Wong, L.-H. et al. (Eds.) (2013). Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education

Relation between Behavior and Result in Pair Programming: Talk and Work Leads to Success

Tomoo INOUE [a*]
[a]*Faculty of Library, Information and Media Science, University of Tsukuba, Japan*
*inoue@slis.tsukuba.ac.jp

Abstract: Pair programming, a programming technique conducted by two programmers work together at one work station, has been adopted for learning programming. Although it is known to be effective in various aspects, micro observation of the learning activity, collaboration, has yet to be conducted in relation to the outcome. In this study, behavior in pair programming learning was investigated in terms of verbal communication and programming action, and was compared in relation to the success of problem-solving. Findings are that, in the successful cases, 1) the learners took programming actions more frequently, and 2) the learners took more programming actions immediately after the dialogue. From this, it is suggested that closely-knit dialogue and action can be an indicator of successful problem-solving, and the findings can be applied to a collaborative learning support systems.

Keywords: Pair programming, behavior analysis, collaborative learning, problem-solving

1. Introduction

In the programming education, the ability to understand grammar of a program language and writing of a program and the ability to assemble the algorithm are required. Computer programming is not only the process of developing code, but also it is more like a process of innovation of programmer's ideas. To improve the effectiveness and efficiency of programming, collaborative programming came into being(Constantine, 1995).

As one major form of collaborative programming, pair programming was originated in industry as a key component of the eXtreme Programming (XP) development methodology(Beck, 1999). It 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 controls a keyboard and a mouse is called a "driver," and another who is responsible for observing the code input, giving suggestions, contributing to the programming verbally, is called a "navigator".

Pair programming has been accepted in more and more fields because of the higher code quality created and less time spent compared with solo programming (Nosek, 1998; Williams, Kessler, Cunningham, and Jeffries, 2000; McDowell, Werner, Bullock, and Fernald, 2002; Muller, 2004). Furthermore, it could improve programmers' and learners' programming experience and their cooperative consciousness (Muller, 2004; Nagappan, Williams, Ferzli, Wieve, Yang, Miller, and Balik, 2003). However, of course the learners in pair programming face with problems in their learning activity. They are sometimes successful in problem-solving but not always. Unsuccessful problem-solving can also be useful in learning, but in general the problem-solving not going smoothly might decrease the learners' motivation or might result in the learners' negative emotion.

Therefore to know how to be more successful in problem-solving in pair programming is an important issue. Around this issue are research as follows: The programmers' behavior plays a key role in the performance of pair programming (Sfetsos, Stamelos, Angelis, and Deligiannis, 2006; Bryant, Romero, and Du Boulay, 2006; Chong and Hurlbutt, 2007; Hirai and Inoue, 2012). The cooperative work between the pair has an immediate influence on the programming result and experience (Pollock and Jochen, 2001; Gehringer, Deibel, Whittington, and Hamer, 2006; Wei, 2012). However there are few studies that focused on the micro interaction between the learners in pair programming with quantitative results.

In this study, pair programming was conducted in an introductory programming course. The pair's behaviors in programming were videotaped and analyzed regarding the learner's utterance and the programming-related operation of a personal computer. As a result, it was found that operation covered more time and each operation lasted for longer time in the success cases. Moreover, the behavior pattern "the operation after dialogue" was more common in the success case, in terms of the percentage to all the operations and in terms of frequency per minute.

The rest of the paper is structured as follows. Literature review is presented in the next section. The data collection and the data analysis are described in Section 3. The analysis results are presented in Section 4. After the discussion of the results in Section 5, Section 6 concludes the paper.

2. Literature Review

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. Pair teams were found to usually develop the program and software with higher quality (Nosek, 1998; Williams, Kessler, Cunningham, and Jeffries, 2000; McDowell, Werner, Bullock, and Fernald, 2002), but the time spent was shorter than individual programmers (Williams, Kessler, Cunningham, and Jeffries, 2000). Programmers worked in pair were more self-sufficient, generally performed significantly better on projects and exams (McDowell, Werner, Bullock, and Fernald, 2002; Nagappan, Williams, Ferzli, Wieve, Yang, Miller, and Balik, 2003).

These researches have shown the efficiency of pair programming, but they did not get into the details of behaviors or the interaction processes. In contrast, the granularity of our research is on behavior level. This study focuses on the behavioral difference in relation to the outcome.

2.2 Behavior Analysis in Pair Programming

Behavior in pair programming has been paid increasing attention in more researches. Sfetsos, Stamelos, Angelis, and Deligiannis (2006) showed that productivity for pairs is positively correlated with communication transactions. Bryant, Romero, and Du Boulay (2006) presented that the expertise distribution would influence the pair communication interaction, and noticed that the operation behavior was assisting intra-pair verbal communication. Chong and Hurlbutt (2007) presented that the 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. Hirai and Inoue (2012) compared the utterances in successful and in unsuccessful problem-solving cases, and found that the successful cases had longer utterances, less repeating explanations and less continuous speeches.

Cooperation plays important role in many group works in different domains. Cooperative behavior was also regarded as key component in pair programming and analyzed in some researches. Pollock and Jochen (2001) presented that the cooperative behavior of pair in a programming course, such as think-pair-share, group question and role play, made students work in high efficiency. Gehringer, Deibel, Whittington, and Hamer (2006) presented that the similar cooperative behavior increased retention and boosted the performance of at-risk students. Wei (2012) learnt that cooperative learning method was perceived to be effective in teaching programming classes from students' survey in a pair programming course.

Our study, on the other hand, focuses on the relation between the behavioral difference and the result of cooperative problem-solving based on the objective data, where the behavior includes both utterance and programming operation.

3. Method

3.1 Data Collection

The pair programming data was collected from one introductory programming course named "Programming I," where C language was the major teaching content. The target of the course is freshmen in the university's school of informatics. The aim of the course is to let the students understand the grammar of the programming language and write a program. Each lecture of the course lasted 75 minutes, where 30 minutes were allocated for pair programming practice. A part of the practice was videotaped.

In each pair programming practice session, a programming exercise from where already taught was given to the students. Preparations were done before the data collection, such as the pair combination, the initial Driver/Navigator role assignment for each pair.

Three cameras from different angles were set for a pair; one for recording the pair's communication, one for recording the pair's behavior and working activities such as typing a keyboard, using a mouse, pointing at the display, referring to the textbook, and the last one for recording the screen. Figure 1 shows the scenes from those three cameras.

Figure 1. Scenes from the recorded data.

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

- ☐ The time limit is 30 minutes. However the exercise should be completed as soon as possible.
- ☐ Only the driver can operate a keyboard and a mouse. The navigator should observe and support the driver's work without touching the keyboard and the mouse.
- ☐ The textbook can be consulted with but not the Web.

3.2 Data Analysis

A pair encounters problems in a programming session. They may or may not be solved successfully. We regard each problem-solving process as a case. A case begins with the problem encountered and ends with it being solved or ends when the time is up. A session can include multiple cases. A case with successful result is called the success case and a case with unsuccessful result is called the failure case in this paper.

A video annotation tool ELAN was used to synchronize the three videos into one integrated video and to label and annotate behaviors on the integrated one. Thirty three sessions were analyzed and 29 success cases and 16 failure cases were found.

As the major behaviors, utterance and computer operation were labeled. An operation includes typing a keyboard and handling a mouse.

Besides the basic values such as average length of an utterance, average length of an operation, and the percentage of operation in the data, two behavior patterns were investigated. One is "operation after dialogue" pattern, where the last two utterances before an operation comprise a dialogue. The other is "operation accompanied by dialogue" pattern, where a dialogue continues after an operation.

4. Results

As for utterance, the average length of an utterance in the success cases (1.82 sec.) was significantly shorter than that in the failure cases (2.52 sec.) with Mann-Whitney U test ($p = 0.008$).

As for operation, the percentage of operation in the success cases (34.0%) was significantly higher than that in the failure cases (22.1%) with Mann-Whitney U test ($p = 0.02$). Also the average

length of an operation in the success cases (7.80 sec.) was significantly longer than that in the failure cases (4.99 sec.) with Mann-Whitney U test ($p = 0.03$).

As shown in Figure 2, the ratio of the operations after dialogue in the success cases (53.8%) was significantly higher than the one in the failure cases (16.0%) with Mann-Whitney U test ($p < 0.001$). Similarly, as shown in Figure 3, the frequency of the operations after dialogue in the success cases (1.53 per min.) was significantly higher than the one in the failure cases (0.44 per min.) with Mann-Whitney U test ($p < 0.001$).

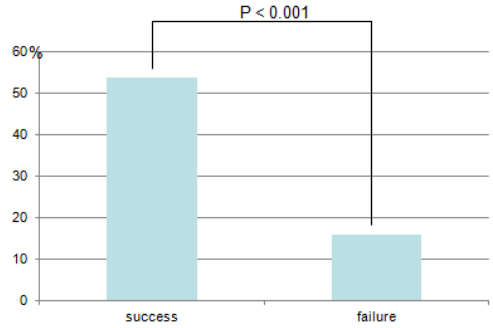

Figure 2. Ratio of the operations after dialogue to the overall operations.

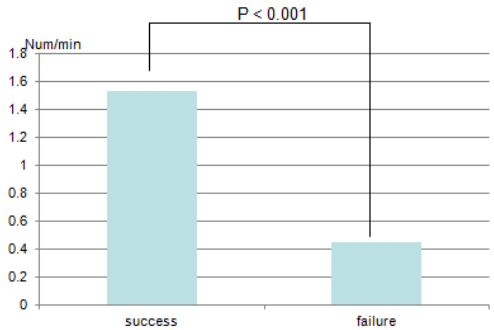

Figure 3. Frequency of the operations after dialogue per minute.

As for the other “operations accompanied by dialogue” pattern, no significant difference was found between in the success cases and in the failure cases in terms of the ratio of the operations accompanied by dialogue to the overall operations, and the frequency of the operations accompanied by dialogue.

5. Discussion

The utterance analysis resulted that the success cases had shorter average utterance length than the failure cases. As for the utterance ratio and the utterance frequency, no significant differences were found between the success and failure cases. The same results were reported in Hirai and Inoue (2012), and we reconfirmed them by using more data in this study. In their research, the rates of repeated explanation and the rates of consecutive speech were also analyzed. Students who failed in problem-solving explained more to each other and had more repeated sentences possibly because of non-smooth understanding of each other. In our study, although the utterance behavior was analyzed because it was one of the basic parameters, more foci were on the operation behavior and the behavior patterns that have not been investigated.

In the success cases, the operation ratio was higher, and the average operation length was longer. Operation covered more time and each operation lasted for a longer time in the success cases. According to our observation, students who failed in problem-solving often took other actions such as searching in the textbook or writing on the paper because they need to search for ideas and solutions to the problem. In contrast, students in the success cases generally could solve the problem smoothly with the knowledge they already have. This means that the time for searching for solutions was not

needed and that they typed the code fluently. This could be the reason of more operation time and longer average operation length in the success cases.

The success cases had higher ratio and frequency of operation after dialogue than the failure cases. From our observation of the data, the dialogue was mainly the opinion exchange between driver and navigator, a kind of cooperative work between the pair. With this, it is suggested that the decision of higher quality was made through the dialogue and then operated by the driver. As presented in the previous researches, cooperation was found as a factor that would influence the efficiency in many domains, including programming. Students in a programming course performed in high efficiency because of the cooperative activities, their retention and performance were increased and boosted.

Although not statistically significant with this data, the number of operations accompanied by dialogue covered more percentage to the total number of operations in the success cases. Similarly more number of operations accompanied by dialogue was found in a minute in the success cases. We noticed that students in the success cases liked to ask their partner about the operation just done, or preferred to explain the reason of the operation. There was more favorable interaction and good understanding between the pair in the success cases.

We analyzed two behavior patterns in our study, “operation after dialogue” and “operation accompanied by dialogue”. They are two different patterns, but overlapping between these two patterns is possible as shown in Figure 4.

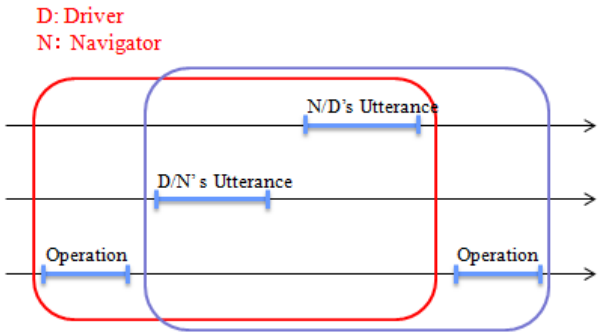

Figure 4. Possible overlap of “operation after dialogue” and “operation accompanied by dialogue” patterns.

This kind of overlap happens when only the behavior was considered. From only the behavior, we cannot judge that if the dialogue was related to the preceding operation or the coming operation. To make the behavior pattern analysis accurate, we took the dialogue content into consideration. What the pair programmers talked was transcribed, and which operation was related to the dialogue was judged.

6. Conclusion

In this study, pair programming practice sessions were conducted, recorded and analyzed in relation to the success and failure of the problem-solving results. We found that, in the successful cases, 1) the learners took programming actions more frequently, and 2) the learners took more programming actions immediately after the dialogue. From this, it is suggested that closely-knit dialogue and action can be an indicator of successful problem-solving.

These findings can be applied to a design of collaborative learning support systems. When a system finds behavioral signals that indicate future probable failure of collaborative problem-solving, such as long utterance length, few programming actions, and few patterns of operation after dialogue, the system can recommend the learners to take more opposite behaviors for successful result. We will develop this adaptive learning environment based on the obtained results.

Acknowledgements

We would like to thank Junshan Hu who helped to prepare the previous version of this document.

References

Constantine, L. L. (1995). Constantine on Peopleware. Yourdon Press.

Beck, K. (1999). Extreme Programming Explained: Embrace Change. Addison-Wesley.

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

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

McDowell, C., Werner, L., Bullock, H., & Fernald J. (2002). The Effects of Pair programming on Performance in an Introductory Programming Course. *Proceedings of the ACM SIGCSE'02* (pp.38-42). ACM Press.

Muller, M. M. (2004). Are Reviews an Alternative to Pair Programming? *Journal on Empirical Software Engineering, 9*, 335-351.

Nagappan, N., Williams, L., Ferzli, M., Wieve, E., Yang, K., Miller, C., & Balik, S (2003). Improving the CS1 Experience with Pair programming. *Proceedings of the ACM SIGCSE'03* (pp.359-362). ACM Press.

Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. S. (2006). Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming – An Empirical Study. *LNCS 4044*, 43-52.

Bryant, S., Romero, P., & Du Boulay, B. (2006). Pair Programming and the Re-appropriation of Individual Tools for Collaborative Software Development. *Proceedings of the 2006 Conference on Cooperative Systems Design* (pp.55-70).

Chong, J., & Hurlbutt, T. (2007). The Social Dynamics of Pair programming. *Proceedings of the International Conference on Software Engineering* (pp.354-363).

Hirai, Y., & Inoue, T. (2012). Collaboration Estimation in Pair Programming Learning: Conversation Differences between Success and Failure in Problem Solving. *Journal of Information Processing Society of Japan, 53*(1), 72-80.

Pollock, L., & Jochen, M. (2001). Making Parallel Programming Accessible to Inexperienced Programmers through Cooperative Learning. *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education* (pp.224-228).

Gehringer, E. F., Deibel, K., Whittington, K. J., & Hamer, J. (2006). Panel: Cooperative Learning - Beyond Pair Programming and Team Projects. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp.458-459).

Wei, D. (2012). An Evaluation of a Cooperative Learning Method in Programming and Problem Solving I. *Journal of Computing Sciences in Colleges, 28*(3), 69-77.

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., & Sloetjes, H. (2006). ELAN: A Professional Framework for Multimodality Research. *Proceedings of the 5th International Conference on Language Resources and Evaluation* (pp.1556-1559).