Transactions in Learning Dyads in a Computer Programming Class

Anthony Joseph C. Ocampo

Two learning dyads were observed and interviewed as they went about trying to solve a programming problem supplied by their teacher. The two dyads were differentiated based on their programming ability – one dyad had equally high performing members while the other dyad had equally low performing members. Differences were noted between the two dyads regarding control and conflict resolution. One member of the low performing dyad was found to monopolize control over their computer. The same member also evaluated the suggestions made by the other member without empirically testing the suggestion. In the high performing dyad, both members were observed to have equal control over their computer. The same dyad was also observed to empirically test suggestions made by either member. Results tend to indicate that dyad composition may be related to performance. A quantitative study is recommended to verify this claim.

INTRODUCTION

Computer literacy has become such a necessity in the last two decades that Information Communication Technology (ICT) has found its way into the basic education curriculum, even at the primary levels in some private schools. Unfortunately, the public school system in the Philippines is having a hard time catching up with its private counterparts – it only introduces ICT in the secondary level and cannot meet the 1:1 computer to student ratio provided in private schools. Although the cost of a computer relative to its speed has gone down dramatically over the years, it is still not possible for the Philippine government to provide a 1:1 computer to student ratio in its schools. This forces public school teachers to assign at least two students to a computer.

Fortunately, this situation may be advantageous to the public school system. Researches have shown that cooperative learning in computer classes enhances performance more than standard instruction.

In a meta-analysis of researches on the use of cooperative learning in post secondary science, math, engineering and technology (SMET) courses, Donovan (1997) found that students who learn in small groups had greater academic achievement and more favorable attitudes towards learning than their traditionally taught counterparts.

These results were also noted in researches done in purely computer mediated environments (in computer classes). Tsai, Bethel and Hunstberger (1999) found that the male respondents felt more comfortable and were likely to perform better in a cooperative learning environment than in an environment where students were contextually taught learning strategies. This gender-dependent result was attributed to the culture of the respondents (Taiwanese) which expected females to be passive. Likewise, McInerny, McInerny and Marsh (1997) found that when prior computer competence was controlled, the cooperative strategic group outperformed or were better than the direct instruction group with regards to achievement, self-concept and sense of control.

Purpose and Methods

The researches cited dealt with the effectiveness of cooperative learning as a teaching method. These researches, however, did not show how learning in a group in a computer class occurs. The "how" was simply supported by theory or sometimes by experiences in non-computer mediated situations. Thus this study aimed to 1) identify the transactions in cooperative learning groups/pairs (learning dyads) in a computer programming class and subsequently, 2) ascertain how these transactions are beneficial or detrimental to learning.

Transactions were verbal and non-verbal negotiations (e.g. conflict resolution) and exchanges of ideas and information between members of a learning dyad. These transactions also included the interaction between members of the learning dyads with their computer which was mostly non-verbal.

In order to fulfill these objectives, the researcher employed qualitative methods. Two learning dyads were observed and videotaped as they went about trying to solve a programming problem supplied by their teacher on two separate class sessions (1 hour each). The following aspects were noted during the observations: 1) how the dyads shared ideas; 2) how the dyads resolved conflicts in ideas; and 3) how the dyads negotiated control of computer input.

The dyads consisted of four female 4th year high school students enrolled in a computer programming class at the University of the Philippines Integrated School during the first semester of AY 2002-2003. The observations were done during the second quarter. Post observation interviews (30 minutes per dyad member) were also done to supplement the observations.

In order to ascertain whether a form of transaction was beneficial or detrimental to learning, these two dyads were differentiated using programming ability which was based on their first quarter performance in the periodic test of the said subject. Periodic test score was chosen over actual grade in the first quarter to represent programming ability since the periodic test can be considered a less subjective assessment of student performance. One dyad had equally high achieving members (with a combined average of 82%) and the other had equally low-achieving members (with a combined average of 61.5%). Initially, a third dyad was identified (one with mixed prior achievement). However, during the course of the observations, one member was found to be working separately on another computer.

Differences in transactions between the two groups can thus be attributed to programming ability. It was assumed that these transactions could also be observed in the first quarter.

For future reference, members of the high-achieving dyads will be identified as S11 and S12 and the members of the low-achieving dyads will be identified as S21 and S22. Table 1 shows the characteristics of these individuals.

Table 1. Characteristics of the Selected Dyads

Dyad	Members	1st Quarter Periodic Test Score (%)	1st Quarter Grade (%)
1	S11	81	95
	S12	83	95
2	S21	66	75
	S22	57	71

To prevent bias, the researcher did not have any knowledge of the dyads' prior achievement in the subject during the observation period, and the dyads were selected for the researcher by the subject teacher. Furthermore, unintended effects of observation like the presence of a video recorder were mitigated by stationing the video recorder and the researcher in the classroom for two classroom sessions prior to the actual observation period.

Unfortunately, gender and access to a computer at home could not be controlled in the selection process since the teacher had allowed the students to form their own dyads at the start of the course. The teacher later informed the researcher that only females formed dyads and that the males preferred to work alone except for one pair which had one member who was always absent.

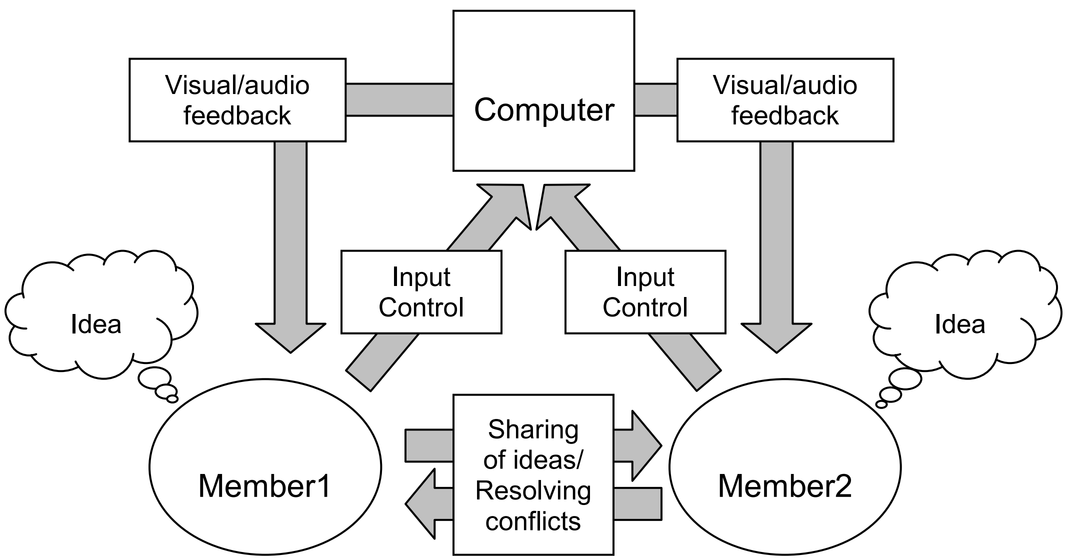
Another variable which could not be controlled was the programming problem the dyads underwent. Since the researcher could not observe both dyads at the same time, the dyads were observed while solving different programming problems. Fortunately, the programming problems were related. Dyad 1 was observed while attempting to construct a quiz program that consisted of open-ended questions. On the other hand, Dyad 2 was observed while attempting to construct a quiz program that consisted of multiple-choice questions.

DISCUSSION OF RESULTS

Information and Control

Data gathered revealed a general pattern of information flow between the members of the dyad and between individual members and the computer. Another pattern was also noted regarding the control one or both members of the dyad had over computer input (via mouse and/or keyboard). Figure 1 below combines both in one model.

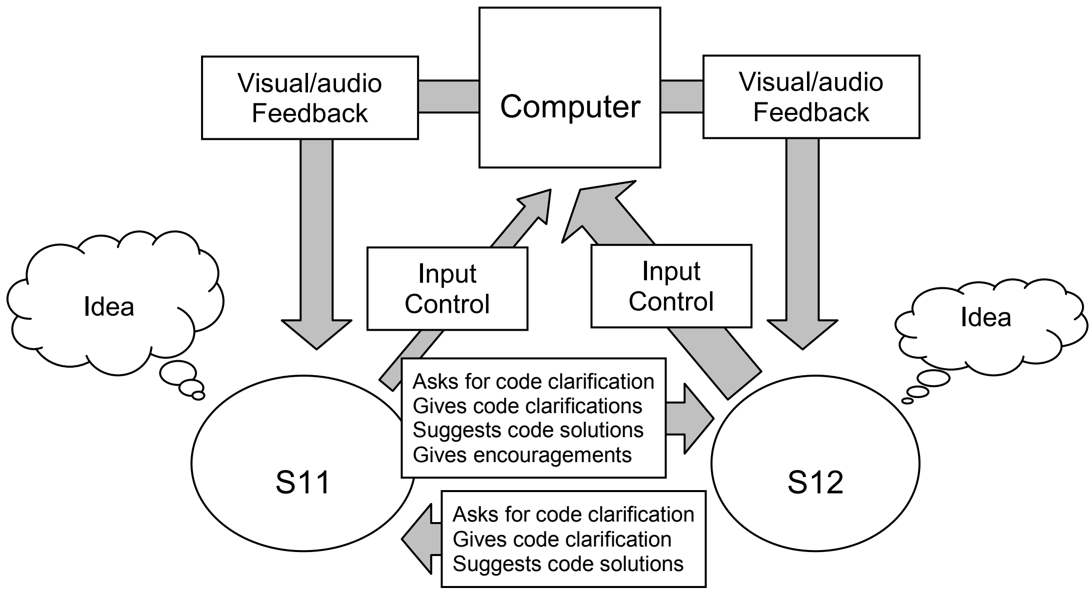
Fig. 1. General Information and Control Model



The arrows between the members of the dyads typically reflect the nature of their conversations during work (i.e. sharing of ideas and/or resolving conflicts). The amount of ideas shared is reflected in the size of the cloud callout emanating from either member of the dyad. The size of the Input Control arrows reflects the amount of input control (keyboard and/or mouse usage) one member has relative to the other (e.g. the arrow of member1 will be larger if s/he hogs the computer most of the time). The size of the visual/audio feedback arrows reflects the amount of information received by a member of the dyad relative to the other (e.g. the arrow of member1 will be smaller if s/he does not take note of the computer output most of the time).

Figure 2 illustrates what was observed to be Dyad1's version of the model.

Fig. 2. Dyad1 Information and Control



While S11 was observed to have given more ideas, it was S12 who had more control over input. Aside from keyboard and mouse control, S12 was observed using the mouse to assist S11 in inputting code. Both members of Dyad 1 were equally distanced from the monitor and seemed equally concentrated on the feedback provided by the computer.

Regarding the transactions (verbal or otherwise) between S11 and S12, both asked for and gave code clarifications. Code clarification refers to queries regarding the nature (what for) of the code they are inputting. Below are some examples:

S12: (editing code)
S11: Kasama ba doon ang sagot? (Is the answer part of this?)
S12: Hindi puwede. (No.)
S11: Ay oo nga pala! (You're right!)

S11: (editing code)
S12: Para saan ba yan? (What's that for?)
S11: Para yan sa FOR-NEXT. (It's for the FOR-NEXT loop.)

S12: (editing code) Wala na 'to, di ba? (I could take this out, right?)
S11: Oo. (Yes.)

Aside from code clarifications, both were observed to have given unsolicited code solutions while the other one was editing code. In such instances, the one editing complied with the suggestion or if s/he failed to understand the suggestion, the one who gave it, takes over the keyboard. Below are examples of unsolicited code solutions:

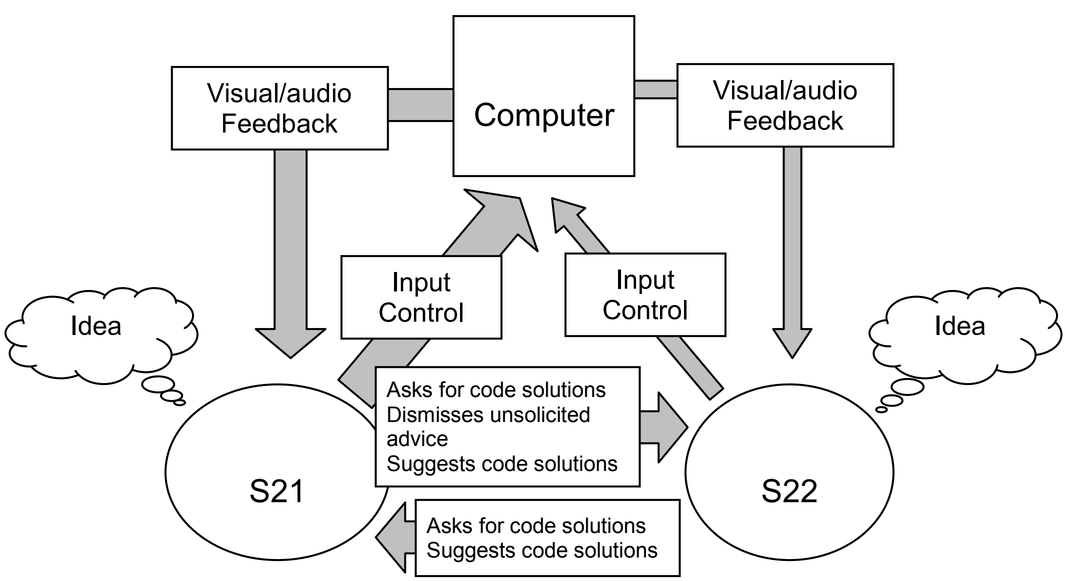
S11:	Dapat ano... di may kanya-kanya silang ... (I think you should ... aren't they supposed to each have...)
S11:	Baka kailangan nating gumawa ng isa pa. (We should probably make another one.)
S12:	Mukhang dito siya di ba para walang lalabas sa ... (I think you should put that here so nothing will come out...)
S12:	Kung paghiwalay-hiwalayin kaya natin ... (What if we split this up?)

The only difference between S11 and S12 was that S11 gave encouragements as exemplified in the following:

S11:	Ayan, dapat may kanya-kanya silang sum3 (That's it ... they should each have sum3)
S11:	Di bale, kung hindi natin ma-homework mamaya ... (It's ok if we can't work on this at home...)

Figure 3 shows what was observed to be Dyad2's version of the model:

Fig. 3. Dyad2 Information and Control



Most of the time, S22 was observed sitting behind S21 while S21 sat in front of the computer practically monopolizing input control; hence, the limited input control and visual/ audio feedback of S22. There were times, however when S22 managed to take control of the computer although she had to grab the keyboard and mouse from S21. Despite the difference in control and viewing angle, both were keenly intent in observing the computer's feedback of their code experimentations.

With regards to the transactions (verbal or otherwise) between S21 and S22, both asked for and gave code solutions. Below are some examples:

S21: Answer ito? (Is this the answer?)
S22: Oo. (Yes.)
S21: Tapos dito? (How about here?)
S22: Next line dot ... yung dot text. (Next line dot ... you know, dot text.)
S21: So text1 dot text?

S22: Paano na nga yung mali ka di ka aalis? (Do you remember how the code goes for an error trap?)
S21: Yan ang hindi ko alam. (That I don't know.)

In the first of the two preceding exchanges, only S22 seemed to be doing the programming without any help from S21. However, with regards to unsolicited code solutions, S21 made sure she was followed...

S21: Naalala mo yung ginawa natin. Nilagay natin lahat yon. (Do you remember what we did. We wrote it down.)
S22: Sa notebook? (In the notebook?)
S21: Sige kunin mo. (Go get it.)
S22: (leaves to get notebook)

S21: Eto pa sa form2, eto na lang. (Here's another one in form2, use this instead.)
S22: Yung Fix? (The Fix statement?)
S21: Hindi, yung randomize timer. (No, the randomizer timer statement.)
S22: Yun nga. (That's what I said.)

Meanwhile, S22's unsolicited advices were usually disregarded...

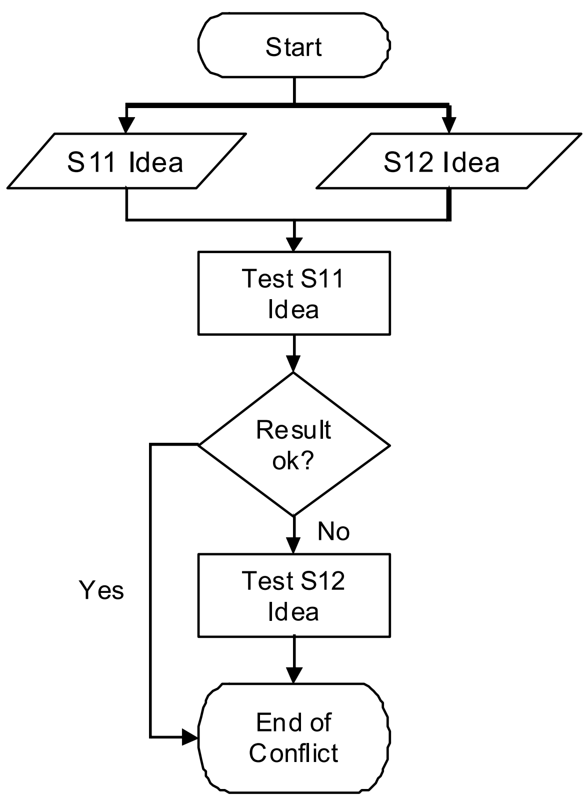
S21:	(inputting code via keyboard)
S22:	(interrupts and keys in additional code)
S21:	(deletes S22's input)
S21:	(inputting code via keyboard)
S22:	A ayan, huwag kang maglagay ng sign. (Don't put a sign there.)
S21:	Ganyan ang enter e. (But that's how enter works.)
S22:	(interrupts and keys in her suggested code)
S21:	Ibahin na lang natin. (let's try another solution.)
S22:	(notices error) Bakit nagkapalit sila? (Why did you interchange them?)
S21:	Hayaan mo na yan. (Let it be.)

These transactions resulted in a surprising difference in the conflict resolution model of the two dyads as seen in Figures 4 and 5.

Conflict Resolution

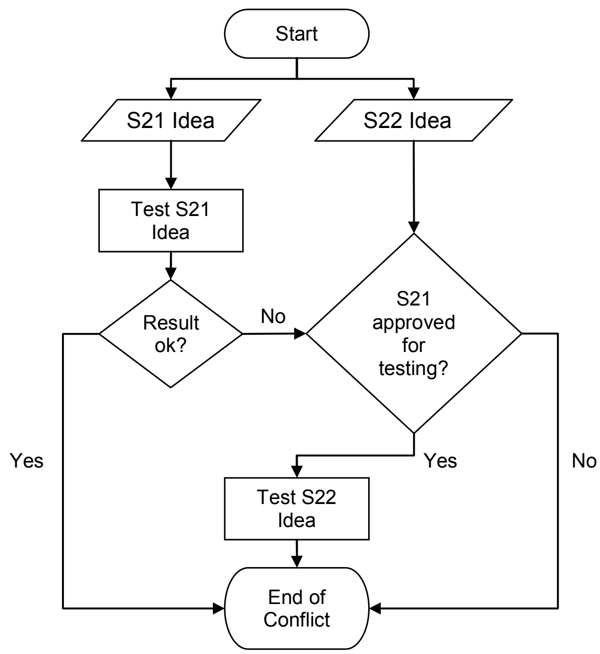
When two programmers work on a programming problem, a difference in opinion usually arise regarding programming techniques or methods which can be used to solve a particular problem in the program. The following figures illustrate how the two dyads resolved such differences.

Fig. 4. Dyad1 Conflict Resolution Model



In Dyad1, when the two suggested different solutions to a problem, they would first test the first solution given, usually by S11. If the solution given did not solve the problem then S12's idea would be used. The first solution given was always tested first. S12 admitted during the interview that S11 was faster in formulating solutions.

Fig. 5. Dyad2 Conflict Resolution Model



In Dyad2 an almost similar procedure was employed except that S21 judged S22's idea first before allowing it to be tested. This was uncovered during observations and later confirmed by S21 during her interview.

Other Findings

The interviews revealed that members of both dyads were not aware of their partner's grade or score in the first quarter periodic test. Thus, grade consciousness did not determine their actions. Also, both dyads had no predefined division of labor. The teacher had not instructed the dyads to operate in that manner or in any other manner. A programming problem can be divided into sections or procedures. Each can be done separately (one after the other) since they only had one computer to work on. Apparently, the dyads decided to work on every section together.

In the case of dyad formation, Dyad1 was formed by convenience; they were seatmates when their teacher told them to find a partner. Dyad2, on the other hand had previously agreed to work together even before their instructor asked them to form dyads because they knew each other pretty well, being classmates on and off since Grade 3.

On working as a team, all respondents admitted that working in dyads help facilitate learning. However, S11 and S21 felt that more than two persons in a group will not be effective since it will be much harder to resolve conflicts. At the same time S12 felt that success in class depends on how good the teacher is. According to her, dyads are unnecessary if the teacher teaches well.

Based on observations, Dyad2 used pattern recognition to solve a programming problem. They copied sections of code found in their notes (S22 did most of the note taking) or old programs and adapt them for use in their current programming problem. On the other hand, Dyad1 was never observed to do this. Apparently they approached each problem from a fresh perspective.

CONCLUSIONS

The two observed dyads exhibited different human-computer-human transactions. In Dyad1, although one member had greater control over computer input, the other member had more ideas in terms of code solutions. In Dyad2 both members had an equal amount of ideas for code solutions but only one member had greater control of computer input – even the input of ideas.

The two observed dyads also exhibited different human-human transactions. Dyad1 gave unsolicited code solutions and asked for and gave code clarifications. One member of Dyad1 was also observed giving encouragements. On the other hand, Dyad2 asked for and gave code solutions. Dyad2 was also observed giving unsolicited code solutions. However, one member of Dyad2 tended to dismiss such unsolicited advice.

All transactions exhibited by Dyad1 (the high performing dyad) can be considered beneficial to learning. Asking for and giving code solutions provides immediate feedback to questions that may not have been raised during the teacher's lecture-discussion prior to the exercise. Moreover, providing unsolicited code solutions preempts the asking of code clarification. Of course, the benefit of such a transaction is limited by an individual's receptiveness to unsolicited advice. Fortunately, both members were open to such advice.

On the other hand, all transactions exhibited by Dyad2 (the low-performing dyad) can be considered detrimental to learning. One member had a monopoly of input control whether it be physical (keyboard) or mental (code solutions) thus limiting the other member's chances of empirically testing her ideas. This robbed them of the opportunity to learn from their own mistakes. In addition, asking for and giving of code solutions and not code clarifications shows the low-performing dyad's reliance on rote learning.

Dyad2's reliance on rote learning was further exhibited by the method they use in solving a programming problem. Dyad2 simply assembled pre-existing codes without modifying the codes (e.g. renaming variables) to suit their program. Modification occurred only after the computer delivered an error message and if they realized that the error stemmed from an erroneously named variable in the first place. Although such an exercise (reusing code) required knowledge of the code being reused, the way this dyad employed this method suggested a lack of depth in understanding the meaning of the code – they simply knew that the code worked but not how it worked. If both members of dyad continue behaving this way, it would be logical to surmise that their combined efforts would result in limited learning opportunities and as such, a lack luster performance.

In contrast, the high-performing dyad approached the programming problem from a fresh perspective. The high-performing dyad's method generated more innovative ideas and thus created more learning opportunities.

From the limits of this study, it would seem that equally high performing individuals seem to produce high performing dyads while equally low performing individuals seem to produce low performing dyads. This leads to another possible determinant of performance – dyad composition. It may be best that a dyad should consist of at least one member with an in-depth understanding of the code or a high performing individual. Combined with openness to unsolicited advice, such a dyad would have, at least, a better performance than a dyad with two equally low performing members.

These conclusions were made based on a limited sample as is the nature of qualitative research. Therefore further verification through a quantitative study in the future is recommended.

References

McInerney, V., Mcinerney D. M. & Marsh, H. W. 1997. Effects of metacognitive strategy training within a cooperative learning context on computer achievement and anxiety: An aptitude-treatment interaction study. Journal of Educational Psychology 89(4): 686-695.

Springer, Stanne and Donovan. 1997. Measuring the Success of Small-Group Learning in College SMET Teaching: A Meta-Analysis. National Institute for Science Education, Wisconsin University at Madison. http://www.wcer.wisc.edu/nise/CL1/CL/resource/scismet.htm.

Tsai, Bethel and Huntsberger, 1999. Cooperative Strategic Learning for More Computer Achievement, Better Computer Attitude, and Less Computer Anxiety. Simon Fraser University, British Columbia, Canada. http://www.educ.sfu.ca.narstsite/conference/tsaietal/tsaietal.html.