2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #38390

The Combination Approach: Increasing Student Learning and Understanding of Introductory Computer Science Topics

Mr. Thomas Rossi, Penn State Behrend

Thomas Rossi is a lecturer in Computer Science and Software Engineering at Penn State Behrend. His research focuses on improving the post-secondary experience for students through the use of current computing tools and technologies. Thomas graduated with his MS in Computer Science from the University of New Hampshire in 2016.

Dr. Paul C. Lynch, Penn State Behrend

Paul C. Lynch received his Ph.D., M.S., and B.S. degrees in Industrial Engineering from the Pennsylvania State University. Dr. Lynch is a member of AFS, AIST, SME, IISE, and ASEE. Dr. Lynch's primary research interests are in metal casting, manufacturing systems, engineering economy and engineering education. Dr. Lynch has been recognized by Alpha Pi Mu, IISE, and the Pennsylvania State University for his scholarship, teaching, and advising. He was awarded the Penn State Behrend School of Engineering Distinguished Awards for Excellence in Advising (2018), Teaching (2019), and Research (2020). Dr. Lynch was also awarded the Penn State Behrend College Awards for Excellence in Advising (2018), Teaching (2019), and Outreach (2021). He received the Institute of Industrial and Systems Engineers Engineering Economy Teaching Award in 2018. Dr. Lynch received the Outstanding Industrial Engineering Faculty Award in 2011, 2013, and 2015, the Penn State Industrial & Manufacturing Engineering Alumni Faculty Appreciation Award in 2013, and the Outstanding Advising Award in the College of Engineering in 2014 for his work in undergraduate education at Penn State. He worked as a regional production engineer for Universal Forest Products prior to pursuing his graduate degrees. He is currently an Associate Professor of Industrial Engineering in the School of Engineering at Penn State Erie, The Behrend College.

The Combination Approach: Increasing Student Learning and Understanding of Introductory Computer Science Topics

Abstract

One of the key components to an introductory Computer Science course is the lab component. This serves as a time for students to gain hands on experience with the concepts they are learning in lecture that week. Typically, the way the lab time is structured is students will be given the assignment and be allotted the entire lab period to work on their own with instructor help available if need be.

While straightforward enough, this approach is less than ideal. With lab sizes in introductory courses increasing the number of students who need instructor help during the lab time increases. This approach leads to students not being able to get the attention they need as the instructor needs to move between students quickly or even worse…students may "fall through the cracks" as demand for help outpaces the instructor resources available. The result is students leaving lab with knowledge gaps regarding the topic that prevent them from creating a solid foundation on which to build their basic programming knowledge. Even worse is the fact this approach teaches students when they are handed a programming task to dive straight to code as fast as possible which may not be consistent with how they will work in industry.

The goal of this paper is to outline a new paradigm for structuring the lab period which teaches students how to work with peers to solve a problem, think before they code, and build conceptual understanding. In this approach students do a combination of group work, individual work, and whole class work to solve the problem. This allows the instructor to better manage the students in the class and enables them to point out common "pain points" with the material being covered that week and show ways to optimize / speed up the code being written.

This paper discusses the effectiveness of this approach by looking at qualitative student feedback as well as analyzing the student performance (grades) across sections of the class that had this new, combination approach versus the normal approach to lab of giving students the lab and having them work independently. The initial statistical analysis (difference in means, 95% confidence level) shows a statistically significant increase in lab, homework, and overall course grades for students that experienced the new, combination approach when compared to the students that experienced the normal approach to the computer science lab experience.

1.0. Introduction

One of the key components to an introductory Computer Science course is the lab component. It is here that students get hands on with the concepts being discussed that week in the lecture. This hands-on time is especially important in introductory programming classes where many students are getting their first exposure to programming. The challenge though is how to structure this time effectively to ensure that students get as much out of it as possible.

In prior semesters the format for such time was to begin with a quiz on the previous week's material and then have students work on the lab individually. A faculty member would be present to field questions and troubleshoot issues students may face while they are working on their solution to the lab assignment. This proved to be problematic though as it resulted in some significant issues as enrollments increased in entry level Computer Science classes:

1. Students would not be able to get the attention they needed
2. Students would "fall through the cracks" since there was only one faculty member in the class

Clearly, there was room for improvement. In redesigning the lab experience for this introductory class, there were a few goals in mind:

1. Teach students how to work with peers in Computer Science
2. Get students to think about what they were going to code before they coded it
3. Build conceptual understanding

This paper outlines the revised approach taken because of these goals and discusses the results of this approach in an introductory programming course. The hope is this work will serve as a template that can be applied in other schools to help improve understanding of early Computer Science concepts.

2.0. Background

As argued by Penny et al, lab activities exist to help students understand the intangible concepts they may be dealing with in the lecture portion of a class [1]. This is typically done by the student completing a lab assignment / experiment where they build a small program using the topic being discussed that week. For example, if a student is learning about loops in lecture, they could be asked to write a program using loops to generate a multiplication table during the lab period. Hazzan et al assert this allows students to be engaged in their learning rather than a bystander similar to what you might see in laboratories for the natural sciences [2].

Prior engineering education research has clearly shown that inductive teaching styles in lectures and lab sessions show the students the importance and application of the subject matter by showing the students particular examples while challenging them to keep building concept by concept to solve complex challenges [3] [4]. These inductive teaching methods typically use a scaffolded approach to lecture and lab teaching methods where inquiry learning, problem-based learning, and project based learning are utilized. This scaffolded approach utilized in inductive teaching is more student centered than the traditional deductive approach where topic generalities and mathematical proofs are covered in the class followed by homework outside of

the classroom [4] [5]. Utilizing this inductive teaching approach with a scaffolded approach, utilizing multiple, active learning focused teaching methods, mastery of the concepts is now part of the learning process as students actively work through problems or projects [5].

It has also been shown in engineering education research that student satisfaction, self-efficacy, and motivation are all significant parts of overall student perception of their learning environment and thus affects their learning of their subject material being taught. Students gain satisfaction in lecture and lab periods in many different manners, some of which include achieving through actively working through problems and projects while receiving guidance and positive reinforcement for their work by their instructor. In addition, being able to interact freely and comfortably with their instructor helps drive student satisfaction with their courses [6] [7]. Helping the students through active learning lecture and lab sessions can help students build strong belief in their capabilities as they work through the challenges of solving the problem or completing the project. The hands on, positive reinforcement given to them by their instructors during these activities can help to increase self-efficacy, which is known to influence the amount of effort students will put into their classes [8] [9]. By increasing student satisfaction with their classroom or lab experience and building up student self-efficacy beliefs, instructors an ultimately improve student motivation and the will to work harder in the classroom or laboratory [10].

The problem though is how labs can be poorly administered resulting in situations where instructors are monopolized and the time devolves into a study hall type session where the assignment is more like a homework (deductive approach) rather than an active practice session (inductive approach) [11]. Lab time needs to be used in such a way that benefits the student. They need to be growing their skills as programmers so they are ready to take on the more complex challenges that lay ahead. Additionally, one other main step in the process students need to spend time on from the very beginning as noted by Proulx et al is program design [12]. Students need to be able to build solutions that are well thought out and not just something that works.

Clearly there is room for improvement in the lab space. What seeks to be a way of helping students gain practical experience and be involved in their learning can become a study hall type environment with no clear benefit if it is not carried out with these active learning and inductive pedagogical methods put into place. Moreover, this can lead to a missed opportunity at the Computer Science 1 (CS 1) / Computer Science 2 (CS 2) level to begin having them think about program design.

3.0. Implementation

Given the goals outlined in the introduction, the introductory Computer Science course lab was restructured. In this restructuring the lab was broken into two key components:

1. Algorithm Generation
2. Implementation

3.1 Algorithm Generation

After the weekly quiz, the first ten minutes of the lab period is dedicated to algorithm generation. Students begin by working in small groups to develop a series of steps to solve the programming assignment. This is done in English with the students writing a set of steps to solve the problem instead of pseudocode or a flowchart as these topics are not taught in the introductory course. Each student is expected to submit either a picture or PDF copy of this as part of their lab submission. The algorithm is graded based on the student's attempt rather than correctness and is worth two points out of the twenty the lab is worth in total.

After the ten minutes have elapsed, the whole class works with the instructor on the chalk / white board to come up with a class algorithm to solve the problem. The instructor will point out areas where the students may have steps missing, out of order, or may need to break given information down further to capture all the necessary details of the problem. Once the class algorithm has been determined, the focus shifts to implementation.

3.2 Implementation

At the beginning of the implementation phase, the instructor will ask the students to work on a subset of the steps from the algorithm. Depending on the work involved with those steps, the students will get somewhere between ten and fifteen minutes to complete this portion of the implementation. At the end of this time, the instructor will use a computer that is connected to the lab's projector to walk through the steps the students just worked on with their input as to what they did to accomplish each step. The goal here is to give students more immediate feedback as to whether they were on the right track as well as to assist students who struggled to complete the steps indicated. Additionally, the instructor can use this opportunity to point out common *pain points* they have seen students have with that week's topic. Previous work has clearly shown that instructor interaction and feedback is the main variable that significantly impacts the student learning outcome ratings in undergraduate lectures and labs [13].

After the steps have been discussed, the instructor will then indicate the next set of steps they would like students to work on as well as give them an allotment of time in which to complete those steps. As before, the instructor will go over those steps when the time has elapsed. This process repeats until all steps have been completed. Students submit the completed code as part of their lab submission for up to eighteen points.

3.3 Applying Methodology in Other Courses

While this modified approach is used in CS 1, this approach is not used in the intermediate Computer Science course. This approach is feasible in CS 1, but it needs work before it can be adopted in CS 2. Given CS 2 is designed for students who now have prior experience coding it doesn't make as much sense for the instructor to be discussing the entire implementation, going back to the beginning topics in CS 1. Similarly, in higher level classes it could potentially come across as insulting to students if they are given the same level of guidance as they received in CS 1 in lab. One possible way of addressing this would be to have the students be responsible for more of the lab on their own as they go further along in the major to the point where by their

advanced coursework they are working almost completely on their own, with minimal instructor guidance.

4.0. Results

4.1 Quantitative Results

Quantitative data (lab, homework, quiz, and overall scores) was collected from the ***Normal Treatment*** or ***Traditional Approach*** and from the ***Experimental Treatment*** or ***New Approach*** to delivering the lab periods for the introductory Computer Science course. To diminish any additional variables in the data sets, the same instructor was in place for both the ***Traditional Approach*** and the ***New Approach***. The data collected for the ***Traditional Approach*** was collected in the school year prior to the ***New Approach*** data being collected. The quantitative (grade) data was collected for all (27) students in the ***Traditional Approach*** lab sections of the course. The quantitative (grade) data was collected for all (63) students in the ***New Approach*** lab sections of the course. A summary of the ***Traditional Approach*** and ***New Approach*** grade statistical data is shown in Table 1 & 2 respectively below:

Table 1: Grade Statistics (%) for the ***Traditional Computer Science Lab Approach***

Grades:	Lab	Homework	Quiz	Overall
AVG	81.57	75.32	68.36	71.67
STDEV	17.67	19.78	11.26	14.47

Table 2: Grade Statistics (%) for the ***New Computer Science Lab Approach***

Grades:	Lab	Homework	Quiz	Overall
AVG	92.01	85.13	64.44	83.80
STDEV	15.04	19.58	17.11	15.93

In an effort to test whether or not the new paradigm for structuring the lab period in the ***New Approach*** had a significant effect on student learning and performance, a statistical analysis was carried out on the grade statistics in the ***Traditional Approach*** and the ***New Approach***. At the 95% confidence level, difference in means calculations were carried out for all four sets of grade statistics (lab, homework, quiz, overall). The variances were assumed to be unknown and unequal for the difference in means calculations. The following formulas (Equations 1, 2, and 3 in Table 3) were used to calculate the difference in means where the ***New Approach*** was sample A and the ***Traditional Approach*** was sample B.

Table 3: Formulas for Difference in Means Calculations (95% Confidence Level)

Difference Between Two Means (Unknown and Unequal Variances)	
Equation 1: Degrees of Freedom	$$DOF = \frac{\left(\frac{s_A^2}{n_A}+\frac{s_B^2}{n_B}\right)^2}{\frac{\left[\frac{s_A^2}{n_A}\right]^2}{(n_A-1)}+\frac{\left[\frac{s_B^2}{n_B}\right]^2}{(n_B-1)}}$$
Equation 2: t value	$$t = t\alpha_{/2,DOF}$$
Equation 3: Confidence Interval	$$P\left[(\overline{X}_A-\overline{X}_B)-t*\sqrt{\frac{s_A^2}{n_A}+\frac{s_B^2}{n_B}}<\mu_A-\mu_B<(\overline{X}_A-\overline{X}_B)+t*\sqrt{\frac{s_A^2}{n_A}+\frac{s_B^2}{n_B}}\right]=1-\alpha$$

The 95% confidence intervals for the difference in means between the grade statistics for the ***New Approach*** (sample A) and the ***Traditional Approach*** (sample B) are shown in Table 4 below.

Table 4: Difference in Means Calculations (95% Confidence Level: Sample A – Sample B)

Grades:	**Lab**	**Homework**	**Quiz**	**Overall**
Upper Limit (95%)	18.30	18.98	2.20	19.06
Lower Limit (95%)	2.56	0.64	-10.03	5.19

Since this paper specifically addresses the lab portion of this course, it is important to mention how each lab session is graded. Each lab session is graded out of a total of 20 points. 18 out of the 20 points are allotted to writing the program and solving the problem that was assigned and worked on in the lab session. 2 out of the 20 points are assigned based upon attendance. In other words, as long a student attends the lab session and participates in the algorithm generation, the student receives the 2 points.

The results in Table 4 show a statistically significant increase in lab, homework, and overall course grades for students that experienced the experimental treatment or new approach when compared to the students that experienced the normal treatment or traditional approach to the computer science lab experience. There was no statistically significant change in quiz grades observed between the traditional and new approach.

4.2 Qualitative Results

To get feedback directly from students regarding this new lab approach, student end of semester evaluations were used. These evaluations are anonymous to the instructor and there is no mechanism through which the instructor can de-anonymize the feedback. Student feedback regarding this new approach to lab has been positive with students indicating this has helped them understand the concepts being presented each week in the lecture. Students have also enjoyed getting to interact with their peers during the lab period [13]. Students do still have some misgivings about the lab component of the course, but these are not germane to the topic of this paper.

5.0. Conclusion and Future Work

In this paper, a new combination approach for teaching lab has been discussed. This approach has shown a statistically significant improvement in student performance on lab, homework, and their overall grade which shows this approach does have merit. Moreover, the student feedback on end of semester surveys shows students do in fact see the merit of this approach and feel they have benefited from it regarding how ready they are to approach the homework for that week in the course. In the future, it would be interesting to see if by changing to more industry standard ways of planning code such as the use of pseudocode or flowcharts would impact the student outcomes.

6.0 Works Cited

[1] J. P. Penny and P. J. Ashton, "Laboratory-style teaching of computer science," *ACM SIGCSE Bulletin,* vol. 22, no. 1, pp. 192-196, 1990.

[2] O. Hazzan, N. Ragonis and T. Lapidot, Guide to Teaching Computer Science, Cham: Springer, 2020.

[3] M. Prince and R. Felder, "The Many Faces of Inductive Teaching and Learning," *J. College Science and Teaching,* vol. 36, no. 5, pp. 14-20, 2007.

[4] M. Prince and R. Felder, "Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases," *Journal of College Teaching,* vol. 36, no. 5, pp. 14-20, 2007.

[5] T. Ruutman and H. Kipper, "Teaching Strategies for Direct and Indirect Instruciton in Teaching Engineering," in *Proceedings of 14th International Conference on Interactive Collaborative Learning*, Slovakia, 2011.

[6] A. Poulsen, K. Lam, S. Cisneros and T. Treust, "ARCS Model of Mtivational Design," November 2008. [Online]. [Accessed December 2014].

[7] S. Bjorklund, J. Parente and D. Sathianathan, "Effects of Faculty Interaction and Feedback on Gains in Student Skills," *Journal of Engieering Education,* vol. 93, no. 2, pp. 153-160, 2004.

[8] P. Hsieh, J. R. Sullivan and N. S. Guerra, "A Closer Look at College Students: Self-Efficacy and Goal Orientation," *Journal of Advanced Academics,* vol. 18, no. 3, pp. 454-476, 2007.

[9] M. M. Chemers, L.-T. Hu and B. F. Garcia, "Academic Self-efficacy and First Year College Student Performance and Adjustment," *Journal of Educational Psychology,* vol. 93, no. 1, pp. 55-64, 2001.

[10] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett and M. K. Norman, "What Factors Motivate Students to Learn?" How Learning Works: Seven Research-Based Principles for Smart Teaching, San Francisco, CA: Jossey-Bass, 2010, pp. 66-90.

[11] T. A. Beaubouef and J. Mason, "Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations," *ACM SIGCSE Bulletin,* vol. 37, no. 2, pp. 103-106, 2005.

[12] V. K. Proulx, R. Rasala and H. Fell, " Foundations of Computer Science: What are they and how do we teach them?," *ACM SIGCSE Bulletin,* vol. 28, no. SI, pp. 42-48, 1996.

[13] Schreyer Institute for Teaching Excellence, "SRTE," 2023. [Online]. Available: srte.psu.edu. [Accessed 11 February 2023].