Mohd Ayub A. F. et al. (Eds.) (2011). Workshop Proceedings of the 19th International Conference on Computers in Education. ChiangMai, Thailand: Asia-Pacific Society for Computers in Education.

Toward better Collaborative Problem-solving in Programming Learning: Use of Pair Programming and Its Observation

Yuki HIRAI[a*] & Tomoo INOUE[a]**
[a]*Graduate School of Library, Information and Media Studies, University of Tsukuba, Japan*
*s0930522@u.tsukuba.ac.jp **inoue@slis.tsukuba.ac.jp

Abstract: To realize better programming learning, we have adopted "pair-programming" method to the introductory programming course of a university. Pair-programming is a programming method that two persons get involved with a single programming task using a single computer terminal, where only one person types the keyboard. Though there have been several researches on pair-programming learning practice and been reported its usefulness, only impressions were reported. Through the actual pair-programming practice, we could observe both successful case and failed case in solving the problem that arose in the course of completing the assigned task, and found that there seemed to be difference in utterance patterns between the successful case and the failed case.

Keywords: Pair-programming, Programming learning, Problem-solving, Computer-supported collaborative work (CSCW)

Introduction

The ability to understand the grammar of a programming language, to write a program, and to assemble an algorithm, is required in programming education. When a learner actually creates a program, some problems typically occur, even if the grammar and a (relatively easy) example of the program language are understood [12]. In programming education, numerous practices, including the support of problem-solving, have been developed to date. Education and study methods have also received considerable attention [10].

The programming method called 'pair-programming' originated in industry as a key component of the Extreme Programming (XP) development methodology [1]. As the name suggests, two programmers work together at the same machine while developing code. One programmer (the driver) operates the keyboard and focuses on entering code, while the other programmer (the navigator) observes the work of the driver and offers suggestions in the code. The programmers regularly exchange roles. Creating a program by pair-programming is collaborative work, and offers further benefits in respect of sharing and enhancing programming expertise and refining collaborative technique [16]. In the Computer-supported cooperative work (CSCW) context, interruptions of software teams have been investigated [4], and studies have explored interruption patterns among software developers who program in pairs versus those who program solo.

In some programming education, pair-programming has been conducted as one of the programming learning methods. Especially in introductory programming courses, for example, it has been reported that pair-programming is better than solo-programming in respect of improving the quality of programming [6,7,8,11,15]. However, there were numerous instances in which the pair-programming had faced the problem which

problem-solving did not go well. Of course, the effect of pair-programming varies with the actual composition of the pairs, but failure on the part of one of the pair, in the introductory stage, can easily spill over into later, more involved tasks. Moreover, if problem-solving does not go well, a decrease in motivation to study will typically occur. In this case, we must seek to support the pair, with a view to improving their pair-programming learning.
In this study, pair-programming was conducted in an introductory programming course. Success and failure cases in pair-programming were compared. In the comparison, we focused on the conversation between the pair in pair-programming. In the failure cases, it was found that speech length tended to be long, and there might be a great deal of continuous speech.
Our research objective in broader sense is to support programming learning. Pair programming has been focused as one of the promising techniques of programming learning. We do not intend to just using pair programming. We intend to expand pair programming to computer-supported pair programming (CSPP). This means that a computerized environment (not the computer used for programming basically) senses the learning status of the pair, and once the environment senses something wrong with the pair it intervenes in the learning. This could be a mixture of ICAI (Intelligent Computer-Aided Instruction) and CSCL (Computer Supported Collaborative Learning) under the ubiquitous computing technology. To realize such CSPP, we thought we need some symptoms to indicate the status of pair programming. This led to the study in this paper.

1. Related Works

1.1 Pair-programming in an Introductory Programming Course

Previous research suggested that pair-programming was better than solo-programming in numerous respects. For example, it was better in respect of the quality of program code [6,7], the success rate in programming courses [7,8], results of mid-term or final examinations [8], and/or submission rate of assignments [15]. Rountree et al. reported that understanding and/or ability to create program code were improved after pair-programming was conducted [11].
The aforementioned research reported the positive effects of pair-programming, but did not analyze the process of pair-programming or the pairs whose problem-solving did not go well. In this research, the conversations of some pairs in pair-programming were analyzed, and specifically, pairs that failed in problem-solving were studied.

1.2 Communication Analysis in Pair-Programming

In previous research (which did not focus on introductory programming courses), conversations of the pairs in pair-programming was analyzed. Chen et al. recorded the utterance of pairs and described the context of pair-programming. They suggested that there was a mental distance between the driver and the navigator, and communication supports such as visualizing the rules of the pair were necessary [3]. Chong et al. also recorded the utterance of pairs and described the context of pair-programming. They suggested that the distribution of expertise among the members of a pair had a strong influence on the tenor of pair-programming interaction, and keyboard control had an effect on decision-making within the pair [5]. Bryant et al. investigated the distribution of utterance categories in pair-programming, and suggested that there was no significant difference in the distribution between the driver and navigator, and both driver and navigator work at similar levels of abstraction [2].

These studies analyzed the conversation of pairs, but did not compare success and failure cases in pair-programming. In this study, interactions between the driver and navigator have been observed, communications in pair-programming have been analyzed, success and failure cases have been compared, and the characteristics of failure cases have been studied.

1.3 Roles of Conversation in Pair-Programming

Wray [16] described the roles and effects of conversation in pair-programming from his own experience. He mentioned that the roles of conversation were sharing expertise among pairs and getting on the track for problem solving. He predicted that programmers who chat about their programs more should be more productive and that those who pose deep questions for each other should be most productive of all.
His description suggests that problems occurring in pair-programming might be solved through conversation among pairs, and that conversation may be a significant indicator in comparisons between success and failure cases in pair-programming. In the present study, differences in conversation between success and failure cases in pair-programming were explored.

2. Pair-Programming Practice

2.1 Practice Setting

In this study, pair-programming was conducted in an introductory programming course, "Programming I", which targeted freshmen in the university's department of information. The goals of the course were as follows:

- Learners understand the description and composition of software and the mechanism of programming.
- Learners can compile and execute a program written in C language.
- Learners understand the basis of C language, such as variables, control of flow, functions, arrays, character and string handling, and file I/O.

The course involved ten weekly 75-minute lectures, from September 2010. Pair-programming was conducted in six 30-minute practice sessions as the part of the lecture.
As preparation for pair-programming practice, the training session was conducted. The training was conducted in the same setting as the following pair-programming practice, because of the possibility that some learners had not experienced pair-programming.
In each pair-programming practice session, a program-creation assignment, involving contents hitherto studied, was given to the participants. An example of the assignment is shown in Table 1. The following six instructions were given to the learners:

- Only the driver can operate the keyboard and mouse. The navigator must not touch them, but may point to the display. The navigator must observe and support the work of the driver.
- The assignment ends when the program is executed and a correct answer to the assignment is obtained. Please end the assignment as soon as possible.
- The driver and navigator may refer to the textbook [14]. You must not refer to any web pages.
- The teacher and teaching assistants (TA) do not accept any questions concerning the assignment while practicing. Please call on them only in the event of equipment trouble.
- Please create the program easy to understand by adding pertinent comments.
- You have 30 minutes to success the assignment. Please submit your code even if failure, when the time limit is reached.

A total of 62 learners participated in the practice session (52 freshmen and 10 upper-years). Pair combinations were decided by one of the authors. The participants did not exchange roles (of driver and navigator) in each practice session because the practice time was short. Figure 1 shows a screenshot of the practice session. Figure 2 shows a scene from the practice session. Three cameras were used for recording communication.

Table 1. An example of the exercises in the pair-programming class.

Assignment 1:
Create a program for permutation and combination according to the following specification.

Specification
* Input: n, r (integer)
* Output: "nPr = ?, nCr = ?"(? is calculated value)

Example
When 8 is input to n and 3 is input to r, the calculated result is displayed as follows:
8P3 = 336, 8C3 = 56

Hint
As for permutation and combination, the general formulas are given as follows:

$$_nPr = \frac{n!}{(n-r)!} \quad (n \geq r > 0), \quad _nCr = \frac{n!}{r!(n-r)!} = \frac{_nPr}{r!} \quad (n \geq r > 0), \quad n! = \begin{cases} n \times (n-1)! & (n \geq 2) \\ 1 & (n = 0,1) \end{cases}$$

Figure 1. Setup of the cameras for data collection.

Figure 2. Scene from the practice session.

2.2 *Definition*

In this study, "Success" , "Failure" and "Problem" are defined as follows:

- Success: "Success" is the identifier of showing the problem was solved. It does not relate to the learner's "success" of learning.
- Failure: "Failure" is the identifier of showing the problem was not solved within the given limited time. It does not relate to the learner's "failure" of learning.
- Problem: "A problem" is a compilation error that occurs when learners compile their program, or a runtime error that occurs runtime including whose result does not meet the learners' expectation.

Although we know those concepts of Vygotsky's Zone of Proximal Development and Lave and Wenger's Legitimate Peripheral Participation, and "failure" is not just failure there [9], we do not deal with that "failure" in this paper. There failure can be resource for learning. Here the term "failure" is used as an identifier of unsuccessful result of solving the error that occurred during programming. In other words, the term "failure" and "success" in this paper do not imply any notion known in learning sciences. They are simple and clear identifiers of the result of solving the errors.

2.3 Problems Occurring in the Practice Session

Table 2 shows the problems which occurred among the pairs whose communication was recorded. These problems occurred in pairs of first-year students. Table 2 shows six success cases and three failure cases. Some pairs attempted to solve two or more problems in a given practice session. Problem-solving went well in the success cases. The problems listed in Table 2 were causes of the error that the pair finally identified. In Failure Case A and B, problems which the authors recognized by observing the video are listed, because the respective pair did not recognize the cause of error. There were only three failure cases in this practice session. This is because the assignments given to the participants were easy. Most of the pairs completed the assignment within the time limit.

Table 2. Problems occurring in the practice session.

Case	Pair	Problems
Success A	Pair A	Compilation error Semicolon was not written at the end of a line.
Success B	Pair B	Compilation error The string "enum" was a reserved word.
Success C	Pair B	Compilation error The source file was not preserved in the superscription.
Success D	Pair B	Compilation error, Segmentation error The "scan" sentence was written like 'scanf("%d", a);'. – "&" was missing.
Success E	Pair C	Run-time error Beginning of a block did not correspond to the end. There were some spelling mistakes.
Success F	Pair C	Run-time error The return value of a function was not correctly returned.
Failure A	Pair D	Run-time error The case divided by 0 was included in the "for" sentence.
Failure B	Pair A	Run-time error The value of a variable was not correctly substituted by the global variable declaration.
Failure C	Pair E	Compilation error Neither the main file nor the header file was correctly linked.

3. Difference between Success and Failure Cases

Success and Failure cases in problem-solving were analyzed and compared in term of pairs' conversation. The utterances of the pairs and the context of pair-programming were recorded with iCorpusStudio [13], which is a video-analysis support tool. With the tool, we can simultaneously view the recorded data as multiple video, audio, and motion, while annotating the interpretations of the interactions as labels.

3.1 Examples of Success and Failure Cases

We show two example sequences including utterances and some descriptions; one for "Success" case and the other for "Failure" case.
Table 3 shows a conversation in Success case A. In this case, the following error message "19: error: expect ';' before 'return'" was output. The learners solved this problem in 100 seconds. Speech length marks the time from the point that the learner started his/her speech, to the point that the learner ended the speech.
Table 4 shows a part of conversation in Failure case B. In this case, there was no output though the program was executed and the driver input a value to a variable. The learners tried to move the "while" sentence to another line. The learners spent 588 seconds solving this problem, but the problem was not solved. The driver uttered 19 times in this case, while the navigator uttered 61 times.

Table 3. A conversation in Success case A.

Utter. no.	Speaker	Speech length (sec.)	Utterance
1	D	0.9	The 19th line.
2	D	0.9	Ah… This line.
3	N	1.5	Ah… "return 0".
4	N	4.1	Line numbers are shown when a setting is changed.
5	D	0.9	Really?
6	D	1.8	I do not compile this program.
7	N	2.7	Did you save this program? Ah, you did.
8	D	1.9	I try to delete unnecessary lines.
9	D	1.1	(I think) the way is not good.
10	N	1.6	return 0…
11	D	0.7	This point
12	N	1.4	Ah… after the "printf" sentence.
13	N	0.7	Um…
14	N	1.5	functional…
15	N	1.2	The 19th line
16	N	1.3	No changes are appeared.
17	D	1.5	This program consists of 17 lines.
18	D	1.7	Ah…, 19, the last line…
19	N	3.1	Parentheses… Let's make sure the position of parentheses
20	N	1.7	The number of braces is wrong? …
21	D	0.7	Ok. (the problem was solved)

* Speaker - D: Driver, N: Navigator
* Speech length - The length more than 2 seconds is highlighted.
* Utterance - Description in the parentheses is the supplement by the authors.

Table 4. Part of a conversation in Failure case B.

Utter. no.	Speaker	Speech length (sec.)	Utterance
14	N	1.2	The “while” sentence...
15	D	0.4	Umm.
16	N	1.6	Let’s move outside of the ”main” function.
17	D	1.1	”Main”?
18	N	1.1	Please move above the function.
19	D	0.7	Umm.
20	N	1.7	From this line to this line... Ok.
21	D	1.1	Umm.
22	N	2.6	Please cut the selected lines.
23	N	0.7	Next...
24	N	1.2	Let me see...
25	N	1.2	”While” sentence... (The driver operates.)
26	N	2.9	Not ”while” sentence. Sorry, please undo.
27	N	2.1	Sorry, it became strange.
28	N	4.4	You may move this function outside. (The driver operates.)
29	N	3.5	From this line to this line... (The driver operates.)
30	N	5.6	Because this function was moved outside, the declaration of the variable might be wrong.
31	N	4.0	”jyun” (= a variable) is ok. ”ans” (= a variable) is ok. ”n” (=a variable) is ...
32	N	1.3	”n” is...
33	N	1.9	Is it correct to declare this variable outside the function?
34	N	3.0	Global...?
35	N	0.7	Index...
36	N	4.5	Global... global variable.
37	N	1.6	Ok. It is possible to declare this variable outside the function.

* Speaker - D: Driver, N: Navigator
* Speech length - The length more than 2 seconds is highlighted.
* Utterance - Description in the parentheses is the supplement by the authors.

3.2 Findings obtained from the Examples

As for the speaker, in the failure case, the driver and navigator spoke alternately from utterance 14 to 22. From utterance 23, however, the navigator spoke continuously; that is, the driver did not talk. The navigator spoke more continuously in the failure case than in the success case. As for the speech length, there were 9 utterances that are more than two seconds in length in the failure case. Especially, from utterance 26 to 31, the navigator spoke continuously and all of his succeeding utterances were more than two seconds in length.
The investigation of these example dialogues suggests that there may be a relationship between speech length and/or speech continuity and success/failure of problem-solving.

Discussion of the relation may require further investigation; for example, through observing more cases in the practice sessions.

4. Conclusion

We have adopted pair-programming method in software engineering to programming learning. Naturally there occurred both successful case and failed case in solving the problem when the problem arose in the course of completing the task. We observed a few such cases and found that there seemed to be difference in utterance patterns between successful case and failed case. We will analyze the learners' conversation and behavior more in detail to obtain clearer symptoms to indicate the status of pair programming. Then we will develop a computer-supported pair programming system that uses the symptoms.

Acknowledgments

The authors would like to thank Naiwen Tei for her contribution to the data collection. This research was partially supported by the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for JSPS Fellows 23.2956, the JSPS Grant-in-Aid for scientific research 22500104, and the Research Projects of Graduate School of Library, Information and Media Studies, University of Tsukuba.

References

[1] Beck, K. (1999). Extreme Programming Explained: Embrace Change, Reading, PA: Addison-Wesley.
[2] Bryant, S., Romero, P., and Boulay, B. (2008). Pair Programming and the Mysterious Role of the Navigator, *International Journal of Human-Computing Study*, *66*(7), Academic Press, 519-529.
[3] Chen, W. and Nordbo, M. (2007). Understanding Pair-Programming from a Socio-cultural Perspective, *Proc. Computer-supported Collaborative Learning (CSCL)*, International Society of the Learning Sciences (ISLS) Press, 138-140.
[4] Chong, J., and Siino, R. (2006). Interruptions on Software Teams: A Comparison of Paired and Solo Programmers, *Proc. Computer-supported Cooperative Work (CSCW)*, ACM Press, 29-38.
[5] Chong, J., and Hurlbutt, T. (2007). The Social Dynamics of Pair Programming, *Proc. International Conference on Software Engineering (ICSE)*, IEEE Press, 354-363.
[6] Hanks, B., McDewell, C., Draper, D., and Krnjajic, M. (2004). Program Quality with Pair Programming in CS1, *Proc. Innovation and Technology in Computer Science Education (ITiCSE)*, ACM Press, 176-180.
[7] McDowell, C., Werner, L., Bullock, H., and Fernald J. (2002). The Effects of Pair-Programming on Performance in an Introductory Programming Course, *Proc. ACM SIGCSE*, ACM Press, 38-42.
[8] 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.
[9] National Research Council. (1999). How People Learn, National Academies Press.
[10] Onishi, K. (2010). How to Teach What in a Course of Programming?: Foreword. *Information Processing, 51*(10), Information Processing Society of Japan (IPSJ) Press, 1341. (in Japanese)
[11] Rountree, J., Rountree, N., Robins, A., and Hannah, R. (2005). Observations of Student Competency in a CS1 Course, *Proc. Australasian Computing Education Conference (ACE)*, Australian Computer Society Press, 145-149.
[12] Shinkai, J., and Miyaji, I. (2009). Effects of C Programming Education Which Makes a Point of Process with Evaluation Activity. *Journal of Japanese Society for Information and Systems in Education (JSiSE)*, *26*(1), 16-25. (in Japanese)
[13] Sumi, Y., Yano, M., and Nishida, T. (2010). Analysis Environment of Conversational Structure with Nonverbal Multimodal data, *Proc. International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction (ICMI-MLMI)*, ACM Press.
[14] Takahashi, M. (2007). Yasashii C (in Japanese Title), Softbank Creative Press, ISBN-10: 9784797343663.
[15] Urness, T. (2009). Assessment Using Peer Evaluations, Random Pair Assignment, and Collaborative Programming in CS1, *Journal of Computing Sciences in Colleges*, *25*(1), 87-93.
[16] Wray, S. (2010). How Pair Programming Really Works, IEEE Software, Jan./Feb., IEEE Press, 50-55.