Analysis of Programming Behavior

筑波大学

図書館情報メディア研究科

2013 年 3 月

胡 君珊

Table of Contents

Chapter 1

Introduction

1.1. Background

Computer programming has been considered as a difficult task for decades. It is not only for the process of developing code of programmer, but also it is more like a process of innovation. During this process, a program would be designed, the code would be written, tested, debugged, and then maintained. When a programmer is developing a program, he should begin to write code as soon as even a trace of fragmented ideas appeared in his mind. After finishing the code, the programmer might suddenly found it could not work. The only way for the programmer at that moment is to abandon the written code and restart a new idea to develop the program. From the idea coming up to the code abandonment, it usually costs a lot of time and leads the programmer to a "blind end". To improve the effectiveness and efficiency of programming, collaborative programming came into being. This definition was first mentioned by Larry Constantine [1] in 1995, that pairs of programmers produced code and developed program faster and freer of bugs.

As one major form of collaborative programming, pair programming was originated in industry as a key component of the eXtreme Programming (XP) development methodology [2], which was created by Kent Beck to improve software quality and responsiveness to change customer requirements. As the name suggests, pair programming is conducted by 2 persons who work on one machine with one set of computer equipments, including one display, one keyboard and one mouse. The

programmer who does the keyboard controlling and mouse handling is considered as "Driver"; while another one, who is responsible for observing the code input, giving suggestions, contributing to the programming verbally, is called "Navigator". As the XP software development methodology had been universally adopted and practiced, pair programming starts to be accepted in more and more fields because of the higher code quality created and less time spent compared with solo programming [3][4][5][6]. Furthermore, it could improve programmers' programming experience and their cooperative consciousness [6][7]. The programmers' behavior plays a key role in the performance of pair programming [8][9][10][11], the cooperative work between the pair has an immediate influence on the programming result and experience [12][13][14].

However, with a better cooperative work, even the pair programmers would outperform, problem would still be encountered. The problem-solving not going smoothly might lead to the programmers' motivation decreased in the commercial industry, or would result in the students' negative emotions to study.

1.2. Purpose

In this study, we focused on the programming behavior and conducted pair programming in an introductory programming course. We kept an eye on the behavior and assumed one hypothesis behavior pattern in pair programming, and compared them between the Success and Failure cases.

We are aiming at analyzing the behavior and the behavior patterns in pair programming, which might be the factors that affect the programmers' performance and the programming result. The further goal of our study is to learn symptoms to indicate the pair programming status from the analysis. The results and findings are expected to be available to expand the collaborative programming study in Computer Supported Collaborative Learning (CSCL). It is expected that this study could help to sense the learning status of the pair and intervene in the pair programming learning.

1.3. Paper Structure

In this thesis, including the Introduction Chapter, there are 8 chapters in total. Chapter 2 is about the previous related researches. In Chapter 3, we described that what kind of programming behavior data we need and how we collected the data for the analysis. In Chapter 4 the data we collected was classified and listed as tables, and the methodology we would use for analysis was illustrated. And the analysis results were expounded in Chapter 5. In Chapter 6 we discussed the results we got from the analysis, and surmised the reason. In Chapter 7 future plan of our study was directed. In the last chapter, Chapter 8, we made the conclusion of the thesis and our study.

Chapter 2

Related Work

In this chapter, we presented some previous researches related to programming.

2.1. Solo vs. Pair programming

In the previous researches which focused on introductory programming courses, it has been proved that pair programming is more outperformed than solo programming.

Nosek recorded the programming process, and according to the comparison he found that pair teams usually developed the program and software with higher quality [3]. Additionally, he found that collaboration improved the problem-solving process, and that might be why pair teams performed better.

In Williams, Kessler, Cunningham, and Jeffries' paper, experiment was conducted in a course: students were divided into solo programmers and pair programmers, then their programming process were recorded [4]. According to the comparison of solo and pair programming, it was reported that through pair programming, software and programs can be produced in less time, with code quality rather better.

McDowell, Werner, Bullock and Fernald's findings reported in his paper were part of a larger study funded by a foundation to assess the effectiveness of pair programming on the performance [5]. They examined the data and suggested that programmers who worked in pairs produced better programs. Furthermore, they performed significantly better on the final exam, compared to students required to program individually.

Nagappan, Williams, et al., they observed and codified many paired and solo lab sections and found that student pair programmers were more self-sufficient, generally perform better on projects and exams [7].

These researches above have shown the efficiency of pair programming, without mentioning anything about the process and the results of pair programming. Does the pair meet any problems while programming? If so, whether the problem-solving go smoothly or not? The programming process and behavior were not analyzed in these researches. In our study, pair programming is the point we focused because the behavior and cooperative work in it are worth more than that in solo programming. The behavior is analyzed, and the comparison of successful and failed cases is done in this study.

2.2. Behavior analysis

Behavior in pair programming has been paid increasing attention in more researches, most of which keeps an eye on the communication; some conveniently observed other behavior as keyboarding using, gestures, etc.

Sfetsos, Stamelos, Angelis and Deligiannis conducted controlled experiments to investigate the behavior in pair programming [8]. According to the observation of the programming process and the questionnaires answered by the students, the results in their research showed that productivity for pairs is positively correlated with communication transactions.

After observing the record of pair programming, Bryant and Romeo, Boulay got the results that the expertise distribution would influence the pair communication interaction [9]. They also noticed that the operation behavior was assisting intra-pair verbal communication. And some other behavior or factors, such as gestures, writing a list, would more or less affect the pair programming.

According to the ethnographic observation, in Chong and Hurlbutt's research about behavior in pair programming, they presented that distribution of expertise among a pair had a strong influence on the tenor of pair programming, and keyboard control had a consistent secondary effect on decision making with the pair [10].

Hirai and Inoue's research of conversation in pair programming is the senior research of our study here [11]. They compared the utterance in Success and Failure cases, and the insights, that successful case had longer speech length, more numbers of repeating explanations and more numbers of continuous speeches, would be available to identify the collaborative work and the programming status in pair programming.

These works analyzed the behavior, especially the communication, in pair

programming. Some also presented their findings about other behaviors as keyboard control, gestures, but concluded just according to the observation. In our study, we analyzed the behavior all based on data analysis. With the objective data, we analyzed the utterance and operation in pair programming, and compared those in Success and Failure cases. We also paid attention to the cooperative pattern in pair programming.

2.3. Cooperative work in different domains

Cooperative work is always regarded as the key component in group work. Many previous works researched on it in many domains, including the software development field.

In the educational programming field, Lory and Mike analyzed students' cooperative work in a program course, when they were doing the mystery program readings, program solution sharing and analyzing, and some other activities [12]. From the questionnaires finished by students, it was presented that students gave positive feedback to a set of cooperative group activities. The cooperative activities made students work in high efficiency.

And Edward, Katherine, Keith, John also observed the cooperative work in a course [13]. The cooperative work in the course, like exercises as think-pair-share, group work activities as discuss and observe, learning activities as group question and role play, could increase retention and boost the performance of at-risk students.

Duo Wei conducted the survey for students and used the cooperative learning method in pair programming, required students to work together to finish the given task [14]. According to his finding, cooperative learning method was perceived to be effective in teaching programming classes.

In the domain of work, Gary and Cheryl, Severin recorded the communication of working partners and analyzed the cooperation among them [15]. The cooperative patterns found in this research provided insight into what aspects of groupware were perceived as helpful to users' cooperative work. And Dacid, Mark, Ian also concerned about the cooperative patterns in working environment [16], and according to their observation, cooperative patterns was found to be helpful in framing the understanding

of phenomena in a new setting, generate design concepts and issues, and envisaging the potential design solutions.

Cooperation of Game-Play is researched by Anastasiia, Peter, et al. [17]. Players' patterns in remote game play were analyzed comparatively. They suggested that with communication, remote players have higher level of collaboration and enjoyment.

In health care field, Claus, Lotte, and Flemming researched the cooperative work of medical secretaries [18]. They focused on four professions: physicians, nurses, physiotherapists, and medical secretaries. After combining the interviews, observation and survey, they suggested that medical secretaries' work was not mere routine, but requiring skill and applying knowledge. With the mandatory knowledge, medical secretaries cooperated with other professions, acting as intermediaries of relatives, patients, and staff.

As we know from the previous researches, cooperation is an important factor affecting the efficiency of the group work. In this study, we assumed one hypothesis concerning with behavior pattern in pair programming, which was expected to be available for identifying the programmers' cooperation.

Chapter 3

Data Collection

For the analysis of the behavior in pair programming, the data was collected in the previous research of our lab, from one introductory programming course in University of Tsukuba, named “Programming I”, in which C language was taken as the major teaching content. This course is held for the freshmen in School of Informatics, University of Tsukuba. It aimed at letting the students understand what C language is, how to write code in C language, and know the basic knowledge of compiling a program and developing software. In this study, the data we used was collected by the senior.

In the previous research done by Hirai, et al. [11], the pair programming practice experiment was conducted in “Programming I” course at University of Tsukuba. They collected the pair programming data in the course of 2010 and 2011, and used some of 2010 pair programming data for analysis in the previous work. Here in this study we used the pair programming data of 2011 “Programming I” course.

The data of pair programming practice experiment used in this study was taken from 2011 “Programming I” course by Hirai et al. Each lecture of the course lasts for 75 minutes, and pair programming practice session is regarded as a part of the lecture. Totally 8 pair programming practice sessions were conducted, and in each session, 4 pairs of programmers’ programming procedures were recorded by cameras; excluding the first session, which was taken as a trial session, only 3 pair programming practices were recorded in it. The total amount of pair programming practices recorded is 31.

Before recording the pair programming practice, some preparations were done by the

experimenter such as the pair combination, the role deciding in each pair (who is the navigator and who is the driver), the cameras setting up, etc. As soon as the pair combination was decided, the roles of driver and navigator could not be exchanged. Here 3 cameras were set up for each pair of programmers to collect as many aspects in pair programming as possible.

Figure 1. Setup of the cameras for data collection

Figure 1 is a screenshot of the practice session in the "Programming I" course. We can see that three cameras were set up in one session; they recorded the pair programming from 3 different angles, for Driver & Navigator (front), for Driver, Navigator & Desk (desk), and for Display (display).

The three cameras are used for collecting pair programming data , the front one is for recording the pair's communication, the desk one is for recording the pair's behavior and activities during pair programming such as typing, using mouse, pointing at the display, referring to the textbook, and some other behaviors; and the other is for recording display. Figure 2 shows a scene from the practice session from the 3 angles

taken by the cameras. To protect the students' privacy, we covered their faces with Mosaic.

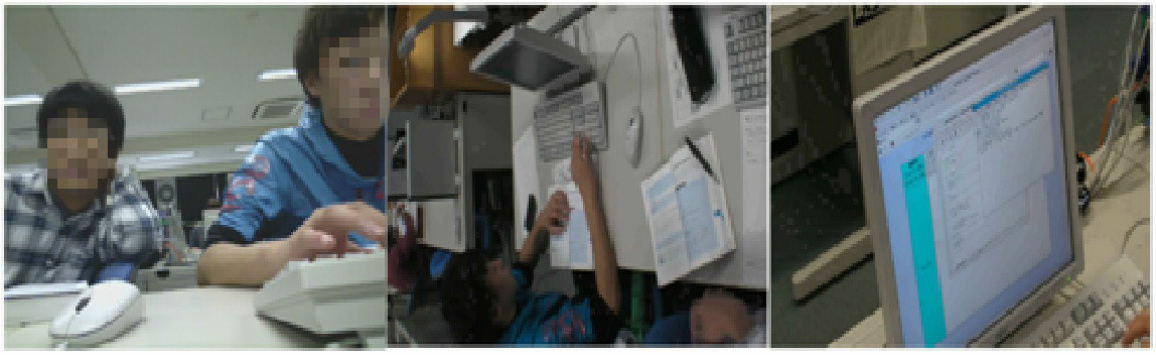

Figure 2. Scene from the practice session

While programming together, the pairs are required to follow the instructions:

- The time limit for the assignment is 30 minutes. Code should be submitted even it is failed or unfinished within the 30 minutes.
- Driver is the only one who can operate the keyboard and mouse. The navigator could only observe and support the work of the driver without touching the mouse or keyboard.
- The assignment should be finished as soon as possible. It ends when the program is executed and a correct answer to the assignment is obtained.
- Driver and navigator could search in the textbook but not be allowed to use the Internet.
- The teacher or the teaching assistants are only available for equipment consulting. They do not accept any questions concerning the assignment while pair programming practice.

- The pair could add pertinent comment to make the program easy to understand as they like.

Chapter 4

Analysis

In this chapter, data we collected and how the data be analyzed was presented.

4.1. Data Processing

During the pair programming, most pairs would encounter different programming problems and then solve them successfully, or not. We consider each problem encountered as one case, in one pair's practice, they would have none, one or more cases. Every case gets successful or failed result at last. In this case, we have exact definition for these successful and failed cases.

A "Case" should be the problem solving process, beginning from a problem encountered and end with it being solved or time up. And a problem is a compilation error that occurs when learners compile their program, or a runtime error that occurs including whose result does not meet the students' expectation. "Success" is that problem being solved by the pair within the given limited 30 minutes. "Failure" is that problem not being solved in the end. Both "Success" and "Failure" cases are just the results of cases in the practice session.

As mentioned above, totally 31 pairs of pair programming practice were recorded in the "Programming I" course. Three pairs among them encountered no problems at all; the programming went smoothly till the end without any case. As to the other 28 pairs, each included at least one case inside, several included 2 or 3 cases. Among the 28 pairs' programming data, there were 36 cases, and according to our definition of "Success" and "Failure" cases, 23 were "Success" and 13 were "Failure".

We recorded each pair from different angles by using three cameras, so actually we

have three videos for one pair: front video, desk video and display video. In this study we use ELAN (EDUICO Linguistic Annotator) [19][20], a tool for the creation of annotations on video and audio resources, to synchronize the three videos into one integrated video, and then to tag and annotate the behaviors in the integrated one. Figure 3 is the screenshot of the video tagging and annotation with ELAN. The videos are shown on the top of the ELAN interface, and at the bottom the tiers and annotations could be added. The tier and annotation information of each pair programming practice are then output for further analysis.

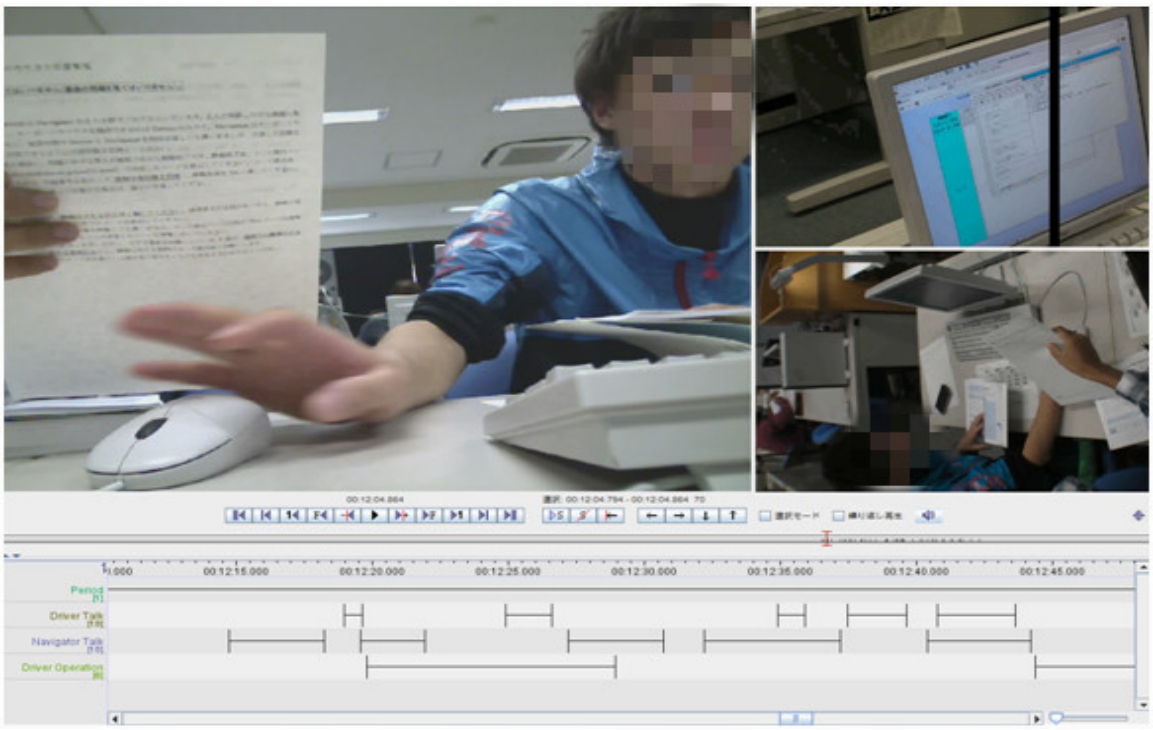

Figure 3. Screenshot of ELAN annotation interface

4.2. Parameter

The Parameter we used for data analysis would be described here.

4.2.1. Utterance data

In this study, we analyzed the utterance behavior in pair programming from three views, “Utterance ratio”, “Utterance frequency”, and “Average utterance length”.

4.2.1.1 Utterance ratio

Utterance ratio is about that “what percentage of the entire case is programmer’s utterance time”. The utterance length divided by data length is the result of pairs’ utterance ratio, since the utterance comes from the two programmers of pair programming. To get each programmer’s utterance ratio, it should be divided by two. The result is shown in percentage.

$$\bullet \text{utterance ratio} = \frac{\frac{\text{utterance length}}{\text{data length}}}{2}$$

With this equation, we analyzed the utterance ratio of each case and listed them in the Table I. The mean utterance ratio value of Success and Failure cases are calculated and compared.

4.2.1.2 Utterance frequency

Utterance frequency is the identifier of showing “how many utterance numbers there are in one minute”. “Minute” is used as the time unit, so the data length in the table is converted to minute for analysis. Programmers’ utterance frequency is calculated from utterance numbers divided by data length (min). To calculate each programmer’s utterance frequency, this result should be divided by two.

$$\bullet \text{utterance frequency} = \frac{\frac{\text{utterance numbers}}{\text{data length}/60}}{2}$$

With this equation, we analyzed the utterance frequency of each case and listed them in the table. The mean utterance frequency value of Success and Failure cases are calculated and compared

4.2.1.3 Average utterance length

Average utterance length is the identifier of showing that "how much time (in second) each utterance lasts". It is calculated from utterance length divided by utterance numbers.

- $\text{average utterance length} = \frac{\text{utterance length}}{\text{utterance numbers}}$

We calculated average utterance length of each case and listed them in the table. The mean value of this of Success and Failure cases are calculated and compared

4.2.2. Operation data

We analyzed the operation behavior in pair programming in the similar way as utterance analysis, from three views, "Operation ratio", "Operation frequency", and "Average Operation Length".

4.2.2.1 Operation ratio

Operation ratio is about that "what percentage of the entire case is the Driver's operation time". Operation ratio is calculated from that operation length divided by data length.

- $\text{operation ratio} = \frac{\text{operation length}}{\text{data length}}$

The operation ratio of each case was calculated and listed in the operation data of 2011 pair programming table. The mean operation ratio of Success and Failure cases

are calculated and compared

4.2.2.2 Operation frequency

Operation frequency is the identifier of showing that "how many operation numbers there are in one minute". Same as the analysis of utterance frequency, the data length in the table is converted to minute for analysis. Operation frequency is calculated from operation numbers divided by data length (min).

- operation frequency $= \frac{\text{operation numbers}}{\text{data length}/60}$

We use this equation to calculate the operation frequency of each case and listed the result in the table as the Operation frequency column. The mean operation frequency value of Success and Failure cases are calculated and compared

4.2.2.3 Average operation length

Average operation length is the identifier of showing that "how much time (in second) each operation lasts". It is calculated from that operation length divided by operation numbers.

- average operation length $= \frac{\text{operation length}}{\text{operation numbers}}$

We calculated average operation length of each case and listed them in the table. The mean value of average operation length of Success and Failure cases are calculated and compared

4.2.3. Operation after Driver and Navigator's dialogue data

Here we assume one hypothesis about the utterance& operation pattern, that

"Success case has higher ratio and frequency of 'operation after Driver and

Navigator's dialogue'."

The previous work suggested that the cooperation in pair programming had a significant impact on the performance, but was not focusing on or analyzing it in detail. We supposed that there would be cooperation pattern in pair programming, which could lead to successful problem-solving. As utterance and operation are the basic behavior in pair programming, we expect there would be correlation between utterance and operation, and this correlation is supposed to show the cooperation of the pair.

There is no doubt that conversation would appear between the pair, and the turn-taking utterance might be the opinion exchange between driver and navigator. With the opinion exchanging, a higher quality decision which was agreed by both driver and navigator would be made and then executed by driver. However, all these are just our assumption and needed to be tested.

We define the "operation after Driver and Navigator's dialogue" exactly. Shown as the Figure 4, if the last two utterances before Driver's operation are the turn-taking utterances spoke by both Driver and navigator, it would be regarded as match with our definition of "operation after (Driver and Navigator's) dialogue". This kind of dialogue must be at least one pair of turn-taking utterances.

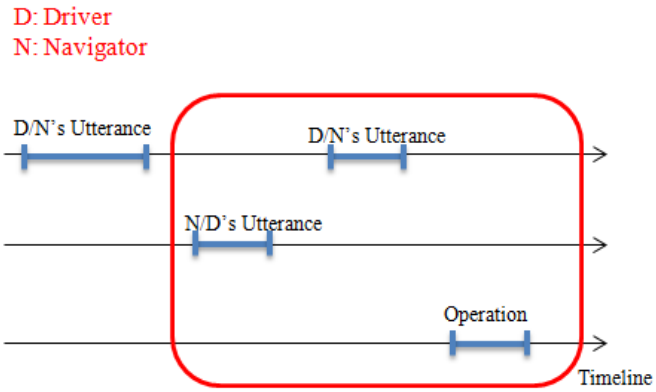

Figure 4. Operation after Driver and Navigator's dialogue

If this hypothesis was true, new clearer symptoms of patterns in pair programming to indicate the status of the programming could be obtained.

4.2.3.1 Ratio of operation after Driver and Navigator's dialogue

Ratio of operation after (Driver and Navigator's) dialogue represents that "what percentage of the operation numbers is the 'operation after dialogue'." It is calculated from that the number of operation after dialogue divided by total operation numbers.

- ratio of operation after dialogue $= \frac{\text{number of operation after dialogue}}{\text{operation numbers}}$

The ratio of operation after dialogue of each case was calculated by following the equation and the results are in percentage, and then listed in the table. The mean ratio of operation after dialogue of Success and Failure cases are then calculated

4.2.3.2 Frequency of operation after Driver and Navigator's dialogue

Frequency of operation after dialogue is the identifier of showing that "how many numbers of operations after (Driver and Navigator's) dialogue there are in one minute". The data length here is also converted to minute for analysis. The frequency of

operation after dialogue is calculated from that the number of operation after dialogue divided by data length (min).

- frequency of operation after dialogue $= \frac{\text{number of operation after dialogue}}{\text{data length}/60}$

With this equation, frequency of operation after dialogue in each case was analyzed and then the result is listed in the table. The mean frequency of operation after dialogue of Success and Failure cases are calculated and compared.

Chapter 5

Results

In this study we used Mann-Whitney U test for assessing if there was significant difference between Success and Failure cases, because of U test's applicability for arbitrary sample sizes.

5.1. Result of Utterance data

We analyzed the utterance in pair programming from "Utterance ratio", "Utterance frequency", and "Average utterance length".

Table I shows the pairs' utterance data of the 36 cases in 2011 pair programming, which was output by ELAN. The utterance contains Driver's talking and Navigator's talking.

The "data length" is counted from the happening to the solution of the problem in "Success" case. In "Failure" case, the ending of the data is the timing that the pair stopped solving the problem. The Driver and Navigator's utterance numbers are counted and listed separately in the table. The utterance length is the sum of Driver's utterance length and Navigator's utterance length. An utterance is the identifier of the programmer's speaking something, no matter whether he/she is talking to his/her partner or to himself/herself. It could be a sentence or just meaningless word as "Ah!", "Eh……", "Mm……", and some other mood words.

Table I. Utterance data of 2011 pair programming

case	data length (s)	Driver utterance numbers	Navigator utterance numbers	Utterance length (s)	Utterance ratio (%)	Utterance frequency (numbers/min)	Average utterance length (sec/number)
success 1	52	6	4	14.8	14.3	5.8	1.48
success 2	210	11	29	67.5	16.1	5.7	1.69
success 3	72	5	8	34.8	24.2	5.4	2.68
success 4	60	8	11	30.8	25.6	9.5	1.62
success 5	404	50	24	97.9	12.1	5.5	1.32
success 6	61	3	2	4.9	4.0	2.5	0.98
success 7	219	18	25	65.1	14.9	5.9	1.51
success 8	403	45	62	205	25.4	8.0	1.92
success 9	98	12	14	44.5	22.8	8.0	1.71
success 10	207	13	14	39.8	9.6	3.9	1.47
success 11	281	40	9	93.0	16.5	5.2	1.90
success 12	377	32	26	140.8	18.7	4.6	2.43
success 13	109	19	12	40.0	18.3	8.5	1.29
success 14	154	21	16	41.2	13.4	7.2	1.11
success 15	284	44	52	184.9	32.5	10.1	1.92
success 16	301	47	48	171.1	28.4	9.5	1.80
success 17	309	22	21	83.6	13.5	4.2	1.94
success 18	166	8	7	41.3	12.4	2.7	2.75

success 19	138	19	15	85.7	31.1	7.4	2.52
success 20	228	15	6	45.2	9.9	2.8	2.15
success 21	158	3	6	22.0	7.0	1.7	2.44
success 22	207	15	16	43.2	10.5	4.5	1.39
success 23	116	3	9	18.9	8.1	3.1	1.57
failure 1	774	37	130	449.2	29.0	6.5	2.69
failure 2	173	10	18	38.0	11.0	4.9	1.36
failure 3	452	27	62	117.5	13.0	5.9	1.32
failure 4	456	83	16	268.3	29.4	6.5	2.71
failure 5	348	24	20	74.1	10.6	3.8	1.68
failure 6	401	16	26	82.9	10.3	3.1	1.97
failure 7	599	30	36	172.8	14.4	3.3	2.62
failure 8	587	33	63	270.1	23.0	4.9	2.81
failure 9	286	36	35	109.8	19.2	7.5	1.55
failure 10	373	33	17	170.5	22.8	4.0	3.41
failure 11	445	18	68	182.9	20.6	5.8	2.13
failure 12	502	17	7	130.6	13.0	1.4	5.44
failure 13	395	9	28	135.0	17.1	2.8	3.65

5.1.1. Utterance ratio

The mean utterance ratio of Success cases is 16.9%, and of failure cases it is 18.0%.

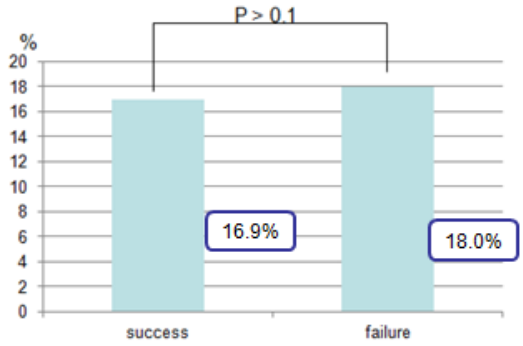

Figure 5. Utterance ratio

Figure 5 shows the mean utterance ratio of the two sets. With Mann-Whitney U test, $p > 0.1$ ($p = 0.29$), there is no significant difference between Success and Failure cases. We cannot say that Success case is with higher utterance ratio; even it has a higher mean value than Failure case.

5.1.2. Utterance frequency

The mean utterance frequency of Success cases is 5.73 numbers in one minute, and of failure cases it is 4.65 numbers in one minute.

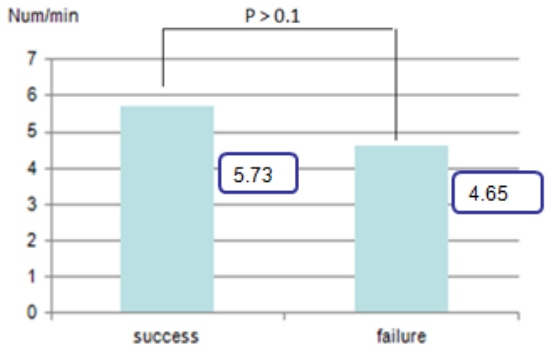

Figure 6. Utterance frequency

Figure 6 shows the mean utterance frequency of Success and Failure. With Mann-Whitney U test, $p > 0.1$ ($p = 0.29$), there is no significant difference between Success and Failure cases. We cannot say that Success case is with higher utterance frequency; even it has a higher mean value than Failure case.

5.1.3. Average utterance length

For Success cases, each utterance lasts for 1.81 seconds averagely, while for Failure each utterance lasts for 2.56 seconds. The comparison result was shown in Figure 7.

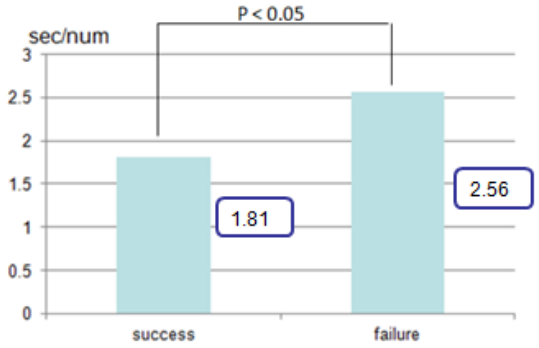

Figure 7. Average utterance length

With Mann-Whitney U test, $p < 0.05$ ($p = 0.03$), the difference between Success and Failure is marginally significant. Success case has shorter average utterance length than Failure case. As a result, each utterance lasts for shorter time in Success case. And this significant result was already obtained by the senior research, which was about the 2010 pair programming analysis.

5.2. Result of operation data

We analyzed the operation behavior in pair programming in the similar way as utterance analysis, from the three views, “Operation ratio”, “Operation frequency”, and “Average Operation length”.

Table II shows the pairs’ operation data of the 36 cases output by ELAN in 2011 pair programming.

Usually we think that operation should be considered as inputting content like code or comment by the keyboard. Actually mouse handling should also be regarded as part of operation. Programmer does the selecting, copying and pasting by using mouse. So in this study, we define operation as both keyboard controlling and mouse handling, and driver is the only one that could operate the input devices.

The “data length” definition is the same as that in utterance analysis, counted from the problem’s happening to the problem’s been solved, or till time up.

Table II. Operation data of 2011 pair programming

case	data length (s)	operation numbers	operation length (s)	Operation ratio (%)	Operation frequency (numbers/min)	Average operation length (sec/number)
success 1	52	5	10	19.3	5.8	2
success 2	210	11	61.1	29.1	3.1	5.55
success 3	72	2	11.2	15.6	1.7	5.6
success 4	60	4	17.4	28.9	4.0	4.35
success 5	404	24	134.8	33.4	3.6	5.62
success 6	61	4	30.5	49.8	3.9	7.62
success 7	219	8	124.7	57.0	2.2	15.59
success 8	403	20	180.4	44.8	3.0	9.02
success 9	98	3	26.7	27.4	1.8	8.92
success 10	207	8	34.0	16.4	2.3	4.25
success 11	281	14	50.5	17.9	3.0	3.60
success 12	377	22	93.7	24.8	3.5	4.26
success 13	109	9	25.7	23.6	5.0	2.86
success 14	154	8	73.6	47.9	3.1	9.20
success 15	284	20	109.6	38.6	4.2	5.48
success 16	301	14	77.0	25.6	2.8	5.50
success 17	309	12	147.2	47.6	2.3	12.27
success 18	166	4	35.5	21.4	1.4	8.87
success 19	138	8	63.8	46.2	3.5	7.97

success 20	228	8	122.9	54.0	2.1	15.36
success 21	158	5	102.5	65.1	1.9	20.50
success 22	207	9	48.8	23.6	2.6	5.42
success 23	116	5	9.0	7.8	2.6	1.81
failure 1	774	48	161.7	20.9	3.7	3.37
failure 2	173	14	36.9	21.3	4.9	2.63
failure 3	452	31	154.4	34.2	4.1	4.98
failure 4	456	22	142.3	31.2	2.9	6.47
failure 5	348	24	91.1	26.2	4.1	3.80
failure 6	401	11	79.7	19.8	1.6	7.24
failure 7	599	20	132.4	22.1	2.0	6.62
failure 8	587	31	149.1	25.4	3.2	4.81
failure 9	286	6	16.6	5.8	1.3	2.77
failure 10	373	12	66.5	17.8	1.9	5.54
failure 11	445	33	142.7	32.1	4.4	4.33
failure 12	502	32	110.6	22.1	3.8	3.46
failure 13	395	12	26.4	6.7	1.8	2.20

5.2.1. Operation ratio

The mean operation ratio of Success cases is 33.3%, while of Failure cases it is 22.0%.

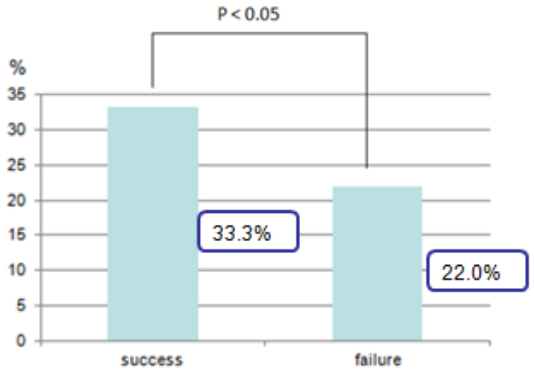

Figure 8. Operation ratio

Figure 8 shows the mean operation ratio of Success and Failure. With U test, $p < 0.05$ ($p = 0.04$), the difference between Success and Failure is marginally significant. Success case had higher operation ratio than Failure case. That is, operation covers more time in Success case.

5.2.2. Operation frequency

The mean operation frequency of Success cases is 3.02 numbers in one minute, and of Failure cases it is 3.10 numbers in one minute. Simply from the mean values shown in Figure 9 we can even see the difference between the two samples is not significant.

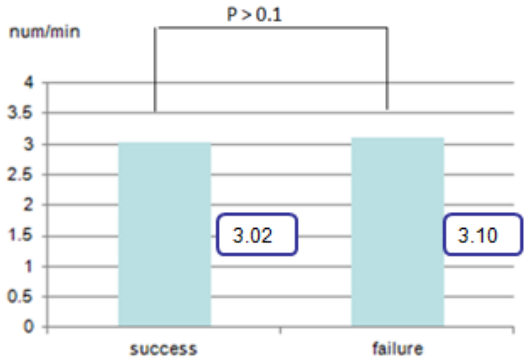

Figure 9. Operation frequency

And with Mann-Whitney U test, $p > 0.1$ ($p = 0.86$), there is no significant difference of operation frequency between Success and Failure cases. So as the result of the test, we cannot say that Success case is with lower operation frequency.

5.2.3. Average operation length

For Success cases, each operation lasts for 7.46 seconds averagely, while for Failure each utterance lasts for 4.48 seconds. From our observation and the analysis result shown in Figure 10, we could say that each operation lasts for a longer time in Success case. It is still necessary to test the result in statistical way.

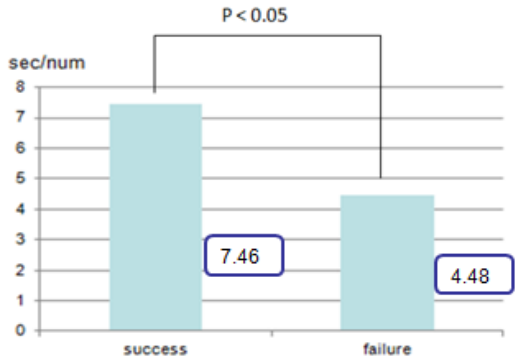

Figure 10. Average operation length

Figure 10 shows the analysis result of average operation length. With Mann-Whitney U test, $p < 0.05$ ($p = 0.03$), the difference of average operation length between Success and Failure is marginally significant. Success case has longer average operation length than Failure case. As a result, each operation lasts for a longer time in Success case.

5.3. Result of operation after Driver and Navigator's dialogue

As described in the hypothesis, we analyzed the operation after dialogue by analyzing "Ratio of operation after dialogue", and "Frequency of operation after dialogue".

Table III shows the data of operation after Driver and Navigator's dialogue in 2011 pair programming. The "data length" and "operation numbers" are automatically output by ELAN. We counted the numbers of operation after Driver and Navigator's dialogue in each case and listed the result as the "numbers of operation after dialogue" column.

Table III. Operation after driver & navigator's dialogue data of 2011 pair programming

case	data length (s)	operation numbers	number of operation after dialogue	Ratio of operation after dialogue (%)	Frequency of operation after dialogue (num/min)
success 1	52	5	3	60	3.5
success 2	210	11	6	54.5	1.7
success 3	72	2	1	50	0.8
success 4	60	4	4	100	4.0
success 5	404	24	10	41.7	1.5
success 6	61	4	2	50	2.0
success 7	219	8	5	62.5	1.4
success 8	403	20	16	80	2.4
success 9	98	3	2	66.7	1.2
success 10	207	8	3	37.5	0.9
success 11	281	14	2	14.3	0.4
success 12	377	22	12	54.5	1.9
success 13	109	9	5	55.6	2.8
success 14	154	8	4	50	1.6
success 15	284	20	12	60	2.5
success 16	301	14	8	57.1	1.6
success 17	309	12	8	66.7	1.6
success 18	166	4	4	100	1.4
success 19	138	8	5	62.5	2.2

success 20	228	8	4	50	1.1
success 21	158	5	3	60	1.1
success 22	207	9	6	66.7	1.7
success 23	116	5	2	40	1.0
failure 1	774	48	9	18.8	0.7
failure 2	173	14	2	14.3	0.7
failure 3	452	31	6	19.4	0.8
failure 4	456	22	6	27.3	0.8
failure 5	348	24	3	12.5	0.5
failure 6	401	11	3	27.3	0.4
failure 7	599	20	4	20	0.4
failure 8	587	31	7	22.6	0.7
failure 9	286	6	2	33.3	0.4
failure 10	373	12	1	8.3	0.2
failure 11	445	33	7	21.2	0.9
failure 12	502	32	2	6.3	0.2
failure 13	395	12	2	16.7	0.3

5.3.1 Ratio of operation after Driver and Navigator's dialogue

The mean ratio of operation after (Driver and Navigator's) dialogue of Success cases is 58.3%, while of Failure cases it is 19.1%. From the value shown in Figure 11 we can see obvious difference between the two samples, but we still should assess that whether there is significant difference with a statistically test.

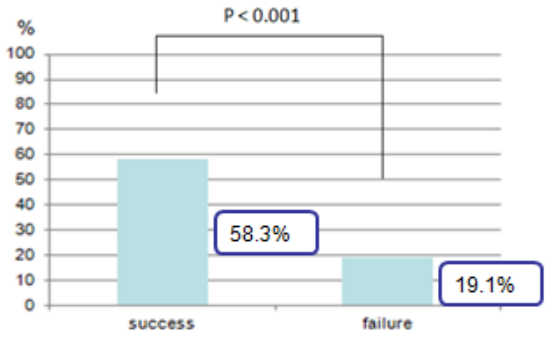

Figure 11. Ratio of operation after Driver and Navigator's dialogue

With Mann-Whitney U test, $p < 0.001$ ($p = 4.01002e\text{-}06$), the difference between Success and Failure is highly significant. We can get the result that Success case had higher ratio of operation after dialogue than Failure case. That is, operation after dialogue covers more percentage among the total operation numbers in Success case.

5.3.2 Frequency of operation after Driver and Navigator's dialogue

The mean frequency of operation after (Driver and Navigator's) dialogue of Success cases is 1.75 numbers in one minute, and of Failure cases it is 0.55 numbers in one minute. From the value shown in Figure 12 we can see there is obvious difference between the two samples, with Mann-Whitney U test we can assess whether there is significant difference statistically.

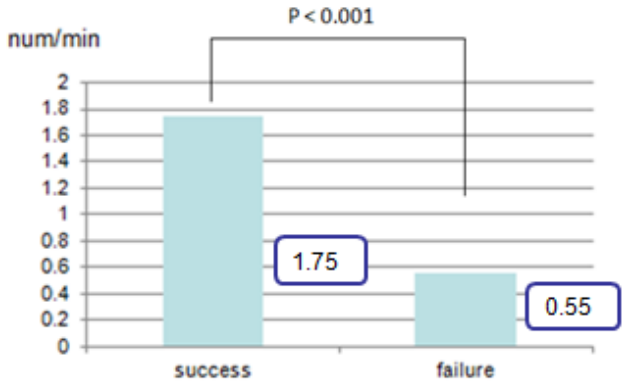

Figure 12. Frequency of operation after Driver and Navigator's dialogue

With U test, $p < 0.001$ ($p = 5.0799e\text{-}06$), the difference of frequency of operation after dialogue between Success and Failure is highly significant. As the result shown, Success case had higher frequency of operation after dialogue than Failure case. That is, there are more numbers of operation after Driver and Navigator's dialogue in one minute in Success case.

Chapter 6

Discussion

The Utterance analysis results shown in Chapter 5 presented that Success case has shorter average utterance length than Failure case. In Success case, students' each utterance lasted for shorter time. As to the utterance ratio and utterance frequency, no significant differences were found between Success and Failure cases.

From the observation and analysis, as the result shown, in Success case, the operation ratio was higher, and the average operation length was longer. In another word, operation covered more time and each operation lasted for a longer time in Success case. It is not surprise to get the result that Success had more operation time and average length than Failure. According to the observation of the data, students failed in problem-solving usually had more other behavior such as searching in the textbook or writing on the paper because they need ideas and solutions to the problem. And students in Success case, they solved the problem smoothly with the knowledge they have acquired, so the time to search for solutions had been solved, they typed the code fluently, which resulted in more operation time and longer average operation length in Success.

As it was expected in the hypothesis of operation after Driver and Navigator's dialogue, it was proved that Success case had higher ratio and frequency of "operation after Driver and Navigator's dialogue". Success case had higher ratio and frequency of operation after dialogue than Failure case. From the observation of the data, this dialogue was mainly the opinion exchange between driver and navigator, which should

be one kind of cooperative work between the pair. As mentioned in previous researches, cooperation was found as one factor what would influence the efficiency in many domains, including the programming field. Students in programming course performed in high efficiency because of the cooperative activities, their retention and performance were increased and boosted. In this study, one cooperation-related behavior pattern was assumed. It was found that operation after dialogue covered more percentage among the total operation numbers, and there were more number of operations after Driver and Navigator's dialogue in one minute in Success case. Dialogue between the pair showed the knowledge and opinion exchange and cooperation in pair programming. With this, decision in higher quality which agreed by both was supposed to be made and then operated by the driver. This kind of operation is effective at the result of a case. As Chong said, their pair programming partner could give suggestions, but fundamentally, the driver, that is, the developer at the keyboard decided which suggestion to follow [10]. So if the driver did not agree with the suggestion, he would not type the code, and then what the partner said became meaningless. For future direction of this study, we plan to conduct the control experiment of pair programming to see whether the cooperative work would really affect the programming result and are considering what element should be controlled now.

Chapter 7

Conclusion

Programming is the process of designing and writing the code to make the computer solve a problem. In order to enable the computer to understand the human's intent, the ideas, methods, and the means of solving the problem should be organized then input to the computer, and then it could accomplish a specific task step by step, by following the given instruction. As the programming requirement increasing, pair programming was originated in industry as a key component of the eXtreme Programming (XP) development methodology. It improves software quality and responsiveness to change customer requirements, and reduces the cost of software development.

In this study, we observed the pair programming practice sessions from a course named "Programming I", and obtained the problem-solving periods as cases then analyzed them. We reconfirmed that Success case had shorter average utterance length, which has also already obtained by Hirai's analysis of 2010 pair programming data; then we found that Success case had higher operation ratio, and longer average operation length than Failure case. We also presented that Success case had higher ratio and frequency of operation after dialogue than Failure case. We would like to learn more about the symptoms which could make pair programming learning and cooperative work more effective and plan to conduct one control experiment to see the cooperative pattern's impact on pair programming in the future.

Reference:

[1] L. L. Constantine, Constantine on Peopleware. Englewood Cliffs, NJ: Yourdon Press, 1995.

[2] Beck, K. (1999). Extreme Programming Explained: Embrace Change, Reading, PA: Addison-Wesley.

[3] Nosek, J. T. (1998) The Case for Collaborative Programming. Communications of the ACM, 41 (3), 105-108.

[4] Williams, L., Kessler, R., Cunningham, W., Jeffries, R. (2000). Strengthening the Case for Pair programming. IEEE software, 17 (4), 19-25.

[5] McDowell, C., Werner, L., Bullock, H., Fernald J. (2002). The Effects of Pair programming on Performance in an Introductory Programming Course, Proc. ACM SIGCSE, ACM Press, 38-42.

[6] Muller, M. M. (2003). Are Reviews an Alternatvie to Pair Programming? Seventh International Conference on Empirical Assessment in Software Engineering, UK.

[7] Nagappan, N., Williams, L., Ferzli, M., Wieve, E., Yang, K., Miller, C., and Balik, S. (2003). Improving the CS1 Experience with Pair programming, Proc. ACM SIGCSE, ACM Press, 359-362.

[8] Sfetsos, P., Stamelos, I., Angelis, L., Deligiannis, I. S. (2006). Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming, in XP/Agile 7, 43-52.

[9] Bryant, S., Romeo, P., Boulay, B. (2006). Pair Programming and the e-appropriation of Individual Tools for Collaborative Software Development, Proc. ACM SIGGROUP, 55-70.

[10] Chong, J., and Hurlbutt, T. (2007). The Social Dynamics of Pair programming, Proc. International Conference on Software Engineering (ICSE), IEEE Press, 354-363.

[11] Hirai, Y., Inoue, T. (2012). Collaboration Estimation in Pair Programming Learning: Conversation Differences between Success and Failure in Problem Solving, Infomration Processing Society of Japan Journal, Vol 53, 72-80.

[12] Lori, P., Mike, J. (2001). Making Parallel Programming Accessible to Inexperienced Programmers through Cooperative Learning. SIGCSE 2001, 224-228.

[13] Edward, F. G., Katherine, D., Keith, J. W., John, H. (2006). Panel: Cooperative Learning – Beyond Pair Programming and Team Projects. SIGCSE 2006, 458-459.

[14] Duo, Wei. (2012). An Evaluation of a Cooperative Learning Method in Programming and Problem Solving I. Consortium for Computing Science in Colleges, 69-77.

[15] Gary, J. C., Cheryl, L. D., Severin, V. G. (1991). Information Exchange Patterns in a Computer-Supported Cooperative Work Environment. SIGCHI 1991, 57-58.

[16] Dacid, M., Mark, R., Ian, S. (2002). Applying Patterns of Cooperative Interaction to Work (Re)Design: E-Government and Planning. CHI 2002, 235-242.

[17] Anastasiia, B., Peter, Q., Karin, C., Wim, L. (2012). The Influence of Cooperative Game Design Patterns for Remote Play on Player Experience. APCHI'12, 11-19.

[18] Claus, B., Lotte, G. J., Flemming, W. (2012). Medical Secretaries' Care of Records: The Cooperative Work of a Non-clinical Group. CSCW'12, 921-930.

[19] Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., Sloetjes, H. (2006). ELAN: a Professional Framework for Multimodality Research. Proceedings of LREC 2006, Fifth International Conference on Language Resources and Evaluation.

[20] ELAN (EUDICO Linguistic Annotator), Available from http://tla.mpi.nl/tools/tla-tools/elan/