Student Engagement, Learning, and Retention in a Freshman, Large Class Setting at the University of Arizona

Loukas Lazos and Elmer Grubbs
University of Arizona

Abstract

This paper describes a new approach to teaching a large lecture *C* Programming class (ECE175) in the Electrical and Computer Engineering Department at the University of Arizona. The approach demonstrates a method of increasing student engagement, student learning and student retention by using Undergraduate Laboratory Assistants (ULAs) in conjunction with Graduate Teaching Assistants (TAs) in a laboratory environment. The class typically has an enrollment of 175 to 230 students per semester and is a service course for students outside of the major, in addition to ECE majors. In the past, the class had an optional attendance laboratory component, staffed by several TA's, who offered homework assistance to students. In the revised course structure, the laboratory component has been transformed to a mandatory attendance recitation/activity/grading 3-hour lab, with 5 sections of up to 49 students each. The lab space is equipped with 49 workstations, all linked to a single computer controlled by the instructor. Each lab section is staffed by at least two TAs and 4 to 8 ULAs, depending on the lab enrollment. Prior to attending the lab, students submit their weekly homework assignment on the DesireToLearn (D2L) online submission system. Weekly homework consists of one to three programming problems related to the main lecture material and a series of multiple-choice, true/false, and short "human-compiled" code questions taken from the online interactive course textbook. In the lab, each individual student is graded by a ULA for a period of 15 minutes on his prior HW submission. Grading is based on a rubric designed to test a) the completeness, correctness, and code efficiency and b) the student's conceptual understanding on the code that he/she developed and related software engineering concepts. The lab also consists of one or two additional programming assignments, which must be completed within the remaining lab time. Initial experiences with the revised course structure indicate that the lab portion has improved student engagement, learning, and retention. Statistical analysis from the online textbook, which has a series of computer graded exercises, final grades, and number of drops per class are used to evaluate whether student learning and retention has been increased.

Introduction

ECE175 is an introductory *C* programming course, which has been taught in the Electrical and Computer Engineering (ECE) Department at the University of Arizona for many years. It has always had enrollment from non-majors, but the last four years it has become a mandatory course for most College of Engineering students. As a result, the majority of enrollees are non-ECE majors. Enrollment has varied over the years from 30 to 40 students up to recent class sizes of 180 to 230 students per semester. It is taught twice a year in one lecture section and multiple laboratory sections per semester. The *C* language was originally chosen, not because it is necessarily the best, most useful language around, but because it serves as a good basic language, which allows students to rather easily transfer over to other languages as needed. It is also still highly useful in industry. The course used to be a traditional lecture course, using a standard text

on *C* Programming and another text on UNIX, where the lecture was used to go over programming concepts and work example problems for the students. Students were assigned homework problems, which were all programs. The class had no lab component, but TAs were assigned to hold extensive office hours in a computer lab to help the students with homework on an as needed basis. The lab was typically underused most of the semester, but became crowded at times just before the programs were due. There were two semester exams, and a final, team-based project. Most ECE majors seemed to do fairly well in the class, some with minimal TA interaction. As the class became larger, the TA interactions became somewhat problematic because of the number of students needing help right before the due date, as well as with academic integrity issues.

The instructors were faced with the problem of redesigning the class to enhance the student learning experience by increasing student engagement. Subsequently, it was decided to "flip the class[1]." The idea was to make these changes in a way that would accommodate larger numbers of non-majors and to make the class more student-friendly. It was thought that this would increase retention, enhance student involvement, promote better understanding of the class material, and deter violations of the academic integrity policy.

The class was re-organized to use the lecture for explaining concepts, working problems by writing programs as well as using active learning techniques. A set of PowerPoint lectures were put on-line for the students to view as needed. An on-line book also was added and a non-Unix windows operating system using Visual Studio was introduced. A new three-hour per week mandatory laboratory was added, which was staffed by TAs and ULAs. This was held in an updated computer laboratory. These changes will be discussed in detail in the next section of the paper.

Basic Course Design

The first part of the idea behind flipping a class is to provide out-of-class activities that are essentially equivalent to the lectures that are normally presented in a classroom setting. The students are instructed to review these lectures during the time they would normally be studying at home. The second part of flipping, which is in the classroom and lab, consists of going over concepts and working problems with the students, either on a chalkboard or a screen or individually with each student. In ECE175, a series of PowerPoint lectures were designed to be used by the students out of class. These lectures cover the material normally lectured on by the instructor in previous years. An on-line book from Zyante (see Figure 1) was also added to the class. In addition to the reading, the book contains exercises that are worked by the individual students outside of class to help them understand the material covered by the book and in lectures. The book software keeps track of the students' progress on a set of interactive exercises listed at the end of each section and of the number of each student's attempts until an exercise is correctly completed. We use the results of these exercises to give a participation grade to the students. In addition to the online book, the course has a main website where the syllabus, lectures, etc. are stored (see Figure 2) and an enhanced discussion board website called Piazza (see Figure 3) that implements a class discussion forum between students, instructors, TAs, and ULAs.

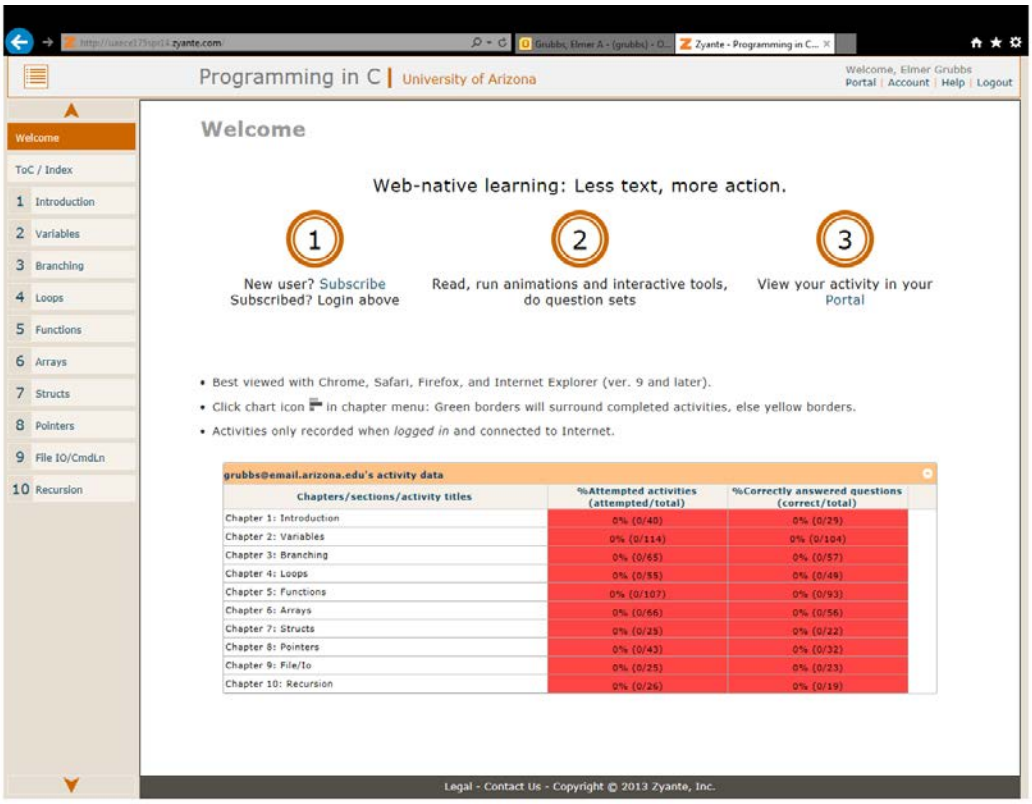

Figure 1. The Zyante online interactive book website.

Figure 2. The course website.

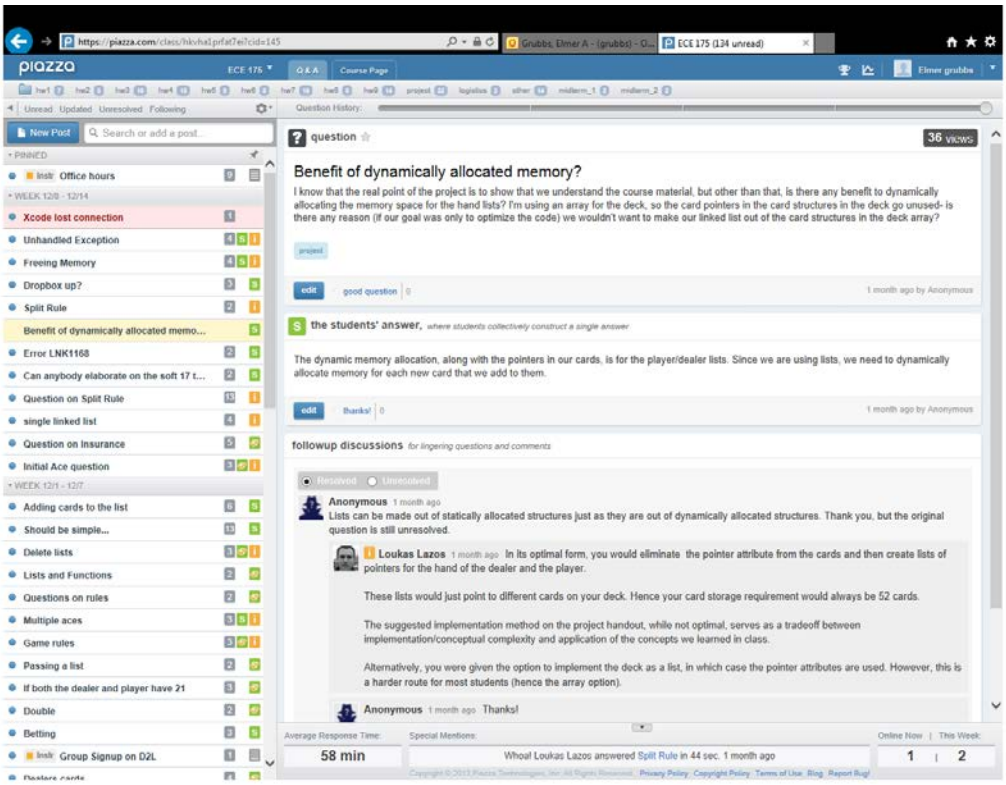

Figure 3. The Piazza website.

Besides the traditional in-class lecture, the class consists of multiple laboratory sections of up to 49 students each. Each session is staffed by at least two TAs and four to eight ULAs. The TAs are graduate ECE students and the ULAs are undergraduate ECE students who have previously taken the class and excelled. These sessions take place in a computer lab with fifty workstations, all networked with the instructor's computer at the front of the room. The lab sections run from Wednesday of each week through Friday. There are typically five sections.

Homework is assigned every Tuesday and is due the following Tuesday by midnight. Prior to attending the lab, students submit their weekly homework assignment on the DesireToLearn (D2L) online submission system. Weekly homework consists of one to three programming problems related to the main lecture material and a series of multiple-choice, true/false, and short "human-compiled" code questions taken from the online interactive course textbook. In the lab, each individual student is graded by a ULA for a period of 15 minutes on his prior HW submission. Grading is based on a rubric designed to test a) the completeness, correctness, and code efficiency and b) the student's conceptual understanding on the code that he/she developed and related software engineering concepts. The rubric, in addition to correct answers, contains specific questions to ensure that understanding of the presented solution. A sample homework problem and its corresponding rubric are shown in Figures 4 and 5, respectively. A sample question set problem from the interactive course textbook is shown in Figure 6. Figure 7 shows a copy of the group project from last year. The appendix contains a copy of the syllabus.

The lab also consists of one or two additional programming assignments, which must be completed within the remaining lab time. Students work on the in-lab assignments with the

assistance of the TAs and ULAs for the remaining lab period. The problems are different for each lab section per week. When a student completes a problem, the ULA comes over to them and grades the problem for them. They must finish prior to the end of the lab period. The students are able to work together on these problems and it is not unusual for some students to finish early and then help the other students to finish the problems. So the lab is really a hands-on work session with peer-learning experiences built-in. We believe this laboratory helps the students out by engaging them more in the process than the typical lab experience might accomplish.

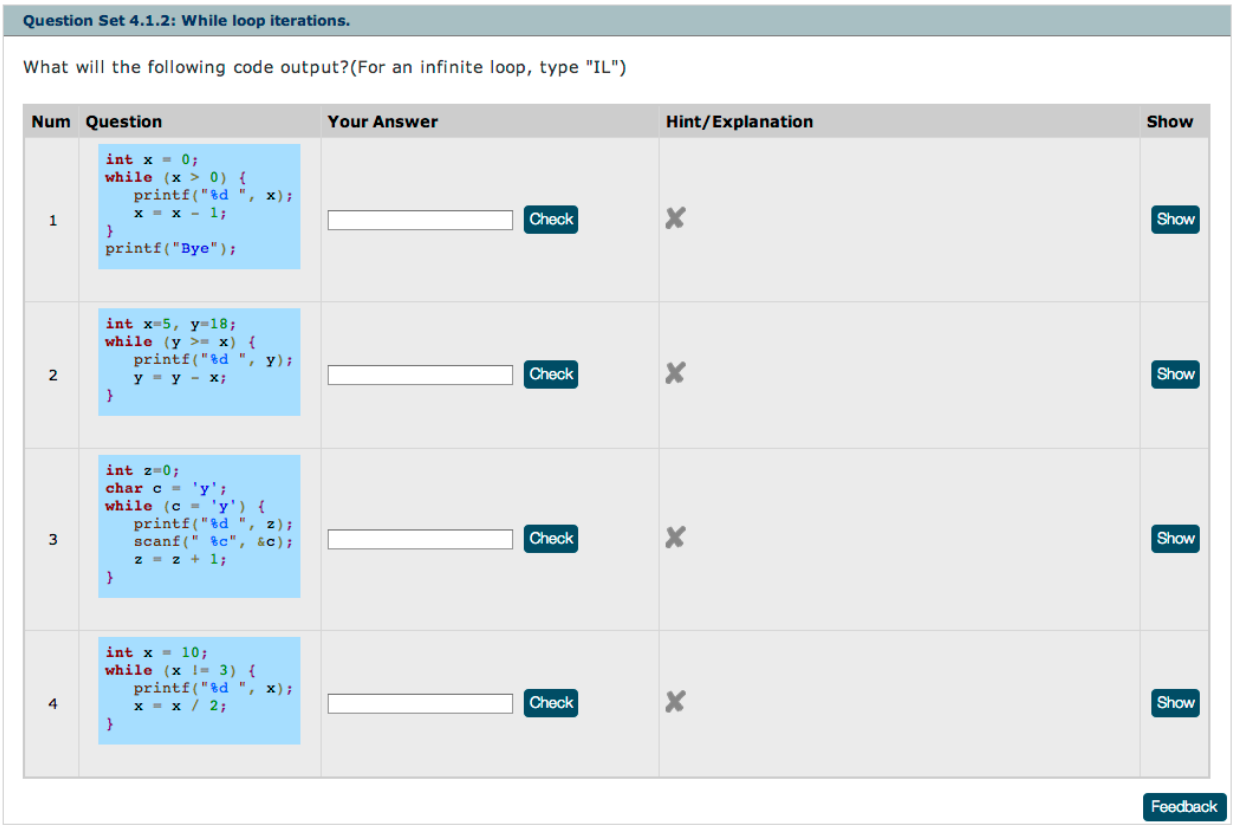

Figure 6. Online book question set

Results

The objectives of the course redesign were:

1. To increase student learning
2. To enhance student involvement
3. To increase retention
4. To ensure students were doing their own work and fully understand their solutions

The goals of enhancing student involvement and making sure that students were doing their own work were accomplished directly by the changes in the class structure. Obviously the students are more involved in the lab since attendance is mandatory, making sure that they work directly with TAs, ULAs and their peers, and assigning a significant portion (60%) of their overall grade to the lab component. The individual oral grading of the students' homework and lab work, using

the professor-supplied rubric, helps to ensure that each student, at the least, understands the work that he/she turned in and has a good understanding of the fundamental concepts. Moreover, during the oral examination, the ULAs offer additional *individual* explanations and demonstrations to cover students' conceptual and technical deficiencies.

Final Project ECE175 Spring 2013
The final project is to write a c program to simulate the card game called "War". The rules are:

1. You are to play with a deck of 52 cards, ranking from low to high as follows: two, three, …ten, jack, queen, king and ace. There are 4 suits (H S D C) of each numbered and face cards, although for the game, suits are ignored. The deck needs to be shuffled, then deal 26 cards to each of two players (This is called a hand).
2. Each player places the top card face up on the table and the player with the higher card takes the two cards and places it in his discount pile (he wins the first round of the game). If the two cards are identical, each player places three cards face down on top of their first card, then one more card face up on top of their pile. Again, the highest card takes the entire two hands. This is called a war.
3. Each player when he or she runs out of cards, then uses the discount pile to form a new hand to play the game from. The cards are not reshuffled when going from the discount pile to a hand.
4. The first player to totally run out of cards loses the game. If you run out in the middle of a war, you still lose.
5. Shuffle the cards using the rand[] function.
6. Represent your hand as a linked list of cards typedef struct card_s {
   Use a typedef struct to define the cards.char suit; int face; struct card_s *listp; } card;
7. At the beginning of each round, you must display the top card for each person, indicate whether there is a war on not, and display the total number of cards still in each person's possession. Upon entering a return key, the game will play the next round. The program should announce the winner at the end of the game.
8. The display should look like the following:

```
      ******                  ******
      *    *                  *    *
      *  8 *                  *  J *
      *    *                  *    *
      ******                  ******
No War            20 cards          32 cards
Next Round
```

Honor code: You are expected to submit your own code. You may ask others for advice, and in general discuss the project, but you should WRITE YOUR OWN CODE. If any part of the code submitted by different teams is identical, ALL involved parties will receive 0 credit on the project and one letter reduction in their final grade. The project is to be worked in teams of two students each, and you should use modular programming, documentation, etc.

Increased retention can be measured by computing the student percentage that drops the class and comparing it with data from class offerings prior to the implemented changes. Student learning can be measured by comparing the percentages of students in each grade category with previous ECE175 class statistics. Refer to Figure 7 for a chart of these comparisons. Note that the number of students scoring 80% or above went from an average of 51.5% to 68.4%, an increase of almost 17%. The percentage of students who dropped the class decreased modestly from an average of 5.4% to 4.4%, and the percentage of students failing the class dropped from an average of 11.5% to 7.3%. We are also attempting to increase student learning by adding some video format learning activities for the students to do outside of class, which cover difficult topics in the class. The evaluation of these goals will be covered in the next section.

Future Work

The University of Arizona currently has an AAU STEM[3] (Association of American Universities Science, Technology, Engineering and Mathematics) initiative project, which this class is a part of. Our role in the overall project is to improve student learning and retention by identifying difficult topics and producing a series of video demonstrations to address these topics. Another set of videos will be used to show how expert programmers solve typical programming problems. These solutions will include program design, algorithm design, debugging, etc. The students will view these videos as needed outside of the classroom environment. We are proposing to measure the success of these videos by using problems from the online book in a before and after fashion. We will assign half of the problems on a particular topic first and measure success based on successful answers, number of attempts, etc. Then we will have them view the videos and assign the second half of the problems. By comparing the success rate and number of attempts statistics before and after the video showing, we will obtain a measure of how well the videos are working. We will be doing these measurements during the Spring and Fall 2014 semesters.

Conclusions

We have described a novel approach for teaching a large introductory programming class at the University of Arizona, which involves flipping the classroom and providing systematic TAs and ULAs assistance in a laboratory setting to increase student involvement in the educational process. We have shown that the techniques described here have increased student learning, retention, and engagement. We have also briefly described future steps for further improving student learning, which we are implementing in the next several semesters.

Bibliography

1. Noonoo, Stephen, “Flipped Learning Founders Set the Record Straight,” *T.H.E Journal*, 20 June 2012, http://thejournal.com/articles/2012/06/20/flipped-learning-founders-q-and-a.aspx.
2. D2L, Desire2Learn website, http://d2l.arizona.edu
3. AAU STEM initiative website http://www.aau.edu/policy/article.aspx?id=12588