
Medium: HTML, CSS, JavaScript
High: HTM, CSS, JavaScript, PHP

Which one they choose will not affect their grade. What matters is whether the team was able to achieve the chosen level. Of course, students were encouraged to try higher levels. We varied the content on the website little by little and provided an interesting way to challenge them by making them understand the UX/UI as well as just creating

a website. Furthermore, we thought that by checking the weekly reflections, we could detect any slacking off.

These approaches were implemented in a 14-week programming class offered in the fall semester from September to December 2022.The students were 176 second-year students in the College of Business Administration who had taken Basic Python in the spring semester of their freshman year. There was one faculty member and two teaching assistants (TAs). In a survey before the class began, 80% of the students responded that they were anxious about programming. In addition, 62% of students stated that they had forgotten Python. However, 26% of students said they wanted to do their best and were looking forward to it. The class uses HTML, CSS, and JavaScript to create web pages, and students taking the advanced challenge use PHP to create dynamic websites. 15.5% of the students had written HTML and 8.6% had created web pages. Students formed teams of 3-5 students, for a total of 48 PBL teams. Initially, 19 teams declared that they would challenge themselves at a higher level. However, 12 of the 19 teams did not include the member who created the web page and subsequently experienced difficulties.

4 Results and Evaluation

4.1 Classroom Results

The class resulted in the grades shown in Table 1. Grading will be based on two criteria: first, mastery of a minimum of HTML, CSS, and JavaScript; second, achievement of the learning objectives set by the team. Almost all of the students received credit for the course because of the supplementary examinations given to students whose grades were not satisfactory.

Table 1. Grade Distribution.

Grade	num
S	10
A	74
B	78
C	14
D	1
sum	176

During the first two weeks of the 14-week course, lectures on digital transformation were given to help students understand the relevance of IT systems to social life and to provide a background for learning, thereby stimulating motivation for the entire class.

The next three weeks were spent learning about basic web development programming. Students were challenged with personal programming assignments. In doing so, they were instructed to help each other until all team members submitted their assignments.

The following weeks, students learned about UX/UI and worked on PBL by designing a website they wanted to develop as a team.

After 14 weeks, all 48 teams had completed their challenge websites. Six teams took on the challenge at the high level, and the level of completion was comparable to those created by information science students. However, few teams were able to fully utilize the development environment, for example, only two teams used Github. In this respect, they are inferior to information science students who have a variety of programming experience. However, the results of the students of the Faculty of Business Administration appear to have been highly complete. Figure 3 shows students presenting their results.

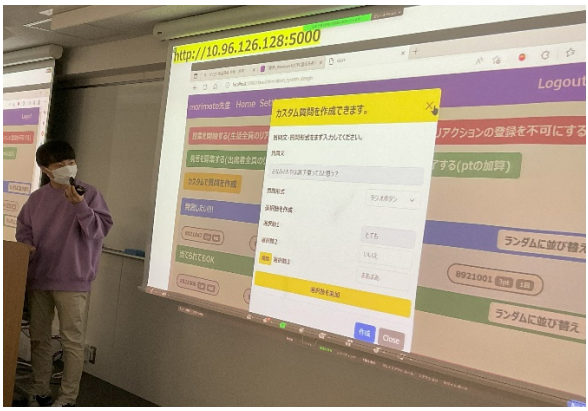

Fig. 3. Student Presentation

In a post-class survey, the number of students who felt anxious about programming decreased to 30%. Additionally, the number of students who answered that programming is fun increased to 64%. Our approach appears to be effective.

4.2 Evaluation

In this chapter, we evaluate our approach. Regarding Team Learning, we were able to maintain high motivation by choosing our own teams. We were also able to help each other despite differences in knowledge levels. Table 2 shows the results of the "Team Satisfaction" section of the student questionnaire. 79.1% of the students were satisfied with 8 or above, indicating that many students were satisfied with their teams.

Regarding Difficulty Adjustment, we were able to decide the level ourselves, which increased our independence. Since we were able to complete the team's work step by step, we were able to create an atmosphere in which we wanted to challenge ourselves to a higher level. This is the same result as in previous studies in which the team's right to self-determination contributed to increased motivation[5]. Although some teams lowered their level, they did not lower their level of completion, so they were able to approach programming in a positive manner. Regarding Pinch Analysis, each class was able to start smoothly by having the TAs identify areas where each team was having

trouble and compile them in a FAQ before class. In particular, by providing features and tips that you should know about the development environment, development has become smoother.

Table 2. Team Satisfaction.

Team Satisfaction	
10	35.6%
9	24.5%
8	19.0%
7	15.3%
6	2.5%
5	3.1%
4	0.0%
3	0.0%
2	0.0%
1	0.0%
sum	100.0%

However, there was no time in class to teach management methods such as configuration management version management, and software testing. The entire curriculum should have been designed not only for learning programming but also for the software development process.

5 Conclusion

In this paper, we reported the results of implementing a team learning approach to help programming beginners and students who are not good at programming fall in love with programming. We confirmed that it was effective to use team learning instead of individual learning, to allow students to choose their own learning level step by step, and to have TAs eliminate as many obstacles as possible such as the environment.

In the future, I would like to further explore ways to encourage non-informatics students to engage in more advanced programming efficiently and positively. We would also like to further examine the relationship between team performance and individual learning.

Acknowledgments

The authors would like to thank Akemi Yamano for her help in collecting experimental data for programming; Yutaro Takaya and Ryo Tateishi for their role as TAs in the PBL.

References

1. The Royal Society: Shut down or restart? The way forward for computing in UK schools. The Royal Academy of Engineering (2012).
2. Ikeno Masaharu.: The Current Status of the Education Reform Under Full Implementation of New NCC in Finland –Discussion on Phenomenon-based Learning and Programming Learning–. Journal of Takasaki City University of Economics, vol. 60, no.1,pp.1-25, (2017).
3. Okamoto Masako.: The First Learning of Computer Programming and Missteps. Journal of Information Processing Society of Japan, vol. 56, no. 6, pp. 580-583.(2015)
4. L. Williams; R.R. Kessler; W. Cunningham; R. Jeffries.: Strengthening the case for pair programming. IEEE Software, vol. 17, no. 4, pp. 19-25,(2000).
5. Yukiko Enokida, Tohru Matsuodani; A Development of Team-Building Skills, Journal of the Society of Project Management, Vol. 6, No.2.(2004).
6. Awad A Younis; Rajshekhar Sunderraman; Mike Metzler; Anu G. Bourgeois; Case Study: Using Project Based Learning to Develop Parallel Programing and Soft Skills, 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), DOI: 10.1109/IPDPSW.2019.00059 . (2019)