Case Study: Using Project Based Learning to Develop Parallel Programing and Soft Skills

Awad A. Younis, Rajshekhar Sunderraman, Mike Metzler and Anu G. Bourgeois
Georgia State University
Atlanta, GA, USA
{amussa, raj, mmetzler, abourgeois}@gsu.edu

***Abstract*—** In today's environment, where every computer, including cell phones, are multicore, it is essential that students develop parallel programming skills. It remains a challenge to develop effective techniques for teaching parallel programing skills. Another challenge is finding time within already packed lectures to cover additional material. To that end, we investigate the effectiveness of using Project Based Learning (PBL) to teach parallel programming skills early in the curriculum by developing and incorporating a PBL module into CSc 3210 (Computer Organization and Programming). This is a core course taken by all computer science majors and is a prerequisite to many of our senior-level classes. In our case study, 124 students are organized into 26 diverse groups, with four or five students per group, and assigned five project assignments, each of two-weeks duration. Given a Raspberry PI, students will explore its multicore architecture and create programs for shared memory parallelism using OpenMP and C language. Our results show that incorporating this PBL module has a significant and direct effect on the student's growth in parallel programming skills. As a side benefit, we also show that there is a direct improvement on a student's personal growth in terms soft skills, which is essential in the professional development and success in the workplace. By having students experience PBL in an early class, close to the midpoint of the academic program, it can serve as a mini-capstone project. Furthermore, students can collaboratively learn by themselves (through teamwork) and apply the fundamentals of parallel programming skills without the need for separate lectures, labs, or workshops.

Keywords- Parallel Computing; Parallel Programming; OpenMP; Shared Memory; Project Based Learning; Raspberry PI; Soft Skills; Teamwork Technology

I. INTRODUCTION

Consider that a majority of people own a smartphone, which are more integrated in our lives, and thought of as essential by about half the owners. Since most smartphones are built with multiple cores, as is a typical computer, it is vital that computer science students have an understanding of parallel programming and take advantage of the shift toward multicore architectures. As a result, parallel programming skills are now required to be covered by computer science curriculum [1]. Learning parallel programming concepts, such as multicore computer architectures, cloud distributed computing, and general-purpose GPU will enable students to gain these needed technical skills.

To this end, modules have been provided by CSinParallel [2] (supported by a grant from NSF-TUES) to teach principles of parallelism and hands-on practice with parallel computing. However, the effectiveness of teaching and learning strategies for those modules have not been considered. Recent research findings [3] have shown teaching parallelism using single-board computers (SBCs) such as the Raspberry Pi as a uniform work environment is effective. The proposed approach, however, used a lecture (in a workshop) as a strategy for teaching, which did not support diverse groups of students in learning to practice their skills. Another challenge is finding time within the already packed computer science curriculum to cover additional topics, such as parallel programming principles.

Beyond the technical skills required to be part of a computer science curriculum, it is critical to develop professional soft skills. Research findings [4] show that senior engineering students are not often aware of soft skills, or ways to use them through conflict resolution. Burrows and Mike in [4] also emphasize the need for embedding soft skills in a course early in the students' program as explicit instruction and emphasizing their importance and utilization in future class projects. It is not sufficient to put students in groups and ask them to work together. Students must first learn the teamwork skills needed to function successfully in group projects. Just as technical skills are critical for success in the workplace, soft skills, such as teamwork, decision making, cooperation, collaboration, conflict resolution, and communication are recognized as crucial for computer science students because technical work is becoming more and more collaborative and interdisciplinary.

In this study, we investigate the effectiveness of using Project Based Learning (PBL) to teach parallel programming concepts and soft skills early in the curriculum. PBL is an engaging instructional model that provides more learning experience and leads to higher-level cognitive development through students' engagement with complex problems [5]. PBL supports diverse groups of students in learning and practicing skills in problem solving, communication, collaboration, and self-management. With collaborative learning, availability of the required materials, and guidance from the instructor, we hypothesize that students can collaboratively learn by themselves and apply the fundamentals parallel programming principles and soft skills without the need for separate lectures, labs, or workshops. Our research hypotheses are formulated as follows:

- Hypothesis 1: There is a difference in emphasis on parallel programming and soft skills between the first and second parts of the semester.
- Hypothesis 2: By incorporating project-based learning, the students acquire personal growth and improvement on their parallel programming and soft skills.

- Hypothesis 3: Students growth in parallel programming and soft skills did increase when greater emphasis is placed on these areas.

To investigate the effectiveness of using PBL, we developed and incorporated a semester-long PBL module in CSc 3210 (Computer Organization and Programming), a core course taken by all majors and a prerequisite to most of our senior-level classes. CSc 3210 is taken close to the midpoint of the academic program and can serve as a mini-capstone course that provides an opportunity for the students to learn the required parallel programming skills early in the curriculum. Our courses are semester based and last a total of 15 weeks. In the first week, we divide the students (124 total) into 26 diverse groups (up to five per group). The team formation is based on multiple criteria including gender, system and programming experience, experience in group work, GPA, and technical writing experience. These criteria are intended to help groups have a balance in ability, of mixed gender and would avoid predetermined groups of friends. After teams have been formed, five project assignments, each of two-weeks duration, are provided to the students across the course of the semester. (Refer to Fig. 1 for the timeline.) With each assignment, we provide the required learning materials (Teamwork Basics [6], Raspberry PI Multicore architecture [7], Shared Memory Parallel Patternlets in OpenMP [8], Introduction to Parallel Computing [9], CPU vs. SOC – The battle for the future of computing [10], and Introduction to Parallel Programming and MapReduce [11]), learning objectives, grading criteria, team policies, peer rating form of team members' contributions to the team, and the tasks.

The focus of the first two-week assignment (described in Section 2) is on soft skills development. In particular, by using the given materials, students collaboratively learn and apply soft and critical thinking skills in terms of teamwork basics, decision making, task identifications, planning and scheduling, cooperation and collaboration, conflict resolution, and communication. We required students to utilize the following teamwork technologies: (1) Slack, a messaging application to communicate, (2) GitHub, a social networking site for programmers to collaborate, create customized workflows, and share code, (3) Google Docs, an online word processor to collaborate and produce project assignments reports, and (4) Videos and YouTube, to shoot, edit, and upload videos to a YouTube channel to present the results. All these technologies are free and available to all students.

The remaining four assignments focus on the technical skills pertaining to parallel programming concepts. Now, equipped with the basic soft skills and by using the given materials, students learn and apply parallel programming concepts in terms of sequential and parallel computation, multi-processor computer architectures (e.g. SISD, SIMD, MISD, and MIMD), processes and threads, parallel programming models, shared memory parallelism, data race, OpenMP and C language, and MapReduce. In order to accomplish these goals, each group is given a Raspberry PI kit (costs $59), a uniform work environment, and a credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. The equipment been paid by a grant funded by the Center for Excellence in Teaching and Learning (CETL) at Georgia State University. The assignments ask the teams to explore the Raspberry PI's multicore architecture and use it to create programs for shared memory parallelism using OpenMP and C. We provide details on these assignments in the next section.

OpenMP has been chosen, because unlike complex parallel platforms, it is designed to make it relatively easy to add parallelism to existing sequential programs, and write new parallel programs from scratch. Raspberry PI, on the other hand, has been chosen because components such as the processor, memory unit, storage device, and others are clearly visible and that makes them readily available for visual and tactile learners. Another motivation is that our CSc 3210 teaches Intel X86 (CISC) architecture, and by using the Raspberry PI, it gives students an exposure to ARM (RISC) architecture. It is also similar to what they may encounter in today's ubiquitous mobile devices, and help them explore its basic Instruction Set Architecture (IST) and compare it with Intel X86 in terms of data movement, instruction encoding, immediate value representation, and memory layout.

To assess the effectiveness of our PBL module, we use the survey proposed by S. Beyerlein, E. Davishahl, D. Davis, J. Lyons and K. Gentili [12]. This survey is used to measure students' personal growth in terms of parallel programming skills, including problem definition, information gathering, idea generation, and implementation, and soft skills including teamwork, communication and evaluation/decision making. The survey is taken twice, once at the mid-point of the semester and then at the end of the term (Fig. 1). After collecting the data, we then use descriptive statistics, T-tests, Cohen's d (Effect Size) [13], Pearson Correlation, and Ranking of students' growth (Composite Score) [12] to determine the impact of the PBL approach by testing the hypotheses.

The paper is organized as follows. Section 2 describes the methodology and PBL organization, Section 3 presents results, Section 4 discusses the results, and Section 5 concludes with comments and issues that need further investigation.

II. METHODOLOGY

In this section, we will first go over the PBL module design and implementation in terms of the module's description, integration, team formation, design, implementation, and evaluation. We will then present the PBL module assessment.

A. PBL Module Design and Implementation

PBL Module description: In this study, we have developed a semester-long Project Based Learning (PBL) module to teach parallel programming principles and soft skills. PBL is an engaging instructional model that provides more learning experience and leads to higher-level cognitive development through students' direct engagement with complex problems [5]. PBL supports diverse groups of

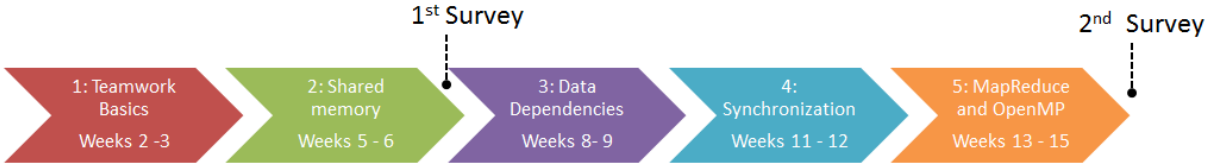

Figure 1. PBL Timeline. The module is broken into 5 assignments, each with 2 weeks duration. Surveys are conducted at the midpoint and end of the semester.

students in learning and practicing targeted skills in problem solving, communication, collaboration, and self-management. In this module, students will first learn and apply soft skills in terms of teamwork basics, decisions making, task identifications, planning and scheduling, cooperation and collaboration, conflict resolution, and communication. Then, equipped with the basic soft skills, students will learn and apply parallel programming concepts in terms of sequential and parallel computation, multi-processor computer architectures (e.g. SISD, SIMD, MISD, and MIMD), processes and threads, parallel programming models, shared memory parallelism, data race, OpenMP and C language.

PBL Module integration: We incorporate the PBL module into CSc 3210, Computer Organization and Programming, a core bridge course and a pre-requisite to many of our senior-level courses in the Computer Science Department at Georgia State University. Two sections of CSC 3210 were selected for this study in Fall 2018 semester. Each section had 62 students enrolled (the first section had 16 females and the second section had 10 females). Both sections were taught by the same instructor and with the same instructional strategy incorporating the PBL module to teach parallel programming principles and soft skills.

Team formation: To form the teams, students in each section were organized into thirteen diverse groups (up to five per group) based on the following criteria: gender, system and programming experience, experience in group work, GPA, and technical writing experience. These criteria are intended to balance groups in terms of ability and assure a mixed gender and avoidance of predetermined groups of friends. Having the instructor form teams based on predetermined criteria has been found to be more effective than when students form their own [14]. Once grouped, each team elects a team coordinator and this role is to be rotated among team members for each assignment. The team coordinator interfaces between the instructor and the team, turns in the documents, reviews the returned assignments and ensures everyone understands why any points were lost, how to correct any errors, and identify, assign, and schedule tasks to the team members, as well as monitor and report the progress of the assigned tasks.

PBL Module design: After teams have been formed, five project assignments, each of two weeks duration, are to be completed. In these assignments, we provide learning objectives, grading criteria, team policies, a peer rating form of team members' contributions to the team, and the required tasks. Each assignment includes the following components: Planning and Scheduling (work breakdown structure: assignee name, email, assigned task, duration in hours, dependency, due date, note), Collaboration, Written Report, and Video Presentation. Each student must participate in the group video, which must be 5-10 minutes long and posted on YouTube. The following guide is used in each presentation to help students focus on their presentation: Introduce yourself and your role; Identify your task for this assignment and 2-3 key things learned; How you will apply what you learned in your next assignment, academic life (future classes), and in the future job; What the best/most challenging/worst experience you encountered.

PBL Module implementation: For each assignment, the groups are provided with the one or more of the following materials to do the assigned assignment:

- Teamwork Basics material [6],
- Raspberry PI Multicore architecture [7],
- Shared Memory Parallel Patternlets in OpenMP [8],
- Introduction to Parallel Computing [9],
- CPU vs. SOC – The battle for the future of computing [10], and
- Introduction to Parallel Programming and MapReduce [11],

In the assignments that have programming in OpenMP and C (Assignments 2 – 5), each group is required to include in the written report what they have done and observed, including screenshots and code snippets. Simply attaching screenshots and code snippets without any explanation will not receive credits. They also need to provide explanations for the observations that are interesting or surprising.

We now provide a brief description of the 5 assignments shown in Fig. 1 and given over the course of the semester.

Assignment 1: Each group, using the Teamwork Basics material [6], will first collaboratively-learn and apply the team Ground Rules: work norms, facilitator norms, communication norms, meeting norms, handling difficult behavior, and handling group problems. Next, they are to watch introductory/tutorial videos on the teamwork technologies that will be utilized. The groups will collaboratively-learn, apply and report how to utilize teamwork technologies such as Slack, a messaging application to communicate, GitHub, a social networking site for programmers to collaborate, create customized workflows, and share code, Online Word processor (e.g. Google Docs) to collaborate and produce project assignments reports, and Videos and YouTube, to shoot, edit, and upload videos to a YouTube channel to present their results.

Assignment 2: Using the Raspberry PI Multicore architecture materials [7], Shared Memory Parallel Patternlets in OpenMP [8] and Introduction to Parallel Computing [9], groups will collaboratively-learn fundamental parallel computing principles based on the following questions: Identify the components on the Raspberry PI B+. How many cores does the Raspberry Pi's B+ CPU have? What is the difference between sequential and parallel computation and identify the practical significance of each? Identify the basic form of data and task parallelism in computational problems? Explain the differences between processes and threads? What is OpenMP and what is OpenMP pragmas? What applications benefit from multi-core? Next, each group will be given a Raspberry PI and a uniform work environment. The groups are required

to 1) download and install the Operating System (RASPBIAN) Images on MicroSD, and 2) setup the Raspberry PI to connect with a monitor or a laptop. Then the groups will use the PI to create, compile, run, and modify a parallel program that illustrates:

1) The fork-join programming pattern,

2) A Single Program Multiple Data (SPMD) pattern for shared memory parallelism using OpenMP and C language, and

3) Shared memory concerns. In particular, students will see that by sharing one bank of memory, programmers need to be a bit more careful about declaring their variables (scope matter) to avoid the data race problem.

Assignment 3: In addition to the materials in [7], [8], and [9], groups will use CPU vs. SOC material [10] to answer the following questions: What is: Task, Pipelining, Shared Memory, Communications, and Synchronization? Classify parallel computers based on Flynn's taxonomy (briefly describe each one of them). What are the Parallel Programming Models? List and briefly describe the types of Parallel Computer Memory Architecture. What type is used by OpenMP and why? Compare Shared Memory Model with Threads Model. What is System On Chip (SOC)? Does Raspberry PI use SOC? Explain what the advantages are of having a System on a Chip rather than separate CPU, GPU and RAM components? Then, using the Raspberry PI, groups will create, compile, run, and modify the following programs:

1) Running Loops in Parallel: illustrates the use of OpenMP's default parallel for loop in which threads iterate through equal sized chunks of the index range.

2) Scheduling of Parallel Loops (static and dynamic): illustrates how to make OpenMP map threads to parallel loop iterations in chunks of size one, two, and three.

3) When Loops Have Dependencies: illustrates the OpenMP parallel-for loop's reduction clause.

Assignment 4: Using the materials in [8] and [9], groups will first answer the following questions: What is the race condition? Why race condition is difficult to reproduce and debug? How can it be fixed? Provide an example from your Assignment 2. Compare the following: Collective synchronization (barrier) with Collective communication (reduction), and Master-worker with fork join. Next, using the Raspberry PI, groups will create, compile, run, and modify the following programs:

1) Integration Using the Trapezoidal Rule: illustrates the use of parallel for loop, private, shared, and reduction clauses.

2) Coordination: Synchronization with a Barrier: illustrates the use of the OpenMP barrier command, using the commandline to control the number of threads.

3) The Master-Worker Implementation Strategy: illustrates the master-worker pattern in OpenMP.

Assignment 5: After reading the paper "Introduction to Parallel Programming and MapReduce" [11], groups will answer the following questions: What are the basic steps (show all steps) in building a parallel program (show at least one example)? What is MapReduce? What is map and what is a reduce? Why MapReduce (show an example for MapReduce)? Explain how MapReduce model is executed. List and describe three examples that are expressed as MapReduce computations. When do we use OpenMP, MPI and, MapReduce (Hadoop), and why? Given the material in [7], each group needs to first report their understanding of the Drug Design and DNA problem and the algorithmic strategy of the Drug Design and DNA in parallel using a sequential, an OpenMP, and a C++11 Threads solutions. Using the Raspberry PI, groups will create, compile, run, and modify the following programs:

1) Apply a sequential, OpenMP, and a C++11 Threads solutions to Drug Design and DNA problem.

2) Measure the running time of each implementation, and answer the following questions:

- Which approach is fastest?
- What are the number of lines in each file (size of the program vs. performance)?
- How does the C++11 implementation compare to the OpenMP implementations?

3) Increase the number of threads to 5 threads, what is the run time for each?

4) Increase the maximum ligand length to 7, and rerun each program. What is the run time for each?

PBL Module evaluation: The PBL module has been assigned 25% of the class overall grade. The assigned grade weight has been equally distributed across the five assignments. Each student who contributes in the assignment will receive the team assigned grade. If a team member refuses to cooperate or partially cooperated on an assignment, a zero grade will be assigned for that assignment. If the problem persists, the team should meet with the instructor so that the problem can be resolved if possible, otherwise, grade of zeroes will be assigned for the remaining assignments. Each assignment will be graded, and a feedback will be sent to the team coordinator a week after the assignment due date. To assess individuals' performance, one quiz after each assignment due date is to be taken (five in total), and two tests are to be taken, one in the middle of the semester (midterm) and the other one at the end of the semester (final exam).

B. PBL Module Assessment

A five-point scale survey proposed by S. Beyerlein, E. Davishahl, D. Davis, J. Lyons and K. Gentili [12] is used to collect data and assess class emphasis and students' growth in terms of parallel programming and soft skills. The survey includes both "Class Emphasis" and "Personal Growth" categories as a mechanism to judge how effective the class is in developing specific skills. The survey considers seven elements: teamwork, information gathering, problem definition, idea generation, evaluation and decision making, implementation, and communication. Class Emphasis scores are described as 1: Did not discuss, 2: Minor emphasis 3: Some emphasis, 4: Significant emphasis, and 5: Major emphasis. Whereas Personal Growth scores are described as

1: I did not use this skill within this class, 2: I used previous skills and had little growth, 3: I grew some and gained a few new skills, 4: I experienced a significant growth and added several skills, and 5: I experienced a tremendous growth and added many new skills. The first item in each of the categories in the survey is the basic definition of that element. For instance, as shown in in Fig. 2, teamwork is defined as "Individuals participate effectively in groups or teams."

TEAMWORK	Class Emphasis	Personal Growth
Individuals participate effectively in groups or teams	1 2 3 4 5	1 2 3 4 5
Individuals understand their own and other member's styles of thinking and how they affect teamwork	1 2 3 4 5	1 2 3 4 5
Individuals understand the different roles included in effective teamwork and responsibilities of each role	1 2 3 4 5	1 2 3 4 5
Individuals use effective group communication skills: listening, speaking, visual communication	1 2 3 4 5	1 2 3 4 5
Individuals cooperate to support effective teamwork	1 2 3 4 5	1 2 3 4 5

Figure 2 Example of Team Design Skills Growth Survey [12]

The next items in that category are components or performance indicators of that element. For teamwork, there are four characteristic items (see Fig.2): Individuals understand their own and other member's styles of thinking and how they affect teamwork, Individuals understand the different roles included in effective teamwork and responsibilities of each role, Individuals use effective group communication skills: listening, speaking, visual communication, and Individuals cooperate to support effective teamwork. The survey is to be taken twice, once at mid-semester and the other at the end of the semester as shown in Fig. 1.

Once the data is collected, we use descriptive statistics, T-test, Cohen's d (Effect Size) [13], Pearson Correlation, and Ranking of students' growth (Composite Score) [12] to determine the impact of the intervention. Cohen's d is used to measure how big the effect of the intervention is. Cohen suggested that d=0.2 be considered a 'small' effect size, 0.5 represents a 'medium' effect size and 0.8 a 'large' effect size. In other words, if two groups' means differ by less than 0.2 standard deviations, the difference is trivial although it is statistically significant. We also used the Composite Score to measure the Ranking of students' growth. The Composite Score is calculated by averaging the 'definition' and the 'overall performance average of individual components. The Composite Score is used because it provides information from two different perspectives: global from the definition and focused from the components.

III. RESULTS

In this section, we will first present the descriptive statistics of our sample and then go over the results analysis.

A. *Descriptive Statistics*

The statistical analyses are based on a sample of 124 computer science students enrolled at a large public university. Each student surveyed was enrolled in a project based in CSC 3210, Computer Organization and Programming course. The survey was first distributed at the midpoint of the semester. The same survey was then distributed to the students at the end of the semester (see Fig. 2). Of the surveys collected, 98(79.03%) were from males and 26(20.97%) were from females.

B. *Results Analysis*

Hypothesis 1: There is a difference in emphasis on parallel programming and soft skills between the first and second parts of the semester. A T-test to determine if there was a difference between the first part of the semester and second part regarding class emphasis (variable1: class-emphasis1 and variable2: class- emphasis2) on parallel programming principles and soft skills was performed. The two variables were created by averaging all class emphasis question scores on the two surveys respectively. The results of the T-test, as shown in Table 1, indicate that the class emphasis comparison was significant. This shows that the instructor did emphasize the skills more during the second half of the semester.

Table 1. T-test: Class Emphasis and Personal Growth

	Mean Difference	t	N	p-value
Class Emphasis	-0.10	-2.63	124	0.039
Personal Growth	-0.20	-5.11	124	0.002

To measure how big the effect of the intervention (course emphasis), Cohen's d (Effect Size) was used. More details about Cohen's d can be found in Section 2 in the Module Assessment subsection. Cohen's d for course emphasis was calculated using the following formula: Cohen's d = (M2 - M1)/SDpooled, where, SDpooled = $\sqrt{((SD12 + SD22)/2)}$. It should be noted that Cohen's d =0.2 is considered a 'small' effect size, 0.5 represents a 'medium' effect size and 0.8 a 'large' effect size. The results of the Cohen's d as shown in Table 2 indicate that the effect of the class emphasis represents a 'medium' effect size and the group means differ by 0.5 standard deviations and thus the difference is not trivial.

Table 2. Cohen's d of Course Emphasis

	First Half Survey	Second Half Survey
Mean (M):	4.023068[a]	4.124365[a]
Standard deviation (s):	0.232416	0.172052
Sample size (n):	124	124
Cohen's d =	(4.124365 - 4.023068)/0.204474 = **0.50**	

a. (4: Significant emphasis and 5: major emphasis [12])

Hypothesis 2: By incorporating project-based learning, the students acquire personal growth and improvement on their parallel programming and soft skills. A second T-test to determine if there was a difference between the first part of the semester and second part regarding students acquiring personal growth on parallel programming and soft skills was performed. The two variables were created by averaging all class emphasis question scores on the two surveys respectively. The results of the T-test, as shown in Table1, indicate that students used more soft and parallel skills after

the first survey than before the first survey. We also performed a second Cohen's d (Effect Size) measure. The results of the Cohen's d as shown in Table 3 indicate that the effect of incorporating project-based learning on students soft and parallel programming skills represents a 'large' effect size and the group means differ by 0.8 standard deviations and thus the difference is not trivial.

Table 3. Cohen's d (Effect Size) of Personal Growth

	First Half Survey	Second Half Survey
Mean (M):	3.81[b]	4.01[b]
Standard deviation (s):	0.262204	0.198497
Sample size (n):	124	124
Cohen's d =	(4.01 - 3.81)/0.232542 = **0.86**	

b. (3: I grew some and gained a few new skills, and 4: I experienced a significant growth and added several skills [12]

Hypothesis 3: Students growth on soft and parallel programming skills did increase when greater emphasis is placed on parallel programming and soft skills. To determine if the class emphasis on parallel programming and soft skills is associated with students' application of these skills, a Pearson correlation was performed on teamwork, information gathering, problem definition, idea generation, evaluation and decision making, implementation, and communication between class emphasis and personal growth for the first and second survey. Each skill score was created by averaging all question scores under each skill. As shown in Table 4, all correlations are positive and highly statistically significant at $p < 0.001$ level. The correlations strength fall within moderate range ($\pm 0.40 - \pm 0.70$) according to J. P. Guilford [15] for all skills except for Evaluation and Decision Making which fall within the high range at $r = 0.73$ ($\pm 0.70 - \pm 0.90$) and Teamwork at only the first half of the semester which fall within the low range at $r = 0.38$ ($\pm 0.20 - \pm 0.40$), which is a definite but small relationship [15].

Table 4. Pearson Correlation Between Class Emphasis and Personal Growth

	First Half Survey			Second Half Survey		
	r	p-value	N	r	p-value	N
Teamwork	0.38	p < 0.001[c]	124	0.47	p < 0.001	124
Information Gathering	0.66	p < 0.001	124	0.68	p < 0.001	124
Problem Definition	0.62	p < 0.001	124	0.61	p < 0.001	124
Idea Generation	0.64	p < 0.001	124	0.57	p < 0.001	124
Evaluation and Decision Making	0.73	p < 0.001	124	0.73	p < 0.001	124
Implementation	0.59	p < 0.001	124	0.61	p < 0.001	124
Communication	0.67	p < 0.001	124	0.67	p < 0.001	124

c. As all *P* values are very small (e.g., p-value for teamwork, first half survey is 1.52716E-22), we reported them using the inequality p < 0.001 [16].

We also analyzed the Ranking of students' course emphasis and perception of personal growth using Composite Score [12]. The Composite Score is calculated by averaging the 'definition' and the 'overall performance average of individual components. More details about Composite Score can be found at Section 2 in the Module Assessment subsection. The data were analyzed by ranking the items relative to each other and looking at growth vs. emphasis and examining components that were not emphasized. The results of the skills Ranking as shown in Table 5 show that students respond that the course placed a heavy emphasis on Teamwork. They also perceived that all of the elements had a strong class emphasis but ranked the emphasis for all of the elements higher in the second half. The elements in the first half of the term were ranked at a similar emphasis. During the second half, teamwork remains significantly out in front. Information gathering followed closely behind of the Evaluation and Decision Making.

Table 5. Ranking of Student Perception of the Course Emphasis

Ranking	First Half Survey (average)	Second Half Survey (average)
1	Teamwork: 4.38	Teamwork: 4.41
2	Implementation: 4.16	Implementation: 4.25
3	Problem Definition: 4.09	Problem Definition: 4.19
4	Idea Generation: 4.04	Idea Generation: 4.09
5	Communication**:** 4.02	Communication**:** 4.03
6	Information Gathering: 3.81	Evaluation and Decision Making 3.98
7	Evaluation and Decision Making**:** 3.66	Information Gathering: 3.91

The results of the skills Ranking in Table 6 show that students indicate they had a more selective growth in each of the elements during the first half of the term, which is demonstrated by a large spread between the top and bottom scores of the elements and by the significantly different values for each of the elements. Students indicate that "Teamwork" skills were their highest growth while "Evaluation and Decision Making" was the least. During the second half of the term, the students perceived that the growth was more equal with "Teamwork" and "Implementation" the top-rated growth items.

Table 6. Ranking of Student Perception of Personal Growth

Ranking	First Half (average)	Second Half (average)
1	Teamwork: 4.14	Teamwork: 4.33
2	Implementation: 4.05	Implementation: 4.22
3	Problem Definition: 3.89	Problem Definition: 4.00
4	Idea Generation: 3.84	Idea Generation: 3.97
5	Communication: 3.83	Communication: 3.97
6	Information Gathering: 3.62	Information Gathering: 3.84
7	Evaluation and Decision Making: 3.36	Evaluation and Decision Making: 3.77

IV. DISCUSSION

Our results show that incorporating Project Based Learning had a direct effect on a student's personal growth and improvement on their parallel programming and soft skills, with a significant p-value= 0.002, and a 'large' effect size Cohen's d = 0.86. These results are especially important as Computer Science programs prepare graduates for contemporary workplaces. If students do not value Project Based Learning opportunities, it is likely that they are not adequately prepared (particularly in terms of team orientation) to successfully negotiate with others in their professional positions. In addition, as the CSC 3210 class is taken at the midpoint of the academic program, incorporating Project Based Learning can serve as a mini-capstone that provides an opportunity for the student to learn the soft skills required to be successful and then hone these skills in the final capstone courses, such as software engineering. Furthermore, with the provided material, guidance from the instructor, and the use of collaborative learning, students can learn by themselves and apply the fundamentals of parallel programming and soft skills without the need for lectures, labs, or workshops. This enables us to retain the full content of the CSC 3210 course.

The Pearson correlation overall results between the class Emphasis and students' Personal Growth in the first and second half of the semester (Table 4) were positive and highly statistically significant at $p < 0.001$ level for all the skills. The correlations strength for five skills fall within moderate range whereas Evaluation and Decision Making as well as Teamwork at only the first half of the semester skills fall within the high range and low range respectively. Overall, the correlation results indicate that the more the instructor emphasized those skills in class, the more students applied those skills. Compared to the other skills, it was observed that the Teamwork skill scores the lowest in terms of correlation strength in the first half of semester 0.38 and in the second half of semester 0.47. To improve this relationship, we might need to provide more material and incorporate one or two more exercises about Teamwork basics in other assignments instead of having Teamwork basics incorporated in only one assignment. We will then compare the results after this addition with the current results (Fall 2018) and see if this improves the relationship between Course Emphasis and Personal Growth.

The results from skills Ranking of student perception of the Course Emphasis (Table 5) and Personal Growth (Table 6) show as students' knowledge of the process increased, they are better able to judge the growth in their learning and are more discerning about how their learning in the second half of the course relates to the elements and components. This was shown by students' perception of a greater emphasis and a greater growth during the second half of the term compared to the first half of the semester even though the class teaching methodology used in each half were similar. It should be noted that students' perception of course emphasis is almost always higher than perceived growth and this has also been found in other comparable results [12]. However, the one exception was in "Implementation" which had almost no differences between perceived growth and class emphasis during the second half (0.03). This can be attributed to the fact that there was a larger growth in implementation skills during the second half of the semester than in the first half. This is expected as students had developed more parallel programs (four programs) in the second half than in the first half where students had only developed one program. According to [12], only if the difference between perceived emphasis and perceived growth is larger than 0.2 should an effort be initiated to modify the course design and delivery to promote higher levels of skill development.

V. CONCLUSION AND FUTURE WORK

In this study, we investigated whether incorporating Project Based Learning module into CSC3210 Computer Organization and Programming curriculum could help develop students' parallel programming and soft skills. In this module, comprising of five assignments, each of two-week duration, 124 students were split into 26 diverse groups (four or five per group). Working in groups, they collaboratively learned and applied parallel programming concepts and soft skills. Our results show that incorporating a Project Based Learning module had a significant and direct effect on a student's personal growth and improvement on their parallel programming and soft skills.

Our efforts show that practical projects, team learning and team work are beneficial ways to cover parallel programming. The side benefit of this approach demonstrates that this new topic can be introduced outside of the lecture time and takes away the need to reduce any existing content coverage. By having students work in groups outside of the class time, they demonstrated the ability to apply newly developed parallel programming skills.

We have observed that the module emphasis on the Teamwork skills needed to be improved. Thus, in Spring 2019, we will incorporate one or two more tasks about Teamwork basics in assignments two to five. We also plan on developing project rubrics, as it helps improve students' learning, identify what quality work is, and reduce the assignments grading overheads. As students demonstrated a strong ability to write parallel programs in multicore processors and shared memory using OpenMP and C, we plan to extend the module to include writing code for multicore processors and distributed memory using Message Passing Interface (MPI) and C language. This would provide students with more flexibility in determining the correct memory architecture to use to solve a variety of problems. Our intent is to begin by using the material provided by CSinParallel [17] and by S. K. Prasad, A. Gupta, A. L. Rosenberg, A. Sussman and C. C. Weems [18].

ACKNOWLEDGMENT

We would like to thank the Center for Excellence in Teaching and Learning (CETL) at Georgia State University for the Mini-Grant that helped us in buying the Raspberry PI kits which was very crucial and a key component to the success of developing this module.

References

[1] The Joint Task Force on Computing Curricula, "Curriculum Guidelines for Undergraduate Degree Programs in Computer Science," Association for Computing Machinery (ACM) IEEE Computer Society, 2013.

[2] CSinParallel, "Welcome to CSinParallel," National Science Foundation (NSF) and National Association of Geoscience Teachers, [Online]. Available: https://csinparallel.org/index.html. [Accessed 1 1 2019].

[3] S. J. Matthews, R. A. Brown, J. C. Adams and E. Shoop, "Portable Parallel Computing with the Raspberry Pi," in In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 2018.

[4] A. C. Burrows and M. Borowczak, "Hardening Freshman Engineering Student Soft Skills," in Session W1A First Year Engineering Experience (FYEE) Conference, 2017.

[5] Buck Institute for Education (BIE), "What is Project Based Learning?" [Online]. Available: http://www.bie.org/images/uploads/general/20fa7d42c216e2ec171a212e97fd4a9e.pdf. [Accessed 5 June 2018].

[6] MITOPENCOURSEWARE, "Lecture Notes,Sloan Communication Program," 15 8 2012. [Online]. Available: https://ocw.mit.edu/courses/sloan-school-of-management/15-279-management-communication-for-undergraduates-fall-2012/lecture-notes/MIT15_279F12_tmwrkBasics.pdf. [Accessed 22 12 2018].

[7] S. Matthews, J. Adams and D. Brown, "CSinParallel: Parallel Computing in the Computer Science Curriculum Workshops," 11 3 2017. [Online]. Available: https://csinparallel.org/csinparallel/workshops/SIGCSE17_rpi_workshop.html. [Accessed 3 7 2018].

[8] CSinParallel, "Shared Memory Parallel Patternlets in OpenMP," [Online]. Available: http://selkie.macalester.edu/csinparallel/modules/Patternlets/build/html/SharedMemory/OpenMP_Patternlets.html. [Accessed 10 8 2018].

[9] B. Barney, "Introduction to Parallel Computing," Lawrence Livermore National Laboratory, [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/. [Accessed 5 7 2018].

[10] N. Zlatanov, "CPU vs. SOC - The battle for the future of computing," in International System-on-Chip Conference at UCI, Irvine, CA, 2015.

[11] Google, "Introduction to Parallel Programming and MapReduce," 2007. [Online]. Available: https://courses.cs.washington.edu/courses/cse490h/07wi/readings/IntroductionToParallelProgrammingAndMapReduce.pdf. [Accessed 9 10 2018].

[12] S. Beyerlein, E. Davishahl, D. Davis, J. Lyons and K. Gentili, "Measuring Added Value Using A Team Design Skills Growth Survey," in American Society for Engineering Education Conference, Portland, Oregon, 2005.

[13] D. Lakens, "Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs," Frontiers in psychology, vol. 4, p. 863, 26 11 2013.

[14] B. Oakley, R. M. Felder and R. Brent, "Turning Student Groups into Effective Teams," The Journal of Student-Centered Learning, vol. 2, no. 1, pp. 9-34, 2004.

[15] J. P. Guilford, Fundamental statistics in psychology and education, New York, NY: McGraw-Hill, 1956.

[16] S. Greenland, J. S. Stephen, J. R. Kenneth, B. C. John, P. Charles, S. N. Goodman and D. G. Altman, "Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations," European journal of epidemiology 31, no. 4, pp. 337-350, 2016.

[17] CSinParallel, "Getting Started with Message Passing using MPI," [Online]. Available: http://selkie-macalester.org/csinparallel/modules/MPIProgramming/build/html/GettingStartedWithMPI.html. [Accessed 30 12 2018].

[18] S. K. Prasad, A. Gupta, A. L. Rosenberg, A. Sussman and C. C. Weems, Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses 1st Edition, Waltham, Massachusetts: Elsevier Inc, 2015.