Yang, J. C. et al. (Eds.) (2018). Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education

# Impact of Pair Programming Dynamics and Profiles to Pair Success

**Maureen VILLAMOR[a,b*] and Ma. Mercedes RODRIGO[a]**
[a]*Ateneo de Manila University, Quezon City, Philippines*
[b]*University of Southeastern Philippines, Davao City, Philippines*
*maui@usep.edu.ph

**Abstract:** In this paper, we investigate the convergence and leader-follower patterns of pairs of novice programmers as they traced and debugged fragments of code and the impact of these patterns against the success of programming pairs. We performed a dual eye tracking experiment, recorded their fixations and computed for the recurrence rate and the diagonal recurrence profile using Cross-Recurrence Quantification Analysis (CRQA). Results showed that programming pairs who chat and work on the same program together and converge more often significantly perform better than those who do not. The highly proficient but poorly acquainted pairs, despite of poor convergence, still perform the best. On the other hand, low proficiency pairs that are highly acquainted have the weakest performance even if they converge very well. Findings also revealed that there is a significant difference as to who is leading who. The more successful participants within pairs are leading the less successful participants majority of the time. This study is important because it provides information about the dynamics that may likely occur in a pair programming setup.

**Keywords:** Pair programming, convergence, leader-follower, eye tracking, CRQA

## 1. Introduction

Pair programming is a collaborative work arrangement where two programmers execute different programming activities together. It may be co-located, i.e., programmers share a single screen or may also occur remotely or in spatially distributed mode in which programmers look at the same code but on different screens (Baheti, 2002). The most common representation of the dynamics of pair programming is that of a "driver" and "navigator" roles that describe the division of labor. The driver is the one that does the typing or writes a design, while the navigator is the one who performs the strategic planning and monitoring (Williams & Kessler, 2000).

In pair programming, it is possible for individuals within pairs to disengage from the partnership (Plonka et al., 2012). Some extent is acceptable to speed up problem solving. However, there are episodes of disengagement where students withdraw from their pair programming sessions because they are no longer able to follow their partner's work or contribute to the task, hence losing the expected benefits of pairing. The fluctuations between engagement and disengagement suggest that collaboration is not stable but is rather a series of convergent and divergent phases (Sharma et al., 2012). The pair is said to be converging when the collaborating partners jointly work to understand the code. In this phase, the participants in a pair are focused on the same part of the program in what is considered a "stable" manner. On the other hand, a divergent episode of interaction is when the participants are looking at different parts of the program as they try to build their own understanding. During convergent episodes, the pair is said to be "looking together".

The "togetherness" of the participants, such as during a convergent episode, is often measured using gaze coupling, which refers to moments when the participants are looking at the same target (Richardson & Dale, 2005). The degree of gaze coupling is tantamount to the degree of joint attention, defined as "attending to something together with someone and being aware the both are attending" (Schilbach, 2015). Both entail following the direction of another person's gaze, which is seen as an essential step to establishing strong patterns of social interaction (Moore & Dunham, 2014). Prior findings suggest that gaze coupling reflects tightness of collaboration (Pietinen et al.,

2008). This can be visualized using cross-recurrence plots and quantified in terms of recurrence rate through Cross-Recurrence Quantification Analysis (CRQA) (Marwan et al., 2007).

The "driver" and "navigator" roles in pair programming can be analyzed using the concept of gaze direction, where persons involved in a mutual interaction may hold the roles as either the "leader" or "follower" (Schneider & Pea, 2013). For joint attention to occur there must be a leader directing someone else's gaze towards a target and a follower going along with the gaze cue provided. These profiles of collaboration are interesting because they can tell us about the dynamics of the pair. Knowing who leads and follows gives us an idea on what is causing the pairs to collaborate and what instigated their convergence. For instance, in a pair programming eye tracking study conducted by Villamor and Rodrigo (2017a), it was found that the one that usually gives the gaze cue, and hence the leader, are the low prior knowledge participants because they are typically the first one to ask help from a more skilled partner.

In recent years, dual eye tracking in the context of pair programming has been explored to study joint attention in collaborative learning situations (Pietinen et al., 2008; Schneider et al., 2013). Two eye trackers, for instance, can be synchronized for studying the gaze of two individuals collaborating to solve a problem and for understanding how gaze and speech are coupled (Pietinen et al., 2008). In a study that highlights leader-follower behaviors, "joint attention" is defined as a measure of how many times both participants looked at the same target on the screen where each member within a pair is identified as "leader" and "follower" (Schneider & Pea, 2013). Knowing who leads and follows gives us an idea on what is causing the pairs to collaborate and what instigated their convergence. For instance, in a pair programming eye tracking study by Villamor and Rodrigo (2017a), it was found that the one that usually gives the gaze cue, and hence the leader, are the low prior knowledge participants because they are typically the first one to ask help from a more skilled partner. On the dynamics of convergence, Sharma et al. (2012) investigated the impact of convergent and divergent phases of interaction on the program comprehension strategies of pairs with different levels of understanding where they found that moments of convergence are accompanied by more systematic execution of the code and less transitions among identifiers and expressions.

The goal of this paper is to investigate pair programming profiles and dynamics, particularly convergence and leader-follower patterns, and determine how these dynamics affect the success of the pairs measured in terms of debugging scores. This paper attempts to answer the following:

1. Is there a significant difference on the recurrence rate and debugging scores between successful and unsuccessful programming pairs based on their convergence patterns?
2. Who converge the most and perform the best in terms of proficiency, gender, and acquaintanceship? How does the frequency of convergence affect the debugging scores of the pairs based on these categories?
3. Between a more successful and a less successful participant within a pair, who is the leader and the follower majority of the time? Is there a significant difference as to who is leading who?
4. Is there a significant difference on the recurrence rate and debugging scores between successful and unsuccessful programming pairs based on leader-follower patterns? How do the leader-follower patterns affect the debugging scores?

This study endeavors to contribute to our main goal of understanding better the different dynamics that may take place during pair programming sessions in our attempt to determine the potential indicators that may impact the success in pair programming.

## 2. Methods

### *2.1 Participants and Structure of the Study*

The study was conducted in 6 universities in the Philippines recruiting 2nd to 4th year level college students who had already taken their college-level fundamental programming course. Eighty-four (84) participants, 56 males and 28 females, were randomly paired regardless of gender, proficiency level, and acquaintanceship level resulting in a total of 42 pairs. The task was to locate and mark the errors in the 12 erroneous programs. A chat program was provided to encourage the pairs to

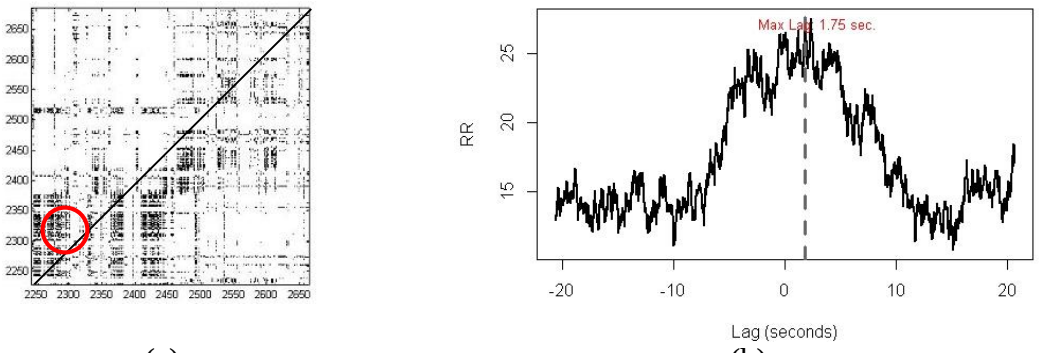
collaborate. We used chat to ensure that the pairs will not be tempted to look away from their screens. The pairs were co-located but they were spaced far enough to ensure that all communications with their partner was via chat only. An artificial limit (e.g., 60 minutes) was set for the debugging tasks to copy a programming laboratory setting where students are given programming exercises to solve or debug within a given time limit.

The result of the program comprehension test was used to divide the participants into high and low proficiency pairs. Pairs whose average score were equal to or above the mean test score were categorized as highly proficient pairs and the rest were tagged as low proficient. The result of the post-test pair evaluation given after the experiment was used to assess the degree of acquaintanceship of the pairs whether they were highly or poorly acquainted. The questionnaire contained items about how well the pairs knew each other, how well they thought they collaborated, and how they felt about their partner.

We recorded a total of 376 cases, where a case is defined as one of the 12 programs under each pair. A pair is successful if the average debugging score for the 12 programs is greater than or equal to the mean score; otherwise, the pair is unsuccessful. There were 19 successful pairs and 22 unsuccessful pairs. A case is successful when both participants within a pair are able to mark half of the bugs in a program. Otherwise, if only one participant marks at least 50% of the bugs or both fail to spot the bugs, then the case is unsuccessful. There were 196 successful cases and 180 unsuccessful cases. The debugging scores were converted to percentage equivalents. To test for statistically significant differences, *t*-test for independent sample means and ANOVA were performed. Pearson's correlation was carried out to determine the relationships between the convergence and leader-follower patterns vis-a-vis the pairs' debugging scores. For a detailed description of the structure of the study, data cleaning, and preparation for analysis, see Villamor and Rodrigo (2018).

## *2.2 Cross-Recurrence Plot and Diagonal Recurrence Profile*

A cross-recurrence plot (CRP) is an *N* x *N* matrix, which represents a time coupling between two time series. The horizontal axis represents time for the first collaborator (C1) and the vertical axis represents time for the second collaborator (C2). Recurrence occurs when two fixations from different sequences land within a given threshold of each other using some distance metric. If fixations *i* and *j* are recurrent, they are represented as a black point (pixel) in the plot (see Figure 1.a). Hence, a point in the plot indicates that the fixations from two different collaborators at their respective times are recurrent. If two collaborators uninterruptedly looked at two different spots on the screen for the entire interaction, the resulting CRP would be completely blank (white space in Figure 1.a). If the two collaborators looked at the same spot on the screen continuously, the plot would show a dark line parallel to the main diagonal. Each diagonal on a CRP parallel to the main diagonal corresponds to a particular alignment between the collaborators' eye movement data with a particular lag time between them. Points exactly on the main diagonal of the plot correspond to synchronous recurrence, such as, collaborators look at the same target at exactly the same time.




(a) (b)

*Figure 1.* (a) Cross-Recurrence Plot and (b) Diagonal-Recurrence Profile Visualization.

A number of measures can be extracted from a CRP. These include recurrence rate (*RR*), determinism (*DET*), average diagonal length (*L*), longest diagonal length (*LMAX*), entropy (*ENTR*), laminarity (*LAM*), and trapping time (*TT*). Cross-Recurrence Rate (*RR*) represents the "raw" amount

of similarities between the trajectories of the two systems, which refers to the degree to which they tend to visit similar state. In eye tracking, this represents the percentage of cross-recurrent fixations that is indicative of the degree of gaze coupling. The definitions of the other measures can be found in (Marwan et al., 2007). This study limits its analysis to include only *RR* as prior literature have used this measure to account for the "togetherness" of two people in terms of where they are looking. Hence, we will also use *RR* to assess the degree of gaze coupling of the programming pairs when they converge.

Analysis on leader-follower trends is focused on the diagonal recurrence profile (*DiagProfile*), which is the diagonal on the CRP that runs from the origin up to the topmost right portion. The *DiagProfile* can be used to analyze which delay or lag maximizes recurrence and to observe the direction of the coordination (Fusaroli et al., 2014). When the recurrence rate is largely distributed above or below the main diagonal of the plot, it has direct bearing on the leading/following patterns of the systems that produced those time series. Given two collaborators C1 and C2, points above the diagonal correspond to fixations of C2 that happen after C1 has fixated the element. Points below the diagonal correspond to C2's gaze leading C1's. Asymmetries above and below the diagonal line could therefore be indicative of leading and following behaviors.

Figure 1.b shows a visualization of the diagonal recurrence profile of the CRP on the left, which was one of the CRPs generated in this experiment. The *DiagProfile* visualization was done in *R* using the function *drpdfromts* from the *crqa* package of Coco and Dale (2014). This function returned a recurrence profile with the length equal to the number of lags considered, the maximal recurrence observed between the two fixation sequences, and the lag at which it occurred. In this study, we focused only on the lag, which informs how much time one participant is ahead of the other. The window size or the argument '*ws*' of this function was set equal to the length of the fixation sequence and gave a span of +/−3 seconds where each time-point represented a fixation of 33 msec. long. Figure 1.b indicates that C1 is ahead of C2 by 1.75 seconds. If C2 is leading C1, the lag becomes negative. If the pair is perfectly aligned, i.e., both participants look at the same spot at the same time, the lag is zero.

### *2.3 A Leader-Follower Example*

Richardson and Dale (2005), in their collaborative eye tracking study, investigated how quickly a test participant fixates on a target after it is mentioned by the partner. This measure indicates how well the listener understood what the partner said. Fixating on a target after it has been referenced by a partner is also another way to tell whether a leader-follower pattern occurs. To illustrate an example how a leader-follower scenario unfolds in this pair programming eye tracking study, snapshots of the program used as a stimulus showing actual communication between collaborating partners and their fixation points are shown in Figure 2.

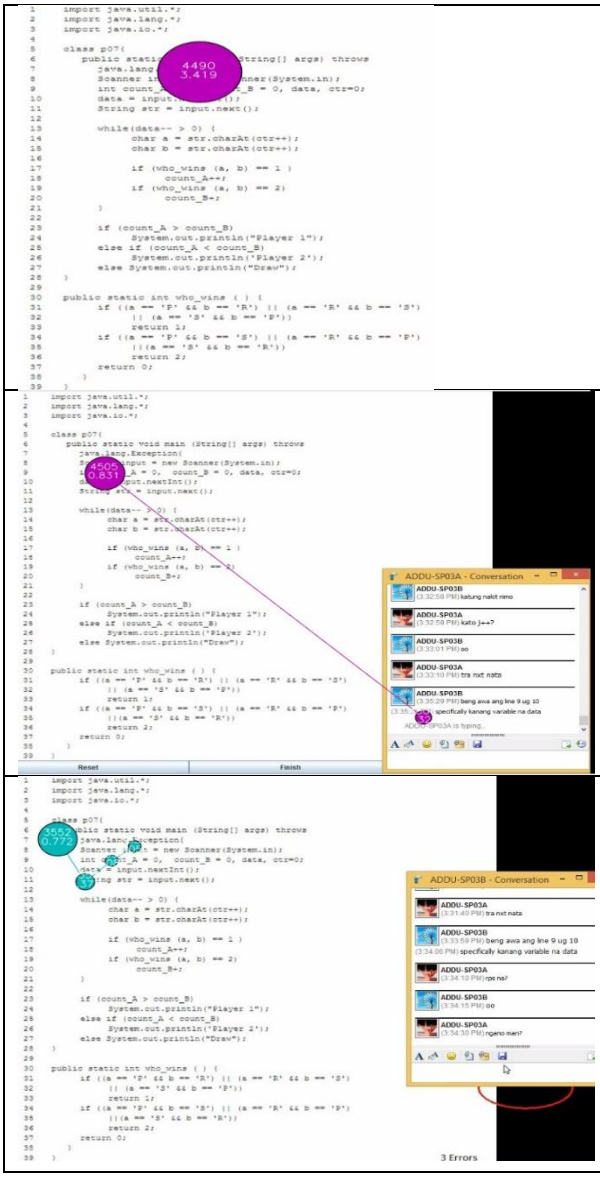
From the snapshots, C1 is considered the "leader" in this scenario while C2 is the "follower". After C1 had fixated long enough on a specific location of the program, C1 initiated a chat with C2 and told C2 to look at lines 9 and 10. C2 then responded to confirm if C1 was referring to the Rock-Scissor-Paper (RPS) program, looked at lines 9 and 10, and asked C1 about what seemed to be the problem. What is the effect after both collaborators have looked at the same location in the program? Since the fixation points of C1 and C2 were positioned at about the same location when they were looking at the program, this made their fixations recurrent based on a set threshold. In Figure 1.a, part of the pixelated regions enclosed in a red circle on the CRP informs us that the fixation points of the two collaborators with respect to the times these fixations occurred are recurrent. Hence, the more leader-follower dynamics occur, the more cross-recurrence fixations will there be causing an increase in the recurrence rate and an increased recurrence rate is an indication of better collaboration.

## 3. Results and Discussion

### *3.1 Convergence Patterns*

In this study, we define convergence in three ways: 1) when the participants in a pair are working on the same program together, either deliberately or not; 2) when they communicated via chat to

talk about the task even if they are not looking at the same program (e.g., C1 might be working already on program 6, while C2 is still on program 5 but they are talking about the errors in program 5); and 3) when they are communicating and working on the same program together, hence, there is mutual awareness.



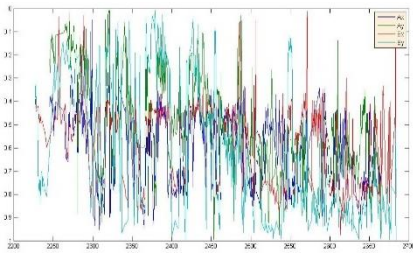
| | |
|---|---|
| | First Scenario:<br><br>Collaborator 1 (C1) looked at a specific target on the program for about 3.4 seconds. |
| | Second Scenario:<br><br>C1 opened the chat program and told his partner to look at lines 9 and 10, specifically the variable "data". |
| | Third Scenario:<br><br>Collaborator 2 (C2) responded to C1 and then looked at lines 9 and 10, as mentioned by C1. |

*Figure 2.* An Illustration of a Leader-Follower Scenario.

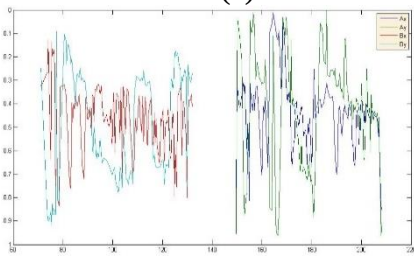
To verify whether the individuals within the pair are working on the same program together, a plot was generated for each pair that showed the scan patterns of the pairs using a line graph. Figure 3 shows an example of these plots. The plot illustrates the fixation *x*- and *y*-coordinates of the two collaborators. The blue and green lines are the fixation *x*- and *y*-coordinates of C1, while the red and aqua lines are the fixation *x*- and *y*-coordinates of C2. The *x*-axis of the plot represents the combined timeline of the two collaborators, and the *y*-axis represents the range of possible values of the fixation *x*- and *y*-coordinates, which is between 0 and 1 (reversed). In Figure 3.a, we can see that all the lines overlap, which means that the pair that this plot represents was working on the same program together. The plot in Figure 3.b has no overlap or there is a break in between, which means that the pair here was working on the same program but at different times, that is, C2 worked on the program ahead of C1.

The chat log of each pair was examined to confirm whether the pair chatted in every program. The number of times they converged via chat was recorded per program. Figure 3.c shows an actual transcript of one of the pairs in this study. Collaborators *A* and *B* were currently working in program 5. Collaborator *B* called the attention of *A*, *A* responded, and then both proceeded to check on

program 4. After some time, *B* asked *A* again if *A* was already working on a program about parenthesis matching. At this point, both were back to program 5. The pair in this example converged via chat twice while working on program 5.



(a)



(b)

| PERSON | TIME | PROG. NO. | TEXT |
|---|---|---|---|
| B | 1:59:25 PM | 5 | There's should be no := |
| A | 1:59:25 PM | 5 | i:=0 |
| B | 1:59:32 PM | | Then you have to be cautious on the last part |
| A | 1:59:38 PM | 4 | Okay |
| B | 1:59:45 PM | 4 | s(i) is supposed to be s[1] |
| A | 1:59:52 PM | | Yes. I saw that one. |
| B | 1:59:58 PM | | Did you see the 3 errors in your lower right, Jo? |
| B | 2:00:05 PM | | That's the indication as to how many errors are thei |
| A | 2:00:36 PM | | Ah okay. |
| B | 2:02:00 PM | 5 | Are you already in the part where there's MATCH? |
| B | 2:03:41 PM | | 8-\| |
| A | 2:03:52 PM | 5 | Yes. I just saw one, still. |
| B | 2:03:57 PM | | What is it? |
| A | 2:04:10 PM | | The 1st "if" |
| A | 2:04:14 PM | | It should be == |
| B | 2:04:17 PM | | Yes. You're right. |
| B | 2:04:25 PM | | I think, the next error is the system.exit |
| B | 2:04:28 PM | | I quite don't know actually about it. |
| B | 2:04:37 PM | | coz at the part where there is MATCH in an "if", I think there's no error in there. |
| A | 2:05:47 PM | | But I think the program is right, coz it'll do exit if it already less than zero |

(c)

*Figure 3.* Line Graphs showing (a) overlapped scan patterns (working on the same program together), (b) scan patterns with no overlap (not working together), and (c) actual chat transcript[1].

To answer the first research question, we recorded the distribution of the number of successful and unsuccessful cases under the three convergence types, which is found in Table 1. The average recurrence rates (*RR*) and debugging scores under the three convergence types are found in Table 2. The average *RRs* of the pairs that worked together on the same program and those that did not work together were comparable and the difference in *RR* was not significant. This could be explained by the element of chance, which means that even if the pairs are not looking or working on the same program together, it is possible that they might have fixated on the same parts of the program but only at different times. Performance-wise, pairs who worked on the same program together scored significantly higher than pairs who did not work together at ($t = -2.754$, $p = 0.006$).

Table 1

*Distribution of Successful and Unsuccessful Cases in the Three Convergent Types*

| Convergence Types | N = 376 | Successful (N) = 196 | Unsuccessful (N) = 180 |
|---|---|---|---|
| Work together | 269 | 161 | 108 |
| Did not work together | 107 | 35 | 72 |
| Chatted | 203 | 126 | 77 |
| Did not chat | 173 | 70 | 103 |
| Both worked together and chatted | 194 | 119 | 75 |
| Neither worked together nor chatted | 182 | 77 | 105 |

Surprisingly, the average *RR* of those who chatted was significantly lower ($t = 3.53$, $p = 0.000$) than those who did not chat. A possible explanation is that majority of the cases where pairs chatted were successful and majority of the cases where pairs did not chat were unsuccessful (see Table 1). In a parallel study that we conducted, we found that the successful cases are characterized as having more incidences of "low *RR*" because of low fixation counts and less similar scan patterns, which both reduced the likelihood of having more recurrent fixations. This also affirms previous findings that pairs who communicated better but have low degree of gaze coupling suggests that these pairs do not have the explicit aim to coordinate their gazes. Instead, the gaze patterns become

[1] The original transcript was in Bisaya (Cebuano) and translated to English.

aligned due to the joint activity of conversation and commonalities between the processes of speech comprehension and production (Richardson, Dale & Tomlinson, 2009).

On the other hand, the unsuccessful cases, which comprised many of the pairs who did not chat, are characterized as having more occurrences of "high *RR*" because of their high fixation counts, more pronounced similarities in their scan patterns, and fixation cluster patterns which all contributed to an increase in *RR*. Pairs with a high degree of gaze coupling but did not chat could just be a result of an unintentional gaze coordination, as opposed to a gaze coordination that is generated by conversational processes. In terms of performance, the average debugging score of those cases where pairs chatted was higher than those cases where pairs did not chat but the difference in their debugging scores was not significant. However, the 126 successful cases where pairs chatted scored significantly higher than the 103 unsuccessful cases where pairs did not chat ($M_{SUCCESSFUL}$ = 90.58, $SD_{SUCCESSFUL}$ = 8.08 / $M_{UNSUCCESSFUL}$ = 71.29, $SD_{UNSUCCESSFUL}$ = 12.91, $t$ = −13.331, $p$ = 0.000).

Table 2

*Recurrence Rates and Debugging Scores under the Three Convergence Types*

| Convergence Types | Recurrence Rate | | Debugging Score | |
|---|---|---|---|---|
| | Mean | Std. Dev. | Mean | Std. Dev. |
| Work together | 0.40 | 0.14 | 81.34 | 14.52 |
| Did not work together | 0.41 | 0.13 | 76.83 | 13.77 |
| Chatted | 0.38 | 0.11 | 81.14 | 14.93 |
| Did not chat | 0.43 | 0.15 | 78.78 | 13.77 |
| Both worked together and chatted | 0.39 | 0.11 | 80.95 | 15.12 |
| Neither worked together nor chatted | 0.42 | 0.15 | 79.10 | 13.63 |

Considering the number of times the pairs converged via chat (M = 1.16, SD = 1.33, min = 0, max = 7), we found that the average frequency of convergence via chat in successful cases was higher than in unsuccessful cases at ($M_{SUCCESSFUL}$ = 1.38, $SD_{SUCCESSFUL}$ = 1.36) and ($M_{UNSUCCESSFUL}$ = 0.93, $SD_{UNSUCCESSFUL}$ = 1.26). The difference in the number of times they converged was significant at ($t$=−3.346, $p$ = 0.001). Since 125 out of 196 successful cases were from the 19 successful pairs, 112 of the 180 unsuccessful cases were from the 22 unsuccessful pairs, this implies that the successful pairs converged more via chat than the unsuccessful pairs.

No significant relationship existed between the frequency of convergence and the debugging scores of all the 203 cases where pairs chatted and the 70 successful cases where pairs did not chat. However, a significant low negative relationship was found between the frequency of convergence and the debugging scores of the 103 unsuccessful cases where pairs did not chat at ($r$ = −0.202**, $p$ = 0.007). This suggests that the more the unsuccessful pairs converge via chat, the more time they spend in a program. More time spent in a program could mean lesser bugs found because of time constraints imposed during the experiment, i.e., they need to find all the bugs in the 12 programs within an hour. This affirmed the statement of Rummel et al. (2011) saying that too much coordinative dialogue reduces the time available for the task itself.

For the third type of convergence, the average *RR* of cases where pairs worked together on the same program and chatted was significantly lower ($t$ = 2.515, $p$ = 0.012) than those cases where pairs neither worked together nor chatted. This is also because majority of the cases where pairs worked together on the same program and chatted were successful and majority of the cases where pairs neither worked together nor chatted were unsuccessful. As previously mentioned, successful and unsuccessful cases are characterized as having more "low *RR*" and "high *RR*", respectively.

The average debugging scores of those cases where pairs worked together on the same program and chatted was just slightly higher than those cases where pairs neither worked together nor chatted and the difference was not significant. However, the 119 successful cases where pairs worked together and chatted scored significantly higher (M = 90.93, SD = 7.98) than the unsuccessful cases where pairs neither worked together nor chatted (M = 72.22, SD = 13.08) at ($t$ = −13.083, $p$ = 0.000).

Given the pair profiles such as proficiency, gender and acquaintanceship, who converged more via chat? Who performed the best? What is the relationship between the frequency of convergence and debugging scores based on these profile categories? The distribution of these categories including the mean and standard deviation are found in Table 3. We will refer to both highly proficient pairs, both low proficiency pairs, and mixed proficiency pairs as HH, LL, and HL, respectively. For the gender category, we will refer to both males, both females, and mixed gender pairs as MM, FF, and MF, respectively. For the acquaintanceship category, we will refer to poorly acquainted and highly acquainted pairs as PA and HA, respectively.

Findings revealed that HL pairs significantly converged via chat the most while LL pairs converged the least ($F = 3.326$, $p = 0.037$). Based on gender, MM pairs chatted the most while FF pairs chatted the least and the difference was significant ($F = 6.529$, $p = 0.002$). Lastly, in terms of acquaintanceship, HA pairs significantly converged more than PA pairs at ($t = -11.848$, $p = 0.007$). No correlation, however, was found between the frequency of convergence and debugging scores based on these profile categories.

Table 3

*Descriptive Values of the Frequency of Convergence based on Proficiency, Gender, and Acquaintanceship*

| | Proficiency | | | Gender | | | Acquaintanceship | |
|---|---|---|---|---|---|---|---|---|
| | HH | LL | HL | MM | FF | MF | PA | HA |
| N | 124 | 100 | 152 | 167 | 46 | 163 | 129 | 247 |
| Mean | 1.19 | 0.89 | 1.33 | 1.43 | 0.80 | 0.99 | 0.20 | 1.67 |
| Std. Dev. | 1.28 | 1.22 | 1.43 | 1.42 | 1.07 | 1.27 | 0.59 | 1.34 |

We combined the three profile categories to determine which combination of these categories converged the most and the least and see how their frequencies of convergence via chat influence their debugging scores. From all the possible combinations of {(HH, LL, HL) x (MM, FF, MF) x (HA, PA)}, there were no HH-FF-HA, HL-FF-HA, and HL-FF-PA pairs. The differences in frequency of convergence as well as debugging scores among these combinations were significant at ($F = 12.154$, $p = 0.000$) and ($F = 4.077$, $p = 0.000$), respectively.

Among the combinations, none of the HH-MF-PA, LL-MM-PA, LL-MF-PA, and HL-MF-PA pairs chatted. The one thing that these combinations have in common is that these are all poorly acquainted pairs, suggesting that pairs consisting of non-friends are not really disposed to communicating or initiating contact with their partners. However, the HH-MF-PA pairs, despite of not having communicated at all, still performed second best. This was also true to the HH-FF-PA pairs, which performed the best despite of being among the bottom five in terms of convergence (including those four who did not chat). These two combinations consist of highly proficient and poorly acquainted pairs. This affirms the findings of Shah and Jehn (1993) that groups composed of skilled strangers will perform best because they already know from experience how to adapt well with other experts and they are more adaptable to the actions of their group mates. This result also confirms our findings from our previous study using a smaller subset (Villamor & Rodrigo, 2017).

We also found that the top two in terms of frequency of chat convergence, which were the LL-FF-HA and LL-MM-HA pairs, were in the bottom five in terms of performance. These combinations consist of highly acquainted pairs, which corroborates the findings of Shah and Jehn (1993) that friendship may diminish performance because they have the tendency to lose focus. As observed, these two combinations are both low proficient and have the same gender. This could mean that pairs with the same proficiency level and gender are more likely to communicate or collaborate.

Combinations of LL-FF-PA, HH-MM-PA, HH-FF-PA, and LL-MM-PA pairs were all in the bottom half in terms of convergence. This implies that poorly acquainted pairs that have the same proficiency level and gender are less likely to chat. As to whether the pairs perform better or not, this is usually dictated by the proficiency level of the pairs regardless of the gender mix and acquaintanceship. The top four in terms of performance were all highly proficient pairs and all, except one, in the bottom six consisted of low proficiency pairs.

### *3.2 Leader-Follower Patterns*

In this experiment, we focused only on the lag times that reveal how much time one participant is ahead of the other. The number of positive (+) and negative (−) values of the lags were recorded. The signs indicate the gaze direction or who is leading/following. A positive and negative sign indicate that the collaborator along the *x*-axis and *y*-axis of the CRP, respectively, is leading. We defined a "more successful participant" within a pair as one that scored higher or found more bugs while a "less successful participant" is one that has found lesser bugs. We refer to these two types of participants within the pair as MSP and LSP, respectively. We included in this analysis only those cases where the pairs converged both by working together on the same program and communicated via chat and excluded those cases where both participants had the same debugging scores.

For consistency, the pair ordering was set with the MSP first and followed by the LSP. One hundred seventy-eight (178) cases were analyzed, but one of these cases had exactly zero lag time. Hence, it was reduced to 177 where 90 and 87 of these cases were led by the MSPs and LSPs, respectively. Findings showed that the MSPs were leading by an average of 2.63 seconds 50.85% of the time, and the LSPs were ahead of the MSPs by an average of 1.66 seconds 49.15% of the time. *T*-test results showed a significant difference as to who was leading who ($t = 2.320$, $p = 0.021$). The lag times of those cases where the MSPs were leading the LSPs consisted of more "long lag times" (long = 31%, short = 21%), whereas the lag times of those where the LSPs were leading the MSPs comprised of more "short lag times" (long = 18%, short = 26%). A lag time is considered long if it is equal to or greater than the 75$^{th}$ percentile, and short if it is equal to or less than the 25$^{th}$ percentile of the data.

Of the 90 cases where the MSPs were leading the LSPs, 55 of these cases had LSPs that were low proficient. This could explain as to why the lag times were longer when the MSPs were leading the LSPs. It could have taken longer for the low proficient LSPs to find the target and then fixate on it after it has been mentioned by the MSPs. On the other hand, of the 87 cases where the LSPs were ahead of MSPs, 66 of these cases had MSPs that were highly proficient. This could account for the shorter lag times when the LSPs were leading the MSPs. It could be that the highly proficient MSPs are quick to find on the program what the LSPs are referring to.

In terms of debugging scores, the average debugging score of those cases where the LSPs were ahead of the MSPs (M = 82.98, SD = 14.79) were slightly higher than those cases where the MSPs were ahead of the LSPs (M = 80.26, SD = 14.98). The difference in debugging scores, however, was not significant. No relationship was found between the leader-follower lag times and the debugging scores of the pairs. The average *RR* of those cases where MSPs were leading the LSPs and vice versa were comparable at ($M_{MSP\text{-}LSP} = 0.39$, $SD_{MSP\text{-}LSP} = 0.11$) and ($M_{LSP\text{-}MSP} = 0.40$, $SD_{LSP\text{-}MSP} = 0.12$). The difference in *RR* was also not significant. The relationship between *RR* and the leader-follower lag times likewise did not exist.

## 4. Summary, Implication, and Future Work

This paper investigated the convergence and leader-follower patterns as well as the collaborator profiles and how these influenced success in pair programming. In summary, pairs working together on the same program and who communicated via chat scored better than those who did not. Successful pairs also converged more via chat than the unsuccessful pairs. On collaboration based on *RR*, pairs who worked together on the same program and chatted had low *RR* on the average than those who did not. This is because majority of those cases who worked together and chatted were successful, which were characterized as having more "low *RR*". Poorly acquainted pairs with the same proficiency level and gender were less likely to converge. As to whether pairs performed better or not, this was usually dictated by the proficiency level of the pairs regardless of gender mix and degree of acquaintanceship. Findings also revealed that there was a significant difference as to who was leading who. The more successful participants within a pair were leading the less successful participants majority of the time.

This study overlays the foundation to understand more the dynamics that take place in a pair programming setup. Results of this study and similar studies in the future can help

programming professors implement pair programming effectively through proper pair matching that will elicit productive dialogues as well as design remediation that can help their students learn programming much easier with the help of a more suitable partner. Researchers can investigate further about the underlying reasons why one would initiate contact over the other and the ways where joint attention can be increased and then assess how these affect the success of the programming pairs.

Our next step would be to look at the nature of the discourse to see what really conspired when the pairs chatted. It could be possible that those who chatted were successful is because one just told the other the answer. We would also assess the quality of the discourse to properly assess its collaboration aspects and see how it impacts the success of the pairs. It is also interesting to explore the divergent episodes to see how much independent work has been done by the participants in the pair and how it influences pair success.

## References


Baheti, P. (2002). Assessing distributed pair programming. In *Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 50-51). ACM.

Coco, M. I., & Dale, R. (2013). Cross-recurrence quantification analysis of categorical and continuous time series: an R package. *arXiv preprint arXiv:1310.0201*.

Fusaroli, R., Konvalinka, I., & Wallot, S. (2014). Analyzing social interactions: the promises and challenges of using cross recurrence quantification analysis. In *Translational recurrences* (pp. 137-155). Springer International Publishing.

Marwan, N., Romano, M. C., Thiel, M., & Kurths, J. (2007). Recurrence plots for the analysis of complex systems. *Physics reports*, *438*(5-6), 237-329.

Moore, C., & Dunham, P. (2014). *Joint attention: Its origins and role in development*. Psychology Press.

Pietinen, S., Bednarik, R., Glotova, T., Tenhunen, V., & Tukiainen, M. (2008, March). A method to study visual attention aspects of collaboration: eye-tracking pair programmers simultaneously. In *Proceedings of the 2008 symposium on Eye tracking research & applications* (pp. 39-42). ACM.

Plonka, L., Sharp, H., & van der Linden, J. (2012, June). Disengagement in pair programming: does it matter?. In *Software Engineering (ICSE), 2012 34th International Conference on* (pp. 496-506). IEEE.

Richardson, D. C., & Dale, R. (2005). Looking to understand: The coupling between speakers' and listeners' eye movements and its relationship to discourse comprehension. *Cognitive science*, *29*(6), 1045-1060.

Richardson, D. C., Dale, R., & Tomlinson, J. M. (2009). Conversation, gaze coordination, and beliefs about visual context. *Cognitive Science*, *33*(8), 1468-1482.

Rummel, N., Deiglmayr, A., Spada, H., Kahrimanis, G., & Avouris, N. (2011). Analyzing collaborative interactions across domains and settings: An adaptable rating scheme. In *Analyzing interactions in CSCL* (pp. 367-390). Springer, Boston, MA.

Schilbach, L. (2015). Eye to eye, face to face and brain to brain: novel approaches to study the behavioral dynamics and neural mechanisms of social interactions. *Current Opinion in Behavioral Sciences*, *3*, 130-135.

Schneider, B., & Pea, R. (2013). Real-time mutual gaze perception enhances collaborative learning and collaboration quality. *International Journal of Computer-supported collaborative learning*, *8*(4), 375-397.

Shah, P. P., & Jehn, K. A. (1993). Do friends perform better than acquaintances? The interaction of friendship, conflict, and task. *Group decision and negotiation*, *2*(2), 149-165.

Sharma, K., Jermann, P., Nüssli, M. A., & Dillenbourg, P. (2012). Gaze Evidence for different activities in program understanding. In *Proceedings of 24th Annual conference of Psychology of Programming Interest Group* (No. EPFL-CONF-184006).

Villamor, M. and Rodrigo, M. M. (2017a) Exploring Lag Times in a Pair Program Tracing and Debugging Eye-Tracking Experiment. In *Proceedings of the 25th International Conference in Computers in Education* (pp. 234-236).

Villamor, M. and Rodrigo, M. M. (2017b) Impact of Both Prior Knowledge and Acquaintanceship on Collaboration and Performance: A Pair Program Tracing and Debugging Eye-Tracking Experiment. In *Proceedings of the 25th International Conference in Computers in Education* (pp. 182-187).

Villamor, M. and Rodrigo, M. M. (2018). Predicting Successful Collaboration in a Pair Programming Eye Tracking Experiment. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. ACM.

Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, *43*(5), 108-114.