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

Tomoo Inoue*

(Received 13 April 2013 and accepted in revised form 9 December 2013)

Abstract  Pair programming, a programming technique conducted by two programmers working together at one workstation, has been adopted for learning programming. Although it is known to be effective in various aspects, micro observation of the learning activity and 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 then behavior was compared in relation to the success of problem-solving. In successful cases, it was found that: 1) the learners took programming actions more frequently and 2) the learners took more programming actions immediately after the dialogue. This suggested that closely-knit dialogue and action can be an indicator of successful problem-solving, and the findings can be applied to collaborative learning support systems.

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

1. Introduction

In programming education, the ability to understand the grammar and writing of a program language and the ability to assemble the algorithm are required. Computer programming is not only the process of developing the code, but it is also a process of innovation of programmer’s ideas. To improve the effectiveness and efficiency of programming, collaborative programming came into being(1).

As one major form of collaborative programming, pair programming was originated from industry as a key component of the eXtreme Programming (XP) development methodology(2). It is conducted by two persons who work on one machine including one display, one keyboard and one mouse. The programmer who controls the keyboard and mouse is called the “driver,” and the other who is responsible for observing the code input, giving suggestions, contributing to the programming verbally, is called the “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(3–6). Furthermore, it can improve programmers’ and learners’ programming experiences and their cooperative consciousness(6, 7). However, learners in pair programming may face problems in their learning activities. They are sometimes successfully solved, but not always. Unsuccessful problem-solving can also be sometimes useful in learning, but in general, a problem that is not solved smoothly tends to decrease learners’ motivations or to result in learners’ negative feelings.

Therefore we think it is meaningful to design a learning environment that promotes successful problem-solving in the process of pair programming. For this purpose, before actually designing such an environment, we need to know how to be more successful in problem-solving in pair programming.

There are several research studies related to this issue and they have included the following two findings: The programmers’ behavior plays a key role in the performance of pair programming(8–11) and the cooperative work between the pair has an immediate influence on the programming result and experience(12–14). Those conclusions may give insights on the relationship between learner’s behavior and result; however they are not very usable for designing a learning environment to support problem-solving in the process of pair programming, because the studies did not provide specific behavior that resulted in successful problem-solving. Actually few studies are found that focused on the micro interaction between the learners in pair programming with quantitative results, which are useful in designing such an environment.

In this study, pair programming was conducted in an introductory programming course. Programming pairs were videotaped and behavior of some pairs were analyzed regarding the learners’ utterance and the pro-
gramming-related operation of a personal computer. As a result, it was found that operations covered more time in the successful cases. Moreover, the behavior pattern “the operation after dialogue” was more common in the successful cases, in terms of the percentage of all the operations and in terms of frequency per minute.

This knowledge can be directly used in the design of a learning environment for pair programming. We can think of a system that detects good and bad learner’s behaviors for achieving successful results, and intervenes with promotional or inhibitory advice in the process of pair programming. Although we can also think of other systems that use the results of pair programming, and/or opinions from the learners after using pair programming, which do not require knowledge of micro interactions in pair programming, learners meet as many unsuccessful problem-solving cases as when working without such systems.

The rest of the paper is structured as follows. Related works are presented in Section 2. The pair programming practice sessions in which the data were collected are described in Section 3. How the collected video data were processed and analyzed is explained in Section 4. The analysis results are presented in Section 5. After the discussion of the results in Section 6, Section 7 concludes the paper.

2. Related Works

2.1 Solo vs. Pair programming

In the previous research studies which focused on introductory programming courses, it has been proved that pair programming had better outcomes than solo programming. Pair teams were found to usually develop programs and software having higher quality(3–5), and the time spent was shorter than that of individual programmers(4). Programmers who worked in pairs were more self-sufficient, and generally performed significantly better on projects and exams(5–7).

These research studies have shown the efficiency of pair programming, but they did not get into the details of behaviors or the interaction processes. In contrast, our research is on the behavior level. We focus 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 by more researchers. Sfetsos et al.(8) showed that productivity for pairs is positively correlated with communication transactions. Bryant et al.(9) noted that the expertise distribution influenced the pair communication interactions and that the operation behavior assisted intra-pair verbal communication. Chong and Hurlbutt(10) stated 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(11) compared the utterances in both successful and unsuccessful problem-solving cases, and found that unsuccessful cases had longer utterances, more repeated explanations and more examples of continuous speech.

Cooperation plays an important role in many group works in different domains. Cooperative behavior has also been regarded as a key component in pair programming and was analyzed by some researchers. Pollock and Jochen(12) found that the cooperative behavior of pairs in a programming course, such as think-pair-share, group questions and role play, made students work at high efficiency. Gehringer et al.(13) showed that similar cooperative behavior increased retention and boosted the performance of at-risk students. Wei(14) learnt from a students’ survey in a pair programming course that students perceived the cooperative learning method to be effective in teaching programming classes.

Our study, on the other hand, focused on the relationship between the behavioral difference and the results of cooperative problem-solving based on objective data, where the behavior included both utterance and programming operation.

3. Data Collection

The pair programming data were collected from one introductory programming course named “Programming I” in 2010 and 2011, where presenting C language formed the major teaching content. The target of the course was freshmen in the University’s School of Informatics. The aim of the course was to let the students understand the grammar of the programming language and write a program. Each course lecture lasted 75 minutes, among that 30 minutes were allocated for pair programming practice. Thus the pair programming practice was regarded as an integral part of the course.

In each pair programming practice session, a programming exercise was given to the students that em-
employed already taught information. Figure 1 is an example of the exercises.

While programming together, pairs were required to follow these instructions:

- The time limit is 30 minutes. However, it is expected the exercise will be completed as quickly as possible.
- Only the driver can operate the keyboard and mouse. The navigator should observe and support the driver’s work without touching the keyboard or the mouse.
- The textbook can be consulted but not the Web.
- Questions to the instructors are not allowed during the session.

Preparations were done before the data collection, such as deciding the pair combinations and the Driver/Navigator role assignments for each pair, and setting up the video cameras.

Figure 2 shows a camera setup for the data collection. Three cameras from different angles were set for each pair: one for recording the pair’s communication, another for recording the pair’s behavior and work activities such as typing on the keyboard, using the mouse, pointing at the display, and referring to the textbook, and the third for recording the screen. Figure 3 shows example scenes from three cameras.

Three or four pairs in the front row of the classroom were videotaped among about 30 pairs in each practice session. This was because of the number of available recorders. The seats were rearranged so that different pairs were recorded as much as possible. There were 24 sessions in 2010 in which 48 students participated and 31 sessions in 2011 also in which 48 participated were videotaped as a result. However, many of the 2010 video recordings were not qualified to be used for the analysis, because they were only partial sessions (the analysis required pair behavior from the beginning to the end of the session) and they were not clear enough to recognize both conversation and coding. Only 5 sessions with 10 participants could be used from the 2010 videos. For 31 sessions in the 2011 videos, 3 sessions did not include any “case”, which is explained in Section 4.1, and they were eliminated before analysis. For the remaining 28 sessions, 7 sessions had overlaps in the participants although the pairings and the roles were all different. We excluded these so finally 21 sessions with 42 participants were used from the 2011 videos. As a result, 26 sessions with 52 different participants were used for analysis in total.

4. Data Analysis

4.1 Data processing

A pair encounters problems while programming that may or may not be solved successfully. Because we are interested in problem-solving, we regard each problem-solving process as a case, to explicitly indicate it as an analysis target. Descriptions and definitions of a case are as follows:

- A pair programming session can include none, one, two, or more cases.
- A case is a problem-solving process. It begins with the problem encountered, and ends with it either being solved or the time is up.
A problem can be a compilation error that occurs when a pair compiles their program, or a runtime error that occurs when the program is executed.

A case ends in either a successful or an unsuccessful result. The former is called the successful case and the latter, the failed case.

In the successful case the compilation error or runtime error is found and solved within the given limit of 30 minutes.

In the failed case the compilation error or runtime error is found but it is not solved within the given limit of 30 minutes.

From the definitions, the data length (expressed as time in unit of seconds) of a successful case is from finding the problem until solving it, and that of a failed case is from finding the problem to the end of the session, for which the maximum length can be 30 minutes.

The video annotation tool ELAN (EDUICO Linguistic Annotator) was used to synchronize the three videos into one integrated video and to label and annotate behavior on the integrated one. Figure 4 is a screenshot of the tool being used. Among the 26 sessions, 36 cases were found and 27 were successful cases and 9 were failed cases.

4.2 Parameters

As the major observable behavior, utterance and computer operation were labeled first. An utterance is a bit of spoken language. It could be a sentence or a meaningless word such as “Ah!”, “Eh…”, and “Mm…”. An operation includes typing on the keyboard and handling the mouse.

We also investigated the basic patterns that were made up from the combination of utterance and operation, which could be described as “operation after dialogue” or “operation accompanied by dialogue”. The importance of pattern has been considered in interaction research. Because this study also focused on interactions between learners, investigating such basic patterns can be an important step to understand pair programming. This is one of the reasons we focused on these behavior patterns.

How multiple members can work cooperatively in small group collaborative work has been ethnographically studied in investigations of subway operation and emergency medical services. It was found for successful teamwork that the members were close to each other, even information given to other members was shared by overhearing, and conversations for shared understanding were had when necessary during work. A similar study was found in collaborative learning, where the approaches were qualitative. However for our purpose more specific and quantitative descriptions of behavior are required. This is another reason we investigated these behavior patterns.

4.2.1 Utterance

Utterance was analyzed as “Utterance ratio”, “Utterance frequency” and “Utterance length”.

Utterance ratio is the percentage of utterance time in a case. Talking length is the sum of the driver’s and navigator’s talking times. Data length is the length of the analysis target. We could get the Utterance ratio with the following formula as a percentage. Both the driver’s and navigator’s utterances together were used in this definition.

\[
\text{Utterance ratio} \times 100 = \frac{\text{Talking length}}{\text{Data length}}
\]

Utterance frequency is the number of utterances in a minute and the following formula was used to calculate it. Multiplying by 60 is because Data length is given in the unit of seconds.

\[
\text{Utterance frequency (min}^{-1}) = \frac{\text{Utterance numbers}}{\text{Data length}} \times 60
\]

Utterance length is the average length of an utterance.

Figure 4. Screenshot Showing Use of the ELAN Annotation Interface.
Utterance length (s) = Talking length / Utterance numbers

4.2.2 Operation

Operation was analyzed as “Operation ratio”, “Operation frequency”, and “Operation length”.

Operation ratio is the percentage of total operating time in a case, and given by the following formula where Total operation length is the sum of each operation time and Data length is the length of the analysis target.

\[
\text{Operation ratio (\%)} = \frac{\text{Total operation length}}{\text{Data length}} \times 100
\]

Operation frequency is the number of operations in a minute and the following formula was used to calculate it. Operation numbers is the total number of operations in a case.

\[
\text{Operation frequency (min}^{-1}) = \frac{\text{Operation numbers}}{\text{Data length}} \times 60
\]

Operation length is the average length of an operation, and given by the following formula.

\[
\text{Operation length (s)} = \frac{\text{Total operation length}}{\text{Operation numbers}}
\]

4.2.3 Operation after dialogue

One of the basic patterns by the combination of utterance and operation, “operation after dialogue” is illustrated in Figure 5. When the last two utterances before a Driver’s operation are turn-taking utterances by both Driver and Navigator, it is regarded as the “operation after dialogue” pattern.

For this parameter, we analyzed the ratio of operation after dialogue, and the frequency of operation after dialogue. The ratio of operation after dialogue represents the percentage of occurrence of the “operation after dialogue” pattern over the number of all operations.

\[
\text{Ratio of operation after dialogue (\%)} = \frac{\text{Number of operations after dialogue}}{\text{Number of operations}} \times 100
\]

The frequency of operation after dialogue represents the number of operations after dialogue in one minute.

\[
\text{Frequency of operation after dialogue (min}^{-1}) = \frac{\text{Number of operations after dialogue}}{\text{Data length}}
\]

4.2.4 Operation accompanied by dialogue

The other basic pattern by the combination of utterance and operation was the “operation accompanied by dialogue” pattern as shown in Figure 6. In this pattern, Driver and Navigator began the turn-taking utterance, a dialogue, after operation.

For this parameter, we analyzed the ratio of operation accompanied by dialogue and the frequency of operation accompanied by dialogue.

The ratio of operation accompanied by dialogue represents the percentage of occurrence of the “operation accompanied by dialogue” pattern over the number of all operations.

\[
\text{Ratio of operation accompanied by dialogue (\%)} = \frac{\text{Number of operations accompanied by dialogue}}{\text{Number of operations}} \times 100
\]
The frequency of operation accompanied by dialogue represents the number of operations accompanied by dialogue in one minute.

Frequency of operation accompanied by dialogue (min⁻¹) = Number of operations accompanied by dialogue/Data length

5. Results

The values of the parameters described in Section 4 were obtained and 36 cases were analyzed, where 27 were successful cases and 9 were failed cases.

5.1 Results of utterance analysis

The utterance ratio in the successful cases was 18.2% and that in the failed cases was 19.6%. They were not significantly different by the Mann–Whitney $U$ test.

The utterance frequency in the successful cases was 6.04 per minute and that in the failed cases was 5.43 per minute. They were not significantly different by the Mann–Whitney $U$ test.

The utterance length in the successful cases was 1.84 s, while in the failed cases it was 2.08 s. They were not significantly different by the Mann–Whitney $U$ test though utterance length seemed longer in the failed cases.

5.2 Results of operation analysis

Figure 7 shows the operation ratio in the successful cases was 35.3%, while in the failed cases it was 24.6%. They were marginally different ($p=0.09$, $U=75.5$) by the Mann–Whitney $U$ test. The successful cases had a higher operation ratio than the failed cases. That is, the time was spent in carrying out more operations in the successful cases.

The operation frequency in the successful cases was 2.99 per minute, while that in the failed cases was 2.99 per minute. They were not significantly different by the Mann–Whitney $U$ test.

The operation lasted for 8.11 s on average in the successful cases, and 5.57 s in the failed cases. They were not significantly different by the Mann–Whitney $U$ test, though operation seemed longer in the successful cases.

5.3 Results of operation after dialogue analysis

As shown in Figure 8, the ratio of the operations after dialogue in the successful cases was 55.0% and that in the failed cases was 29.9%, which were significantly different ($p<0.001$, $U=50.5$) by the Mann–Whitney $U$ test. The operation after dialogue was more common in the successful cases than in the failed cases.

As shown in Figure 9, the frequency of the operations after dialogue in the successful cases was 1.58 per minute and that in the failed cases was 0.80 per minute, which were significantly different ($p<0.001$, $U=37.0$)
by the Mann–Whitney U test. This indicated that the operation after dialogue happened more often in the successful cases than in the failed cases.

5.4 Results of operation accompanied by dialogue analysis

The ratio of the operations accompanied by dialogue in the successful cases was 17.4% and that in the failed cases was 13.7%, as shown in Figure 10. They were not significantly different by the Mann–Whitney U test.

The frequency of the operations accompanied by dialogue in the successful cases was 0.54 per minute and that in the failed cases was 0.39 per minute, as shown in Figure 11. They were not significantly different by the Mann–Whitney U test.

6. Discussion

6.1 On the results

Among the basic parameters, the operation ratio was higher in the successful cases. According to our observation, students who failed in problem-solving often took other actions such as searching in the textbook or writing things on paper because they needed to search for ideas and solutions to the problem. The students in the successful cases, in contrast, solved the problem smoothly with the knowledge they already had without consulting the textbook or other external knowledge sources. This meant that they could save time by not searching for solutions and they could spend more time for coding, which resulted in the higher operation ratio of the successful cases.

As for the basic patterns by the combination of utterance and operation, the successful cases had a higher ratio and frequency of “operation after dialogue” pattern than the failed cases. In other words, more operations after dialogues were found among all operations in the successful cases, and more operations after dialogues...
were found per minute in the successful cases. From our observation of the data, the dialogue in this pattern was mainly knowledge and opinion exchange between Driver and Navigator, which was part of the cooperative programming work between the pair. As presented in previous research, cooperation was found to be a factor that influences the efficiency in many domains, including programming. These results were in line with the previous research in this sense, but moreover they were quantitatively measured.

Also observed were the scenes in which the Navigator gave suggestions but fundamentally the Driver decided which suggestion to follow. If the Driver did not agree with the suggestion, he would not type the code, or would begin another dialogue about the suggestion. With this, decision in higher quality that was agreed by both would be made. This could be one of the reasons why operation after dialogue was more common in the successful cases.

As for the other basic pattern by the combination of utterance and operation, “operation accompanied by dialogue”, although more operations accompanied by dialogues were found among all operations in the successful cases, and more operations accompanied by dialogues were found per minute in the successful cases, the differences between the successful and the failed cases were not statistically significant.

We observed that students in the successful cases liked to ask their partner about the operation just done, or preferred to explain why the operation was done. There seemed to be more favorable interactions and good understanding between the pairs in the successful cases. Thus it was not surprising that the successful cases had higher ratio and frequency of operation accompanied by dialogue. However, when compared with the “operation after dialogue” pattern, both the ratio and frequency of the “operation accompanied by dialogue” pattern were much smaller.

When we think of the basic patterns by the combination of utterance and operation, there are these two patterns, both of which can be regarded as cooperative programming snippets. The results for those patterns indicated that the “operation after dialogue” pattern was more closely related to the success of pair programming learning.

### 6.2 On the possible overlap of the patterns

We analyzed two behavior patterns in our study, “operation after dialogue” and “operation accompanied by dialogue”. They were two different patterns, but the overlap between them was possible as shown in Figure 12. This kind of overlap could occur when only the outlines of the patterns were considered. From the outlines of the patterns, we could not judge if a dialogue was related to the operation before it or the one after it. To make the pattern analysis accurate, we took the dialogue content into consideration. What the pair programmers said was transcribed and we judged if it should be with the operation before it or the one after it.

### 6.3 On the possible effects of the session time

The maximum practice session time was 30 minutes as noted, and the length of a failed case was limited to the same 30 minutes accordingly because the failed case was defined from the time the problem was found to the end of the session. This time limitation might affect the results. For example, if the operation frequency was continuously increased according to the time in a failed case, the average operation frequency would be higher if the failure case continued for 60 minutes, not 30 minutes. Thus the possible effects of the session time on the significant results were considered.
6.3.1 Operation ratio

Operation ratio was given as the total operation time over total case time. It was lower in the failed cases. When we think about problem-solving activity, it becomes inactive over a long period of time even if the problem is not solved. Therefore even if the session time could be made longer and the time for the failed case could be longer accordingly, it would not be likely that the operation ratio in the failed case would become higher. Thus this result was considered to be robust with longer session time. If the session time could be shorter conversely, this result would be affected. We can easily imagine that no problem can be successfully solved within a very short time. Any case needs to have some time span to allow it to be figured out, whether or not it is finally successful. The minimum time span for this result was not investigated in this study.

6.3.2 Ratio of the operation after dialogue

This ratio is the number of operations after dialogue to the number of all operations. It is not easy to reach a result only by a thinking experiment about what the ratio becomes when the session time is longer. Thus, the ratios in the first half of a case and those in the last half were calculated with all the failed cases. They were 29.0% and 31.8%, respectively as shown in Figure 13, and were not significantly different by the chi-square test ($p=0.26$). Therefore this result was considered to be robust regardless of the session time.

6.3.3 Frequency of the operation after dialogue

This frequency is the number of the operations after dialogue in a minute. The “operation after dialogue” pattern was more frequent in the successful cases. Problem-solving becomes inactive over a long period of time even if the problem is not solved. This means that it would not be likely that the number of operations after dialogue per minute would increase over time. Thus this result was considered to be robust with longer session time. If the session time could be shorter conversely, this result would be affected. The minimum time span for this result was not investigated in this study.

6.4 On the stages of problem-solving

In this study, problem-solving was treated as a case and was analyzed with average values of parameters. For more detailed analysis, the stages of problem-solving can be taken into consideration. In general, the stages of problem-solving have been discussed for a long time. Among them, Dewey’s five steps of problem-solving, Wallas’s four stages of creation, and Polya’s four stages of problem-solving, are well known. Although a clear, common understanding of the stages of collaborative problem-solving has not yet been reached, collaborative problem-solving processes have also been researched. Thus analyzing the pair programming cases as stages may also be useful, and might lead to finer understanding of pair programming learning status. Consequently, it may result in the design of more intelligent learning support environments.

7. Conclusion

In programming education, one of the programming learning methods, pair programming which originated from industry, has been adopted recently. Although the effectiveness of the method has been reported, the learning process associated with it is not clearly understood.

In this study, pair programming process was analyzed in terms of talk and work, mainly in a quantitative manner, which should contribute to the design of collab-
orative learning environments. The pair programming activity was considered as successful or failed cases, and the difference of behaviors for the two categories was investigated. It was found that the successful cases had a higher operation ratio and a higher ratio and frequency of “operation after dialogue” pattern than the failed cases had.

We suggest that closely-knit dialogues and actions can be used as indicators of successful problem-solving, which can be applied to future collaborative learning support systems.

References

Tomoo Inoue received his Ph.D. from Keio University, Japan. He is Associate Professor of the Faculty of Library, Information and Media Science of the University of Tsukuba. His research interests include HCI, CSCW, and technology-enhanced learning. He has authored a number of papers and is a recipient of awards including Best Paper Award, Activity Contribution Award and SIG Research Award from the Information Processing Society of Japan. He has served on a number of academic committees, which currently include IEICE SIG Human Communication Science, IEICE SIG Multimedia on Cooking and Eating Activities, IPSJ SIG Groupware and Network Services, IPSJ SIG Digital Content, VRSJ SIG Cyberspace, IEEE TC CSCWD, and APSCE CUMTEL SIG.