NOVICES AND COLLABORATIVE COMPUTER PROGRAMMING: LESSONS LEARNED

Maria T. Earle
ISWD Department
Mississippi State University

Abstract

This paper describes an empirical research study that investigated what might occur when a problem based learning (PBL) framework is used to scaffold novice community college students learning of computer programming, collaboratively. While several PBL frameworks exist, a variant of Nelson's PBL framework was chosen to scaffold students' learning mostly due to its support for a collaborative learning environment. Overall, Nelson's framework proved beneficial. The findings showed that all student groups met project goals. However, study findings also revealed problems students experienced while learning in this type of environment. This paper will discuss the study and offer several recommendations on how to mitigate problems that emerged. In addition, in light of study findings, this paper will offer suggestions on how Nelson's PBL framework could be augmented to better scaffold a computer programming learning project, such as implemented in this study. Finally, study participants' perspectives on learning in this student-centered collaborative environment will be discussed.

Introduction

This study investigated how novice, non-major students in an introductory computer's course at a community college learned fundamental computer programming concepts while working in student-centered collaborative groups. The author's prior research in this area provided anecdotal data that suggests that learning how to program in a collaborative problem based learning (PBL) environment could positively influence novices' learning experience. However, this learning occurred in a teacher facilitated learning environment[13]. The author wanted to more formally assess how a PBL framework might scaffold novice students' learning of computer programming, in a student-centered, i.e not teacher-facilitated, collaborative learning environment.

The first step was to determine a suitable PBL pedagogy to scaffold this collaborative, student-centered, learning environment. There exist several PBL pedagogies. However, Nelson's [24] PBL framework, known as Collaborative Problem Solving (CPS), was eventually chosen to scaffold the programming project. This framework seemed ideal due mostly to its focus on small group collaborative learning, among other things.

Next, the researcher sought a learning aid that would help these novice students learn effectively. Learning about computer programming concepts can be cognitively challenging for novice learners, particularly non-majors [13]. To help minimize their cognitive load and add a little fun to the programming project, a dynamic autonomous humanoid robot was used as a learning aid and programming platform in this study. How the robot helped minimize students' cognitive load while learning collaboratively will be discussed in a subsequent section.

This rest of this paper is divided into six sections. Section one, Course Overview, provides a description of the course objectives. Section two, Cooperative vs. Collaborative Learning draws a distinction between cooperative and collaborative learning. Section three, Nelson's CPS Framework and Implementation outlines how Nelson's theoretical framework was implemented in the computer programming project. Section four, The Robot, describes the functionality of the

autonomous humanoid robot and how it was used in the study as a programming platform and learning aid. Section five, Findings, provides a discussion of study findings, major themes and data validity. And finally Section six, Lessons Learned and Recommendations, provides a discussion on how problems evidenced by emergent themes could be mitigated based on theory along with augmentations to Nelson's CPS framework.

Course Overview

In the fall of 2010, students enrolled in two sections of an introductory computer course (N=40) at a community college in TX were introduced to computer programming via a course semester project. The objectives of the course were to have students gain an understanding of computer hardware, software, procedures, operating systems, and human resources. In addition, students explored integration and application of computers in business and gained basic mastery in word processing, spreadsheets, databases, presentation graphics, and operating system commands. Understanding such concepts would be integral in completing the semester programming project, the final course objective.

Participants in this study collaborated in small groups over a period of three weeks to complete the course semester project which was to learn how to program an autonomous robot to complete a task. In the process of learning about computer programming collaboratively, it was hoped that students would learn basic computer programming concepts, such as flowcharting, module calls, decision points, and run time execution. In addition, while engaged in group learning it was hoped that consequently students' collaborative and computational thinking skills would increase.

Pedagogy

The researcher sought a suitable pedagogy to scaffold students learning in this collaborative, problem solving computer programming learning project. A problem based learning (PBL) theory was subsequently used for this study. Problem based pedagogies focus not only on problem solving but in additional support solving real world problems [30], which is one of the many goals of computer programming. There exist many PBL pedagogies; however, Nelson's [24] PBL framework, known as collaborative problem solving (CPS), was eventually chosen. This framework seemed ideal due mostly to its focus on small group collaborative learning, among other things.

In addition, this study sought to determine how students would learn in a student-centered, not teacher-led learning environment. In a student-centered learning environment knowledge is obtained when students work together to solve a problem [18]. Also, in this type of learning environment, there is the expectation that learning is to be constructed with minimal teacher intervention [5,18]. The justification for this type of learning environment was the lasting knowledge and student motivation that can result [4,28].

Finally, learning about computer programming concepts can be cognitively challenging for the novice learner, particularly non-majors [2, 13]. Thus, the researcher sought to minimize these novices' cognitive learning challenges. To help minimize students' cognitive load and in addition add a little fun to the programming project, a dynamic autonomous humanoid robot was used in this study as a programming platform. The robot and how it helped minimize students' cognitive load will be discussed in a subsequent section.

Cooperative vs. Collaborative Learning

This section will draw distinction between cooperative vs. collaborative learning. It is important to establish this distinction because prior research investigating group learning more often than not involved cooperative, not collaborative learning. Nonetheless, there is a difference.

Developed in the seventies, cooperative epistemologies attempted to move away from the contemporaneous traditional individualized learning during that time [1,22,26]. With cooperative learning, small groups work together to meet a learning goal; however tasks to meet such goals are subdivided among group members. Each member works individually on their task and eventually reports back to the group on their individualized findings/learning [1].

Not to suggest that cooperative learning does not impact learning outcomes, but that this type of piecemeal-learning may not scaffold an understanding of all computer programming concepts. In other words, each group member may not have the opportunity to understand *all* learning objectives in a cooperative learning environment.

Similar to cooperative learning, collaborative learning supports group learning. However, unlike cooperative learning, during the group learning process each member of the group has the opportunity to understand *all* learning concepts, not just some [32,1,14].

Some studies have shown the benefits of small group collaborative learning in cognitive subject areas, such as computer programming [20,25]. Not only does collaborative learning provide for a more holistic learning experience for students but in addition supports meaningful lasting knowledge [1]. However, traditionally, learning about computer programming tends to occur in individualized, teacher-centered learning environments [12]. Moreover, computer programming teachers tend to "value" individualized learning [17]; perhaps to better assess learning outcomes.

Nelson's Collaborative Problem Solving (CPS) Framework and Implementation

Nelson's [24] CPS framework seemed to be an ideal framework for this study, due mostly to its focus on small group collaborative learning. Nelson's framework outlines components necessary for collaborative problem solving. A subsumed list of these components include the following six components:

1. Determine group activity.
2. Form student groups.
3. Prepare students for the activity.
4. Students engage in group activity.
5. Students synthesize and reflect on activity.
6. Assessment/Closure.

The graphic in Figure 1 captures these components. Each component is shown in a chevron. Alongside the chevron are details of how the component was implemented in this study.

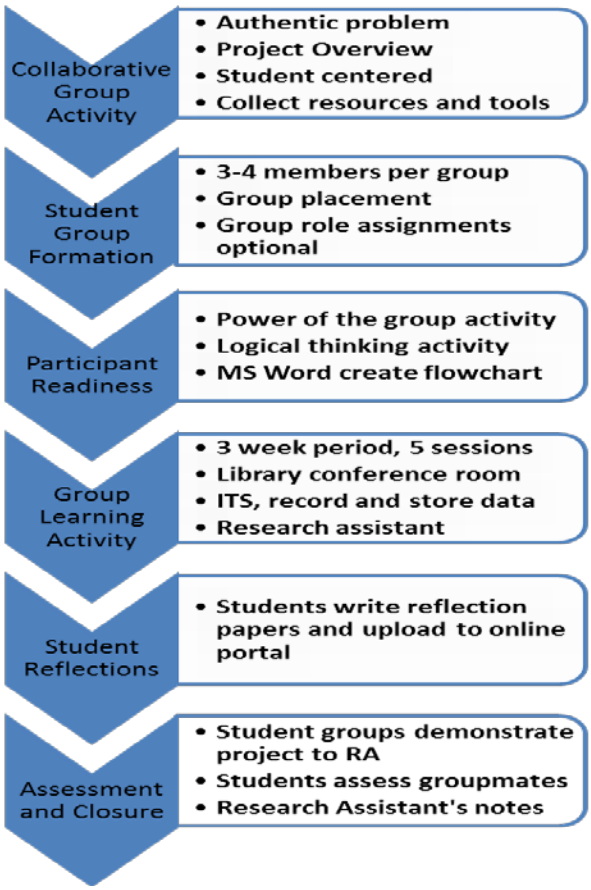

Figure 1: Nelson's CPS Framework (subsumed).

Details of each component follows.

Component 1: Build readiness. The week prior to the start of the study, the instructor/researcher introduced the research assistant (RA) to students/participants. The instructor discussed the research assistant's responsibilities for the study as follows; distribute and collect consent forms, organize the groups, video tape the labs and run each group's final programming script. To ensure a student-centered learning environment, students were told that the assistant was to provide programming assistance only if students became stuck and could not move forward. Then, in the absence of the instructor, the assistant read and distributed the consent forms to students who were instructed to either sign the forms at that time or return them on a later day.

To prepare students for the lab activity, four in-class activities occurred prior to the lab. First, in an attempt to increase an appreciation for group work, the instructor discussed activities where group work, as opposed to individual efforts had proven beneficial to the final product.

Second, a lecture on fundamental computer programming concepts was provided prior to the start of the lab. This lecture provided an overview of programming and programming languages, particularly script programming languages. To introduce programming, students engaged in a simple command line programming activity. With instructor guidance, they opened a DOS window on their computer and entered a simple DOS command line code; "echo Hello." This rudimentary command provided students with a first glimpse of writing code to command the computer to complete a task.

Third, the class engaged in an activity on logic. The activity seemed straightforward - students were instructed to write step-by-step instructions for preparing a peanut butter cracker using peanut butter, two crackers and a knife. The instructor then randomly chose three instruction sheets and attempted to assemble the peanut butter cracker carrying out student instructions exactly. One such attempt netted an entire peanut butter jar on top of a single cracker- the student's first instruction was to "put peanut butter on cracker". After three renditions of this exercise, students seem to grasp the idea for the analogy proposed of creating a peanut butter cracker with the logical sequential code needed to program a computer.

Finally, students were introduced to the robotic programming lab project. First they were given a demonstration of the robot's capabilities. A lab packet was also distributed containing the robot's owner's manual, supplemental lab material, sample lab deliverables, and a list of relevant websites. Upon completion of this programming project, three deliverables were expected from each student; programming scripts, individual reflection papers, and a programming flowchart. Since one of the key lab deliverables was a document depicting a programming flowchart, the instructor demonstrated how to create a flowchart using the MS Word software application.

Components 2 and 4: Form and norm groups, define and assign roles. To capture an authentic accounting of how novice students might work together to program a computer in a discovery learning environment, there was an assumption that students had no prior exposure to computer programming or the robot. In general, at this community college non-computer science students take this course to fulfill requirements for certification. However, in the event prior programming experience surfaced, one of the participant groups was reserved for those students and labeled as the "abstainer group." Although one student had very minimal experience in 5th grade, no students qualified for the abstainer group.

Students who agreed to participate in the study were divided into small groups. This resulted in each of the two course sections containing four groups for a total of eight participant groups.

Groups met five times for thirty minutes at a time over a two-week period during class time. They also had access to an online discussion forum to continue the conversation.

On the first day of the project, and in the absence of the teacher/researcher, the assistant collected research study consent forms and finalized group assignments in the classroom. Given the novelty of the learning experience and accounting for how students might go about learning computer programming, roles were purposely not assigned at that time.

This section will discuss the observation phase of the study. During the observation phase, students actively participated in the lab with their group mates while the assistant videotaped them.

Component 3: Problem definition. For the purposes of the study, each group was expected to figure out how to program the robot to complete a task. Students could work on a task of their choosing, or select one of the following:

1) Program the robot to pick up a ball, throw it, and then make any sound.

2) Program the robot to bowl, knocking down at least two pins, and then make any sound.

Component 5: Engage in collaborative problem-solving process. On days two and three, students further engaged in collaborative work with each other and the robot, becoming more familiar with its functions and capabilities. Using the remote controller they began to explore the robot's three programmable modules; main, vision, and sound. They would start to question how the robot's main memory and sub-memories (vision and sound) worked in order to have the robot solve the problem.

By day four, students were expected to be relatively familiar with controller programming. When the robot is put in controller mode, it can be programmed to carry out a task by executing step-by-step instructions. Students would write this script using Microsoft Word. Then, using the robot's handheld remote controller (Figure 2), students would enter their script code into the robot's memory areas (main, vision, and sound).

The final script should contain three main areas; an algorithm, the programming script, and run instructions. The SH# indicates a button on the remote handheld device that must be pushed in order to complete a programming step. Other functional symbols on the remote control include a square and the letters a,b,c, and x.

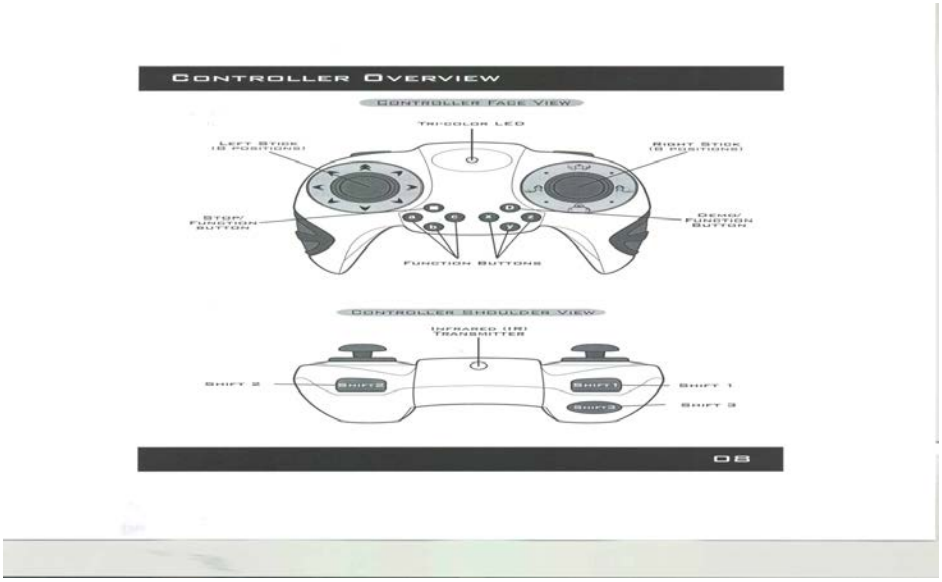

Figure 2: Robot's Handheld Remote Controller.

Dials on either side of the remote allow for positional programming. Asterisks in the script indicate comments. Bolded and italicized text in the script is not part of the executable script. For the example script, upon successful code entry and execution the robot will make five moves, pick up a ball, throw the ball and finally laugh and roar.

Algorithm

The robot must show the following five basic moves:

Lie down/stand up
Bulldoze forward over an obstacle
Back bulldoze
Left Kick then Right Karate Chop

After throwing a ball the robot must laugh and roar.

Pick up green ball
Throw a green ball
Laugh and then roar

END

Script

Initial conditions: Face the robot towards a white wall.

Main program

SH1+SH2+c	*enter main module
Sh1+Sh2+square	*clear contents of memory
	*Once standing, place an
	*object 1/2ft in front of robot.
SH1+Sh3+square	*bulldoze 4 steps forward
Sh2+Sh3+square	*bulldoze 4 steps back
Sh3+z	*left kick
Sh3+c	*right karate chop
Press square button	*stop
Sh1+Sh2+b	*run vision module
Sh1+Sh2+a	*run sound module
SH1+SH2+x	*Store main program; run

Vision module

Sh1+Sh2+b	*enter vision module; set ball
Sh1+Sh2+square	*clear contents of memory
Sh1+c	*right arm pickup ball
	*Move pedestal out of the way
Push L 3 times	*robot walks 3 times
Sh1+a	*right arm throw ball
Press square button	*stop
SH1+SH2+x	*store vision module; run

Sound Module

Sh1+Sh2+a	*enter sound module
Sh1+Sh2+square	*clear contents of memory
Sh1+Sh3+a	* robot laughs
Sh2+Sh3+a	* robot roars
Press square button	*stop
SH1+SH2+x	*store sound module; run

Run Instructions

Press Sh1+Sh2+x *robot starts moves

*As soon as the robot stands up, place an object ½ foot in front of robot's right foot.

*As soon as robot finishes the karate chop, place pedestal directly in front of right foot and place green ball in center of pedestal.

Press square button *stop

Prior to developing the programming script, groups had to create a flowchart using flowcharting symbols found in the Microsoft Word application. In developing the flowchart, students came to understand decision points and data flow. The script was then developed based on their flowchart. All groups struggled with trying to understand how to run individual modules, sound and vision. Eventually they came to realize that the code for the sound and vision modules had to be entered into the robot's memory *before* they could call those modules from the main program.

Component 6: Finalize the project, provide closure. To finalize the project students had to demonstrate their final programming code to the research assistant, write a reflection paper, and assess their group-mates.

By day five, groups were to demonstrate their final program to the research assistant by entering their final code into the robot using the handheld remote controller. After the code had been entered, the group would hand the assistant the run instructions for the program and he would carry out those instructions exactly as stated.

Due to IRB requirements, the researcher had to maintain minimum interaction with students during the study and thus the teacher did not formally assess the students programming lab. In lieu of a teacher's assessment, students assessed their group mates. A peer-evaluation form was completed by each student and turned into the assistant. Students programming scripts and peer-evaluation forms were stored in the division offices along with the daily lab video recordings.

Finally, after lab completion, students wrote lab reflection papers about their experiences. In addition, the instructor provided students with a debriefing on their learning experience including a detailed explanation of computer programming and how they had discovered fundamental computer programming concepts by engaging in their programming activity. Each student turned in the following:

1. Group lab paper depicting their group's algorithm, flowchart, programming script, and run instructions.

2. Peer-evaluation form for each of their group mates.

3. A lab reflection paper.

The Robot

The focus of the computer programming project was not to teach robotics but to use the robot as a learning platform, a learning aid. Educators' use of robotics in their classrooms has enhanced learning [3,31]. The robot used in this project was the Robosapien ™ V2 Robot as depicted in Figure 3.

This robot seemed ideal for this study. This 24" tall second generation Robosapien is capable of "autonomous free roam behavior (i.e., it can be programmed to move around the room) and is capable of multiple levels of environmental interaction with humans" including sensing colors and making and sensing sound [34].

Figure 3. Robosapien Robot.

The Robosapien™ V2 has two programming modes, positional and controller. In controller mode, and with the aid of the hand-held controller, the robot could automatically carry out a series of preprogrammed modular tasks with one instructional code – the objective of the lab. In positional mode, with the aid of the hand-held controller, the robot could carry out a series of individual tasks; however, each task required use of the hand-held remote to code each individual step of the programming script.

Student's programming cognitive load was scaffolded and minimized by using the robot. First, an understanding of programming language syntax was not required. The robot was programmed via a basic scripting language. This essentially reduces the cognitive load on novices for they can forego one of the more daunting concepts of computer programming, programming language syntax [2]. Furthermore, the scripted code was entered into the robot's memory via a handheld remote controller, not a keyboard.

Study Findings and Validity

This section will discuss the findings of the study in light of students' perspectives and emergent themes. In addition, the reliability and validity of study data will be discussed.

Student Perspectives and Themes**.** As discussed earlier, one of the deliverables of the programming project was a student reflection paper. Each individual student from all eight groups were to discuss their perspective and feelings on any aspect of their learning experience. One of the major themes that emerged from analysis of the student reflection papers was that overall, students enjoyed the learning experience with their group mates. Furthermore, some students expressed an appreciation for the opportunity to learn about computer programming with a robot. A few verbatim quotes regarding working collaboratively:

"I enjoyed working with my class mates and thought they were extremely helpful in the process of researching and working with the robot."

"…I believe the work group is a key element as everybody´s ideas enrich the experience."

"Working with my group mates was fun."

In addition, to gather a deeper understanding of student perspectives, a purposeful sampling of students drawn from two of the eight groups were interviewed by the researcher. Members from groups A and B were contacted for interviews; six were subsequently interviewed. Demographics of these six students were as follows: four Caucasians, one African-American and one Hispanic, ages ranged from 18 – 55, five females and one male. A representative picture, of Group A is shown in Figure 4.

The interviews were given after the completion of the programming lab. What resulted were over 900 minutes of video interviews and over 100 pages of interview transcriptions. During the interview, students watched a video of their groups' interaction over the 3-week lab period and discussed their experience. This method of interviewing during video playback is known as IPR, interpersonal process recall [19].

Figure 4. Study Participants - Group A.

A sociological analysis approach was used to analyze transcribed video data, using open coding methods [10, 6, 8]. Such analysis netted 381 raw codes whereby seven themes emerged. After further analysis of the seven themes it was deemed that two themes were subthemes of others and thus were subsumed resulting in the following five emergent themes: 1) frustration with using the robot, 2) frustration while attempting to program, 3) adversity while working collaboratively, 4) premature success, and 5) not staying on course of the stated lab objectives. Details of these themes will be

discussed next in light of two of the eight groups in the study.

Theme 1: Frustrations with Technology

The computing platform used in this study was an autonomous dynamic humanoid programmable robot. Most of the frustration students experienced with the robot were due to its novelty. Students were unfamiliar with the robot's capabilities. Some assumed it had super powers. In addition, the remote controller proved awkward to use. Some students experienced hand fatigue resulting from a fair amount of moving switches, buttons, and sliders around while entering the script and then later while debugging.

Theme 2: Frustrations with Programming

Programming frustration mounted over the three weeks of the lab period. There was considerable frustration experienced by both groups from trying to understand how to write the programming script and then debugging and revising their scripted code. One participant felt that repeated trials indicated failure. However, revising and debugging programming code is a normal, expected part of writing a program. This fact was later disclosed to students during a debriefing session that occurred after the end of the project.

Theme 3 - Premature Celebration

On the first day of the lab, both groups A and B thought they had successfully completed the lab objective and celebrated as could be witnessed on the video. However their celebration was premature. While both groups had successfully programmed the robot to complete a task, they had done so by programming the robot positionally, not automatically.

As discussed in the Robot section, the robot could be programmed positionally or automatically. The expectations were that students would enter code into the robot's memory using the robot's handheld remote controller. Then with a push of a button the robot would automatically complete all steps necessary to complete the programming objective, automatically, not in a piece-meal, step -by-step fashion.

Theme 4 – Getting off Course

Groups could choose to work on one of two lab assignments, as described in the lab handout, or they could develop their own task. For instance, while Group A initially chose one of the lab options, they subsequently decided to pursue an alternative, yet more challenging approach.

By day four, Group A realized they were not being successful. In spite of such and with very little time left, the group quickly opted for one of the original lab assignments and went on to successfully complete the project. In addition, in seeking knowledge, some participants found inappropriate material, such as a video showing how to program the robot to complete advanced tasks, a task not associated with the lab objective.

Theme 5: Adversity

Members in both groups seemingly worked well together, as witnessed on the lab videos. Moreover, students appeared motivated and excited to complete the programming task. However, signs of adversity showed in teammates working/learning style. In trying to construct knowledge collaboratively, group members had to try and come to some understanding on how to work with teammates whose work style differed.

Validity

Before a discussion on how the problems alluded to by these five emergent themes, a discussion on data validity. A concern of any empirical study should be reliability and validity of the data. According to Merriam (1998), these concerns can be approached through "careful attention to data collection, analysis,

interpretation, and the way in which the findings are presented" (p. 2398). This attention was applied to several sources of data that were collected and analyzed, and subsequently corroborated. Data included videotaped observations, interview transcriptions, reflection papers, and assistant field notes.

In addition, the notion of trustworthiness as a form of validity is often used in qualitative studies in lieu of validity measures typically found in quantitative studies. This study employed the following measures to increase the trustworthiness of research findings:

1. Member checks
2. Researcher bias check
3. Peer-Check

Member checks were completed by having interviewees review their transcribed interviews for accuracy. Researcher biases were checked via two methods. First, researcher colleagues analyzed the interview protocol strategy and suggested modifications that were implemented. Second, purposeful sampling was used to select participants for interviews by having the researcher's selection of participants corroborated against the research assistant's assessment of viable interview participants. Finally, a peer debriefer analyzed and developed a set of codes and themes which closely aligned with the study's five emergent themes.

Lessons Learned

Mitigating Emergent Themes. While the majority of students enjoyed their programming experience, the five emergent study themes pointed to problems students experienced while learning. How to mitigate these problems will be discussed in this section.

Theme 1 - Minimizing Frustrations with Technology

The technology used in this study was an autonomous dynamic humanoid programmable robot. At the time of this study robots were not seen widely as a learning aid, although they were beginning to take their place on the educational landscapes in several disciplines.

Some of the frustration students experienced might have been minimized if they had properly prepared for the lab by completing pre-lab activities. One pre-lab activity was to review lab handouts and robot user's guide that explained how to use the robot and the robot's controller. Another pre-lab activity was to view a video of the robot in action.

Students could have benefited from viewing a YouTube video that was provided to show them robot functionality. But, as later disclosed, some students did not complete these lab prerequisites. Also, as was witnessed on the lab videos, several students seemed to enjoy playing with the robot alone.

TPACK (technology, pedagogy, and content knowledge) is a theoretical learning framework that scaffolds dynamic technology rich learning environments [23,27,16], such as the one used in this study. The following components of TPACK could help mitigate this behavior:

1. Ascertain student knowledge with a lab pretest.
2. To get to understand the robot better, allow for individualized play time.
3. Provide additional time for students to understand novel technology during their collaboration.

Theme 2 - Minimizing Frustrations with Programming

Programming frustration mounted over the three- week period. However, this is to be expected when learning about programming, particularly when learning in this type of discovery based learning environment [4]. Nonetheless, a considerable amount of frustration resulted from debugging and revising programming scripts. It became obvious that

students did not understand that a given task in programming is debugging the programming script, repeatedly.

Hackman (1999), known for his work in social and organizational psychology, stated that for successful group work, group members should have an understanding of the task needed to complete a project. In addition, Weinstein (1999) suggested use of organizing cognitive strategy when learning cognitive subject matter, such as computer programming. In addition, the Computer Science Teachers Association (CSTA) provides a set of standards relating to computer science education [7]. One such standard states that students should understand group mates programming experience.

Thus, based on these components, the following recommendations are suggested to minimize frustrations during programming:

1. During pre-lab initiation activities, specifically the PBJ activity, engender an appreciation for debugging, a necessary component in computer programming by requiring students to redo PBJ instructions until perfectly correct.
2. Determine group mates' experience and then ensure all group members understand group skills

Theme 3 - Forestalling Premature Celebration

On the first day of the lab, both groups A and B thought they had successfully completed the lab objective and subsequently began to celebrate. However their celebration was premature. Students required some way of knowing what success looks like, what a successful execution meant.

Weinstein and Mayer [33] recommend use of metacognition and modifying for cognitive tasks such as computer programming. Metacognition requires a higher level of thinking whereby a skeptic would scrutinize group tasks, failures and successes. Based on this component, the following recommendations are suggested to minimize premature celebration.

1. Ensure that students fully understand what denotes successful program execution by demoing the final lab solution.
2. Provide a lab pretest questioning students understanding of the lab objective.
3. Groups should assign one member to be a skeptic to question group successes.
4. Allow students time to recover from failure.

Theme 4 - Recommendations to Stay on Course

Groups could choose to work on one of two lab assignments, as described in the lab handout, or they could develop their own assignment. However, this creative permission resulted in some groups getting off course. In addition some students spent too much time viewing non-germane videos.

When learning in a group collaborative student-centered environment, certain limitations should be imposed. This study suggests the following to stay on course:

1. Several well piloted computer programming assignments should be assigned and adhered to by students.
2. Time should be allowed for students to elaborate and extend their knowledge and creative pursuits of computer programming.
3. Students should be provided with a list of relevant videos.

Theme 5: Adversity

While members in both groups seemingly worked well together, as witnessed on the lab videos, signs of adversity were disclosed during the interviews. In trying to construct knowledge collaboratively, group members had to try and

come to some understanding on how to work with teammates' work style particularly when it opposed their style of learning.

To minimize such adversity, first this study, recommends use of a group charter. Next, it recommends use of methodologies espoused by SCRUM, one of the better known Agile frameworks. Agile frameworks preference group interaction and trust over processes and tools [29]. The following SCRUM values are recommended:

- Focus. Because we focus on only a few things at a time, we work well together and produce excellent work. We deliver valuable items sooner.
- Courage. Because we are not alone, we feel supported and have more resources at our disposal. This gives us the courage to undertake greater challenges.
- Openness. As we work together, we practice expressing how we're doing and what's in our way. We learn that it is good to express concerns so that they can be addressed.
- Commitment. Because we have great control over our own destiny, we become more committed to success.
- Respect. As we work together, sharing successes and failures, we come to respect each other and to help each other become worthy of respect.

In summary, the five aforementioned recommendations are suggested to minimize unnecessary frustrations as emerged from the study. In addition, this study recommends augmentation of Nelson's CPS framework when utilized in a student-centered computer programming project. The next section will discuss this augmentation. Nelson's CPS calls for teacher intervention and sufficient time to complete a learning activity. However, this study utilized a student-centered learning environment. In addition, the project time was a little over three weeks. In retrospect, this may not have been enough time to fully engage.

Nonetheless if a student-centered learning environment is still desired for your computer programming project and time is of the essence, the following recommendations are suggested to augment Nelson's step #5:

- Use piloted, vetted computer programming learning aids.
- Keep on hand, all robotic user's manuals and resources.
- Mandate that students work on one of the lab options, only. As a first programming experience in this type of environment, no creative extensions should be allowed.
- Student groups should keep a log of programming debugging efforts.
- A lab pre-test should be given to assess student's understanding of lab objectives.
- Each group should assign a skeptic or a "devil's advocate" to scrutinize group tasks.
- Novices should work in their natural comfort zone. They should be allowed to determine which tasks they feel most comfortable working on in the group.
- Each group creates a group charter.
- Add SCRUM values to the group charter.

Bibliography

1. Barkley, E. F., Cross, K. P., & Major, C. H. (2005). Collaborative learning techniques. San Francisco, CA: John Wiley & Sons, Inc.
2. Black, M. (2009). A processor design project for a first course in computer

organization. *Computers in Education Journal*, 20(1), 95-103.

3. Bratzel, B. (2005). Physics by Design. Knoxville, TN: College House Enterprises.

4. Bruner, J. (1961). The act of discovery. Harvard: Harvard Educational Review.

5. Bruning, R. H., Schraw, G. J., Norby, M. M., & Ronning, R. R. (2004). Cognitive psychology and instruction. Upper Saddle River, NJ: Prentice Hall.

6. Carspecken, P. F. (1996). Critical ethnography in educational research. New York, NY: Routledge.

7. CSTA. (2011). Computer Science Teachers Association. Retrieved April 20, 2011, from New CS Curriculum Standards (Draft for Public Comment): http://csta.acm.org/includes/Other/CSTAStandardsReview2011.pdf

8. Corbin, J. & Strauss, A. (2008). Basics of qualitative research (3e). Thousand Oaks: Sage Publications, Inc.

9. Del-Siegle. (2003). Mentors on the net: Extending learning through telementoring. *Gifted Child Today*, 26(4), 51-54,63.

10. Denzin, N. K., & Lincoln, Y. S. (2003). Collecting and interpreting qualitative materials (2nd ed.). Thousand Oaks, CA: Sage Publications, Inc.

11. Denzin, N. K., & Lincoln, Y. S. (1998). The landscape of qualitative research: Theories and issues. Thousand Oaks, CA: SAGE Publications, Inc.

12. D'Souza, C., Kazlauskas, A., & Thomas, T. (2009). Scaffolding strategies for teaching introductory programming. Doctoral Student Consortium of Proceedings of the 17th International Conference on Computers in Education (pp. 32-41). Hong Kong: Asia-Pacific Society for Computers in Education.

13. Earle, M.T. (2010, Spring). Connecting kinesthetic learners to computer science concepts. *The Journal for Computing Teachers.*

14. Gokhale, A. A. (1995). Collaborative learning enhances critical thinking. Journal of Technology Education, 7 (1), 22-30.

15. Hackman, J. (1990). Groups that work (and those that don't). San Francisco: Jossey-Bass.

16. Harris, J. B., & Hofer, M. J. (2011). Technological pedagogical content knowledge (TPACK) in action: A descriptive study of secondary teachers' curriculum-based, technology-related instructional planning. JRTE, 43 (3), 211-229.

17. Hoganson, K. (2008). Concepts in computing. Sudbury, MA: Jones and Bartlett Publishers.

18. Jonassen, D. H., & Land, S. M. (2000). Theoretical foundations of learning environments. Mahwah, NJ: Lawrence Erlbaum Associates.

19. Kagan, N., Krathwohl, D., & Miller, R. (1963). Stimulated recall in therapy using video tape: A case study. Journal of Counseling Psychology, 10 (3), 237-243.

20. Matzko, S., & Davis, T. (2006). Pair design in undergraduate labs. Consortium for Computing Sciences in Colleges , 123

21. Merriam (2001). Qualitative research and case study applications in education:

Revised and expanded from Case Study Research in Education. San Francisco: Jossey-Bass

22. Michaelsen, L. K., Knight, A. B., & Fink, L. D. (2002). Team-based learning: A transformative use of small groups in college teaching. Westport: Greenwood Publishing Group, Inc.

23. Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers College Record, 108 (6), 1017-1054.

24. Nelson, L. (1999). Collaborative Problem Solving. Mahwah, NJ: Lawrence Erlbaum Associates.

25. Pastel, R. (2006). Student assessment of group laboratories in a data structures course. Consortium for Computing Sciences in Colleges, 221.

26. Pederson, J. E., & Digby, A. D. (1995). Secondary schools and cooperative learning. New York: Garland Publishing, Inc.

27. Pierson, M. E. (1999). Technology integration practice as a function of pedagogical expertise. Arizona State University.

28. Prince, M. J., & Felder, R. M. (2006). Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases. *Journal of Engineering Education*, 95(2), 123-138.

29. SCRUM Alliance, 2011. Core SCRUM. Values and Roles. Retrieved August 8, 2013 from http://www.scrumalliance.org/ why-scrum/core-scrum-values-roles

30. Smith, P. L., & Ragan, T. J. (1999). Instructional design (2nd ed.). Hoboken: John Wiley & Sons, Inc.

31. Wang, E. (2004). Engineering with LEGO bricks and RoboLab. Knoxville, TN: College House Enterprises.

32. Weinberger, A., Ertl, B., & Fischer, F. (2005). Epistemic and social scripts in computer-supported collaborative learning. Instructional Science, 33 (1), 1-30.

33. Weinstein, C. F., & Mayer, R. F. (1986). The teaching of learning strategies: In M.C. Wittrock (Ed.), Handbook of research on teaching (3rd ed.) (pp. 315-329). New York: Macmillan.

34. Wowwee. (2011). RoboSapien v2. Retrieved September 16, 2008 from wowwee.com: www.wowwee.com

Biographical Information

Maria T. Earle is an Assistant Professor at Mississippi State University, ISWD department. She received her Bachelors in Electrical Engineering from Boston University, Masters in Software Engineering from Penn State University, and her Doctorate in Education from University of Houston. Prior to academia, she worked for over fifteen years in the software industry. Her research interests include studying how instructional technology coupled with motivational pedagogies could scaffold/maintain students' interest in STEM disciplines. She wishes to thank study participants and to express a deep appreciation to her dissertation committee: Dr. Melissa E. Pierson (Chair and Doctoral Advisor), Dr. Allen R. Warner, Dr. Mimi M. Lee, and Dr. Farrokh Attarzadeh.