Improving a Computer Science 1 Course with Flipped Instruction and Faculty Guides

J. Hollingsworth and D. Powell
Department of Computing Sciences, Elon University, Elon, NC, USA

Abstract - *This paper describes and discusses the advantages and disadvantages, for both students and faculty, of flipping an introductory Computer Science course by leveraging an already existing MOOC. Course content and format are described along with the new role of the faculty member (Guide) for this emerging course style. We argue that if faculty members embrace their new roles as Guides, a flipped Computer Science 1 course can be both compelling and rewarding for undergraduates.*

Keywords: Online courses, flipping the classroom

1 Introduction

MOOCs have changed and are changing the learning landscape in Computer Science. High quality, free or low cost, interactive, video based courses are available from a wide range of vendors and cover many aspects of the undergraduate Computer Science curriculum. The hypothesis of this paper is that an entry level Computer Science 1 course can be as effective or more effectively taught using flipped instruction, leveraging professionally developed content from third parties, than the traditional lecture format. For the past three years at Elon University the introductory Computer Science course called Computer Science 1, has been traditionally taught using a course text, Big Java Early Objects, with classes meeting 3 times a week in a computer lab setting [1]. Each class session typically involved a lecture followed by hands-on individual or group exercises. Faculty and students are very comfortable and successful with this model. In the spring 2015 semester the course was flipped with students out-of-class activities requiring the viewing of assigned interactive videos and in-class time spent on individual and group coding activities [2]. Faculty roles changed from lecturing the student to guiding the student. Faculty spend class time answering individual questions or leading problem solving exercises that demonstrated a programming concept or best practice. This paper describes the format and content of the course, the advantages and benefits to students and faculty, the disadvantages and the lessons learned.

2 Course Content and Format

During the spring 2014 semester, the Elon Computing Sciences Department gained experience in flipping an elective course on Advanced Programming for juniors and seniors majoring in Computer Science by curating a course with content from CodeSchool, Lynda and Udemy [3]. The experience gained here was used in developing the format and content of the Computer Science 1 course, in selecting a high quality 3rd party MOOC vendor, in selecting and developing course materials, in planning the 38 semester class session activities and in developing an individual assessment plan. Unlike the spring 2014 class made up of upper class Computer Science majors, this class was made up of freshman from a wide variety of majors including: Communications, Music, English, Business, Math, Engineering and Computer Science. Various high quality MOOC vendors such as Code School, Coursera, Lynda, Udacity and Udemy were searched for a beginner introduction to Computer Science course using Java and covering programming concepts using an objects-first approach. An Udacity course called Introduction to Java Programming was selected. This course is taught by Cay Horstmann, a professor at San Jose University, who is the author of the textbook, Big Java Early Objects, which is used in our traditional course [4]. This online course was an ideal one to choose for the transition to a flipped classroom as it followed the content of the text in sequence. The course is well designed with a large number of short, typically 1–3 minutes, high quality, professionally created videos with interspersed interactive quizzes arranged in lessons. The course also included problem sets that require Java code to be developed and submitted for online grading, forums for asking and researching already asked and answered questions and a dashboard to track completion status. The course has free course materials and has been well vetted and taken by over 200,000 people.

Table 1 shows the Udacity lessons mapped to chapters in the Big Java Early Objects book. The mapping is identical and greatly facilitated the transition from a traditional lecture based course to the flipped classroom using the Udacity materials. The syllabus is almost identical in the amount of time used to cover each topic. In fact, the Udacity materials are much more engaging and require more interaction then the static text book pages. For example, Table 1 shows that 27 videos cover the content in Book Chapter 1. These videos are all short, ranging from 14 seconds to 151 seconds, in order to keep the student fresh and actively engaged. The videos are interspersed with required fill in the blank or coding challenges that must be completed for the student to get credit for it on the course dashboard. The challenges typically require user interaction in one of every two videos. Of the 27 videos comprising lesson one 17 (63%) required a correct user response or code snippet. The code snippets are submitted through a browser window. Test cases are automatically run at Udacity and then the success or failure results are displayed in the browser. If the user cannot solve the problem then the user can choose to continue and the solution is presented and described in the next video. The lessons match the associated book chapter in an identical sequence and are by and large appropriately chunked to allow a student to complete each lesson in a two-hour period before class. A good example of the chunking is Chapter 7 on Arrays and ArrayLists. The lesson is broken into three pieces with lesson 7-1 covering ArrayLists in 44 minutes of videos, lesson 7-2 covering one-dimensional arrays in 25 minutes and lesson 7-3 covering two-dimensional arrays in 22 minutes. The proper size chunking greatly facilitates the development of the course syllabus and the out of class assignments.

As shown in Table 1, many of the lessons are followed by a problem set generally consisting of 20 shorts problems requiring input of one to four lines of code and 3 problems requiring a more substantial amount of code to solve. The user must correctly submit the code to get credit on the dashboard. The solution code is not directly provided after each problem as in the lessons. The problem sets are ideal to assign as in class activities for students to work on during a class session so they can discuss any problems with nearby classmates and the instructor. Moreover, problems sets provide a motivational tool to allow students who readily understand the material to complete it before class instead of during class and be rewarded with optional class attendance.

The use of lessons and problem sets with the direct mapping to book chapters made the transition for a traditional class easy. Our course and learning outcomes are unchanged. The same number of classes was used to cover each chapter but instead of spending time during class lecturing and classes exercises being done outside of class the student were solving problem sets and class exercises during class and watching and interacting with the videos for the lesson outside of class. In both the traditional and flipped classroom, each chapter concluded with an assessment.

In order to provide additional motivation to the students to complete the video assignments before class and the in-class problem sets before the next class, students were required to submit to the course management system (Moodle) a screen-capture of their dashboard showing the completion status of each video interactive exercise in a lesson and each problem in a problem set. The student received a score of 0 for incomplete, 1 for 25% correct, 2 for 50% correct, 3 for 75% correct and 4 for completely correct. The total weight of the snapshot completions to the course grade was 37.5%. The high percentage proved very motivating with 95% of all students completing on time all 11 lesson chunks and 8 problem sets with over 75% complete on each.

The students were still required to purchase the book Big Java Early Objects. The book was primarily used as a reference and as an easy way to review the concepts from the chapter covered in the Udacity videos. Though the interactive videos are a great way to learn, they are difficult to use as a quick and efficient review for a quiz.

Table 1: Udacity Flipped Classroom Materials Mapped to Book Chapters Used in Traditional Course

Topic	Lesson	#Videos	Number and % Interactive	Total Time Minutes	Shortest Seconds	Longest Seconds	Problem Set	Book Chapter
Intro	1	27	17 – 63%	41	14	151	0	1
Objects	2	56	27 – 48%	55	15	222	1	2
Classes	3	45	26 – 58%	57	10	173	2	3
Data Types	4	50	35 – 70%	80	6	224		4
Decisions	5.1	13	9 – 70%	23	19	41	3	5
	5.2	13	11 – 85%	14	4	200		
Loops	6.1	34	23 – 68%	42	5	128	4	6
	6.2	40	35 – 87.5%	67	8	217	5	
ArrayLists and Arrays	7.1	28	20 – 71%	44	6	168	6	7
	7.2	19	11 – 58%	25	6	143	7	
	7.3	11	6 – 55%	22	23	139		
Discovering Classes	8	27	22 – 81%	59	17	392	8	8

3 Assessment

Since the majority of class time is spent collaboratively working on problem sets and class exercises, the focus on the chapter assessment in the flipped classroom is on the individual mastery of the chapter content and being able to demonstrate the ability to apply it to solve a problem. An entire class is spent on each chapter for individual assessment. Content knowledge is tested through a 20 question multiple-choice test and individual coding assignment containing one or more problems to code using the BlueJ IDE. The students receive immediate feedback on the multiple-choice portion of the quiz. The individual coding assignment is graded and returned by the next class. This approach differs slightly from the traditional classroom coding portion which was typically done outside the classroom and in pairs.

Though the grading of hands on coding has changed from pair homeworks to the in class individual coding problems, the quiz questions were kept the same as the previous traditional course for Chapters 1 – 4 and 6 – 8. Note: Chapter 5 in the traditional course did not have a multiple-choice quiz but used an alternative assessment mechanism. The intent was to use identical quizzes to provide comparative data on concept mastery between the flipped and the traditional model. Table 2 shows the average quiz grades of 29 students who took Computer Science 1 with the traditional approach in spring 2013 and 27 students who took Computer Science 1 with the flipped classroom in spring 2015. Both classes had the same instructor and the same book. Though only a comparison with one class, the data indicates that the flipped classroom results are equal to those of the traditional classroom. Since the resulting outcomes are similar or slightly improved, let us consider the advantages or benefits and disadvantages of the flipped classroom for students and faculty.

4 Student Advantages and Benefits of the Flipped Classroom

The spring 2015 flipping of the Computer Science 1 course provided students the following benefits and advantages:

- The interactive videos were available 24x7 and could be accessed from any location with an Internet connection. Students were able to control the learning environment and the time so as to be optimal for their individual need. There was no missed instruction due to an illness, a job interview, or conflict with other activities or classes. Students replayed as often as necessary parts of a lecture that proved difficult to grasp or were missed due to an outside distraction.
- Each instructional interactive video was professionally done and of high quality. Students could control the size of the screen, the screen resolution, the volume of the video and the lighting in their room to be optimal for them. This is in sharp contrast to the less controlled and perhaps not ideal lighting, background noise, instructor's penmanship, data projector sharpness and seating location in a classroom.
- All instructional materials were well vetted. Any mistakes or unclear aspects in the interactive videos had been mentioned in the Udacity forum and corrected. In addition, the course instructor pointed out mistakes or difficult problems in upcoming assignments.
- Morale was improved during class time. Coming to class provided the students with social interaction and the ability to work together to solve exercises instead of passively listening to a lecture. The benefits of pairwise programming can be more readily achieved as the instructor can observe and comment on pairwise programming techniques. Similar to using pairwise programming in the workplace during work hours, there is a dedicated class time for students to work with their partners. This overcomes the challenge and obstacles of scheduling time outside of class [5].
- Flipped instruction is preparing students for lifelong learning. The only constant is change and the ability to continually learn and update skills is critical for short term and long term success.

Table 2: Comparative Quiz Scores for Traditional 2013 Class and Flipped 2015 Class

Class	Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quiz 6	Quiz 7	Quiz 8	Average
2013	72.44	78.66	72.57	80.57	72.71	59.71	70.64	75.02
2015	78.33	87.76	76.14	73.93	73,89	72.96	62.16	81.55

5 Faculty Advantages and Benefit of the Flipped Classroom

Changing to a flipped classroom and becoming a *Guide* was a new and rewarding experience for faculty. The majority of class time was spent working with individuals, guiding students to solutions and answering questions as opposed to lecturing. Since the interactive instructional lessons were professionally prepared, faculty had more time to prepare class exercises, grade quizzes and spend preparing supplemental materials. Faculty were able to more easily monitor and evaluate course progress with dashboards clearly showing student progress and understanding. Finally, the faculty teaching Computer Science II, downstream from the flipped classroom, knows that all students have had a similar experience.

6 Disadvantages to Students

In a well-designed and executed flipped Computer Science 1 classroom, there are minimal disadvantages to the student. The primary obstacle is making the transition from traditional lectures to a flipped approach. In a traditional class students are accustomed to taking notes and underlining key passages in the text to facilitate learning. In the flipped instruction, note taking is done differently and there is no highlighting of video material.

7 Disadvantages to Faculty

There are three possible disadvantages to faculty. First, faculty need to be *Guides* instead of lecturing experts. The changing from preparing a class and giving it as a lecture is a significantly different approach and requires some retraining or exposure to develop the new style. Second, the course offering and reliability is dependent on the vendor. Udacity could close down operation or change their pricing strategy from offering free course materials to requiring the $199 dollars for official course enrollment at any time. If this happens in the middle of the semester then the faculty member must be flexible enough to quickly transition to the traditional role. In this event, our requiring of the course text provides a nice fall back to smoothly move into this style. Finally, The use of a MOOC course with a supporting textbook is an ideal situation. In fact, the authors believe this approach will become more prevalent as current textbook authors update their new course materials to include a MOOC to remain competitive. However, for many courses there may not be a MOOC with a matching text book or a MOOC that matches the complete set of expected course outcomes or a MOOC that does not have an integrated set of student quizzes. In these cases, the MOOC is still very valuable but the instructor will need to spend a substantial amount of time to find and curate the materials and software for the course and smoothly blend them together for the optimal user experience.

8 Conclusions

Flipping the classroom by wrapping an industrial strength Udacity MOOC course vetted by students and professionals throughout the world with high quality, professionally made videos, interactive exercises, dashboards, and forums greatly improved the traditional PowerPoint, lecture/lab based Computer Science I course at Elon. Instead of the majority of the 38 semester courses sessions being filled with 30 minutes or more of lecture, the time was spent by students applying computer concepts to solve faculty guided class exercises. Most faculty do not have the time or resources to create the high quality interactive, web based educational materials created by vendors which serve a large customer base. Why not leverage these materials in a flipped classroom and become a faculty *Guide* to more efficiently and effectively use one's computer science expertise to guide and individually help students in the local classroom lab setting? Though the data provided here only compares one Computer Science 1 course taught using flipped instruction against a single traditionally formatted course, it leads us to believe that we can achieve improved student performance with the flipped format. Elon plans to move to the flipped classroom format for Computer Science 1 in the upcoming academic year.

9 References

[1] Horstmann, Cay. "Big Java : Early Objects". 5th Edition. Wiley. 2013.

[2] Brame, Cynthia. "Flipping the Classroom". Avallable: http://cft.vanderbilt.edu/guides-sub-pages/flipping-the-classroom/ .

[3] Powell, D. and Hollingsworth, J. "Flipping the Classroom by Using Cloud Services and Third Party Online Courses in Traditional Class Settings Enables All Computer Science Students and Equal Education", Proceedings of the 2014 International Conference on Frontiers in Education, Computer Science and Computer Education. 2014. .

[4] Horstmann, Cay, Lee Cheng-Han and Tansey, Sara. " Intro to Java Programming". Udacity.com 2014. Available: https://www.udacity.com/course/intro-to-java-programming--cs046.

[5] Williams, Laurie and Kessler, R., All I Really Need to Know about Pair Programming I Learned in Kindergarten, Communications of the ACM, Vol.43., pp. 108-114. May 2000.