2014 ASEE North Midwest Section Conference

Flipped Classroom

Oct 17th, 11:03 AM - 11:21 AM

Increasing Material Coverage in Software Engineering through the Introduction of the Flipped Classroom

Walter Schilling
Milwaukee School of Engineering

Schilling, Walter, "Increasing Material Coverage in Software Engineering through the Introduction of the Flipped Classroom" (2014). *2014 ASEE North Midwest Section Conference*. 2.
https://ir.uiowa.edu/aseenmw2014/flipped_classroom/2B/2 https://doi.org/10.17077/aseenmw2014.1020

Proceedings,
The 2014 ASEE North Midwest Section Conference,
October 16-17, 2014, Iowa City, IA.

ASEE-NWMSC2014-2B2

Increasing Material Coverage in Software Engineering through the Introduction of the Flipped Classroom

Dr. Walter Schilling
Department of Electrical Engineering and Computer Science
Milwaukee School of Engineering

Abstract

Software Engineering represents a rapidly changing engineering discipline. As a young discipline, the field has not reached the same level of maturity as other engineering disciplines. Furthermore, as a rapidly evolving field, it also is encountering greater change than many other disciplines of engineering. This change leads to a much greater challenge meeting the needs of diverse engineering constituents. More material must be taught in each course and at a faster pace in order to ensure that students are ready for the demands of industry.

At the Milwaukee School of Engineering, curriculum changes have resulted in a reduction in lab content and credit for courses. In one course, Operating Systems, the lab component has been removed entirely. However, through prudent course design and the usage of the flipped classroom, the same amount of content was able to be covered in less time.

This article will present an analysis of the findings of applying the flipped classroom to teaching operating systems to software engineering students. Included will be analysis of student performance from control groups prior to the curriculum conversion, as well as observations from students on the usage of the flipped classroom. In particular, the data indicates improved performance for the bottom group of students when using the flipped classroom approach.

Introduction

Software Engineering is a rapidly evolving field. Each year, there are new innovations and new technologies which must be taught to students. However, this can be tremendously challenging, for degree programs are credit-limited and must be designed with for students to graduate in a timely fashion. In 2010, the Software Engineering Program at the Milwaukee School of Engineering (MSOE) underwent a major curriculum transformation, adding new required courses in Computer Networking, Web Applications Development, Real Time Systems, and Programming Languages. These additional courses were added based on industrial advisory board and recent alumni feedback. However, in order to make space for 15 new quarter hours, compromises needed to be made in other areas of the curriculum. One such area receiving modification was the coverage of operating systems. This article describes the changes made to the course as well as results from the first offering of the course in a flipped classroom format.

Institutional Profile

MSOE offers an accredited Bachelors of Science degree in software engineering, and has been accredited since 2002. As an institution, there is a strong emphasis on small class sizes (14:1 student to faculty ratio) and extensive laboratory experience. Students graduating from MSOE spend on average 600 hours in laboratories related to their major. Institutionally, there is more square footage devoted to lab space than lecture hall space. All engineering students are required to complete a three course capstone experience. While the majority of students on campus are in the engineering fields, the school also offers a nursing program, a technical communication program, and several business programs.

MSOE prides itself in having very few traditional computer labs on campus. Instead, all students enrolled in the university are issued a laptop as part of a technology package which includes the laptop and all relevant software needed for the program the student is enrolled in.

The software engineering program offers several unique learning opportunities. One part of the program is a 12 credit Software Development Laboratory experience where students work on large-scale, industry-sponsored projects. Students are also required to take an application domain sequence of three related, specialized courses which emphasize the application of software engineering material to different domains. Most software engineering courses are offered in the 3+2 format, meaning the course meets in lecture three times for one hour and have a 2 hour associated lab period.

The Baseline Operating Systems Course

In the old curriculum, the Design of Operating Systems Course, CS3841, was shared by the Software Engineering Program and the Computer Engineering Program. The course consisted of 3 hours of lecture per week and a 2 hour lab. Lab activities consisted of labs designed to demonstrate the usage and core designs for an operating system. All labs used a Linux Virtual machine and the C programming language. Outcomes for CS3841 are shown in Figure 1.

Upon successful completion of this course, the student will be able to:

1. Identify the components of operating system process management.
2. Recognize issues related to concurrent processes and synchronization techniques.
3. Discuss and illustrate several approaches to operating system memory management.
4. Discuss and illustrate several scheduling algorithms.
5. Describe input/output handling in operating systems.
6. Illustrate file system interfaces and implementation.
7. Construct software applications which use POSIX/UNIX system calls to spawn additional processes.
8. Construct software which uses multiple POSIX/UNIX threads.
9. be able to perform independent research on a focused technical topic.
10. be able to document research results in a technical paper.
11. Construct software which implements a fundamental data structure in C and manages memory using malloc and free.
12. Construct a software library which manages a memory heap.

Figure 1: CS3841 Design of Operating Systems Course Outcomes

The Design of Operating Systems course earned a reputation as being a tough course from students. In this course, they spent a significant amount of time working on labs, as several of the labs were quite challenging from a technical standpoint, and many students did not fully grasp the C programming language. Additionally, students were required to write a 20 page research paper on computer engineering or software engineering related topics.

The New Operating Systems Course

In developing a new curriculum, 16 quarter hours of classes were added to the program, including 4 new courses with labs. In order maintain the number of credits in the curriculum, compromises needed to be made, and as such, a decision was made to remove the associated lab from the operating systems course, replacing it with a lecture only course. This resulted in a 40% reduction in contact time with the students.

To make the adjustment, the overall outcomes of the course were reduced slightly, resulting in the course outcomes provided in Figure 2. The most significant outcomes removed from the course dealt with student research and the ability to perform independent research, as this topic was moved to another course in the curriculum. Several other outcomes were modified to a lower level of Bloom's Taxonomy. However, this resulted in a challenge, as overall, the technical content of the course had not been significantly reduced nor could be reduced if students were to be prepared for the new Real Time Systems Course which followed.

Upon successful completion of this course, the student will be able to:

1. Identify the components of operating system process management.
2. Recognize issues related to concurrent processes and synchronization techniques.
3. Discuss and illustrate several approaches to operating system memory management.
4. Analyze the usage of memory management systems experimentally.
5. Discuss and illustrate commonly used scheduling algorithms.
6. Describe input/output handling in operating systems.
7. Illustrate file system interfaces and implementation.
8. Apply POSIX system calls.

Figure 2: CS3844 Operating Systems Course Outcomes

In an attempt to make the course more efficient, the course was developed from the start to use a "flipped learning" method. The flipped learning classroom, extensively described in literature [1], [2], allows an instructor to increase the amount of time devoted to active learning exercises while not reducing content coverage, and offers many benefits from an academic standpoint.

Flipping this class involved the development of 20 videos which covered the bulk of the lecture material, as is shown in Figure 3. Videos ranged in length between 6:40 and 18:44, with the average length being 9:23. 25 in class active learning exercises were also developed, ranging from implementing a simple context switcher on a 32 bit microcontroller to developing a simple user shell for Linux. Each video contained an embedded quiz, typically 5 questions, which assessed student understanding of the lecture material.

Topic	Script writing time	Video Recording	Video Editing	Quiz Writing	Overall Video length
Scheduling Part 1	2:32:34	1:16:16	0:54:53	0:13:10	0:06:50
FCFS Scheduling	1:02:05	1:04:22	0:01:25	0:16:04	0:06:40
SJF Scheduling	1:22:41	0:42:44	0:35:33	0:31:37	0:10:13
Processes	0:42:01	0:55:11	0:33:48	0:32:23	0:18:03
Process Operations	1:41:17	0:32	0:46:15	0:20:00	0:11:03
Interprocess Communciations	2:04:03	0:26:53	1:03:14	0:31:14	0:09:17
System Calls	6:50:19	1:52:11	2:10:52	0:12:44	0:18:44
OS Structures	2:56:55	0:45:04	1:34:56	0:09:30	0:10:07
Process Synchronization	1:52:47	1:38:21	1:36:06	0:16	0:13:40
Process Operations (Context Switching)	0:45:00	0:40:00	0:40:00	0:12:00	0:14:02
Sockets	1:52:27	1:28:36	1:16:23	0:41:00	0:12:10
Threads	1:49:14	0:29:11	1:00:10	0:31:15	0:07:24
Thread cancellation and system calls	1:01:00	0:30:00	0:26:00	0:28:38	0:09:42
Priority Scheduling					0:11:26
Virtual Memory					0:14:56
Deadlocks	0:21:00	0:53:00	0:26:00	0:29:00	0:14:54
File Systems	3:16:11	0:40:11	1:48:19	0:14:54	0:16:41
Memory					0:17:44
					2:54:17

Figure 3: Video Lecture data.

Assessment of the First Offering

In order to assess the sentiment of students enrolled in the course, a brief survey was provided at the end of the course for students to provide feedback on the course, shown in Figure 4. Overall, students viewed the course positively. In general, students felt the course had approximately the same difficulty as their other software engineering courses, and they strongly agreed that the videos provided better preparation than a comparable textbook for the course materials. Overall, students felt that the videos were about the right length and overall they would enroll in another flipped classroom if given the opportunity.

The survey did reveal some surprising results. First and foremost, the students indicated that they spent about the same amount of time on this class as they did on their other classes. This was quite surprising, as their other courses all involved extensive labs and were 4 credit courses, resulting in at least an additional 2 contact hours per week as well as outside of lecture work. This sentiment either indicates the students are not spending as much time in their other courses as we feel they are or the survey was misinterpreted. Further analysis of this area will occur in future offerings.

It was also surprising to see the strong support for embedded quizzes in the videos. In the past, other faculty members have attempted to use videos for class preparation in lieu of a textbook. However, any feedback to the instructor has come from a blackboard or paper based quiz. With these videos, quizzes were embedded directly into the videos using Camtasia quiz feature [3]. This prevented students from watching parts of the video without completing the quizzes as well as providing feedback to the instructor on how long students spent watching videos. It also prevented students from taking the quiz without watching the video (though it was possible for students to fast forward through parts of the video, and many students did this). All in all, this seemed to improve the cognitive understanding of the material.

To assess student learning, the results from the midterm exam were compared between flipped and non-flipped courses. This result is shown in Figure 5. Overall, the top performing students generally matched in the flipped classroom offering with those in the most recent 3 traditional courses. However, as the students' grade percentiles decreased, the flipped scores were higher, indicating greater understanding of the material. This trend held constant across all subsections of the exam as well. Students performed better on the multiple-choice, free response, and problem-based sections of the exam in the flipped classroom offering versus the traditional offerings.

Question	Responses	Average	Median	Stdev
Overall, this course was ________ my other SE courses.	(5) Significantly harder than (4) harder than (3) about the same as (2) easier (1) significantly easier	2.88	3	0.59
Overall, I spent ________ my other SE courses.	(5) Significantly more time than (4) more time than (3) about the same time as (2) less time than (1) significantly less time than	3.04	3	0.82
I felt that the usage of videos and online material in advance of class helped to prepare me for lecture better than traditional textbook readings.	(5) Strongly Agree (4) Agree (3) Ambivalent (2) Disagree (1) Strongly Disagree	4.42	4.5	0.64
I prefer the "flipped classroom" approach to a traditional classroom approach.	(5) Strongly Agree (4) Agree (3) Ambivalent (2) Disagree (1) Strongly Disagree	3.62	4	0.94
Overall, I felt that the videos lengths were.	(5) Way too long (4) too long (3) about right (2) too short (1) way too short	3.15	3	0.46
If given the opportunity, I would enroll in another class taught using the "flipped classroom" approach.	(5) Strongly Agree (4) Agree (3) Ambivalent (2) Disagree (1) Strongly Disagree	3.76	4	0.97
I felt that the quizzes in the video forced me to pay attention and watch the videos.	(5) Strongly Agree (4) Agree (3) Ambivalent (2) Disagree (1) Strongly Disagree	4.24	4	0.93

Figure 4: CS3844 Flipped Classroom student feedback.

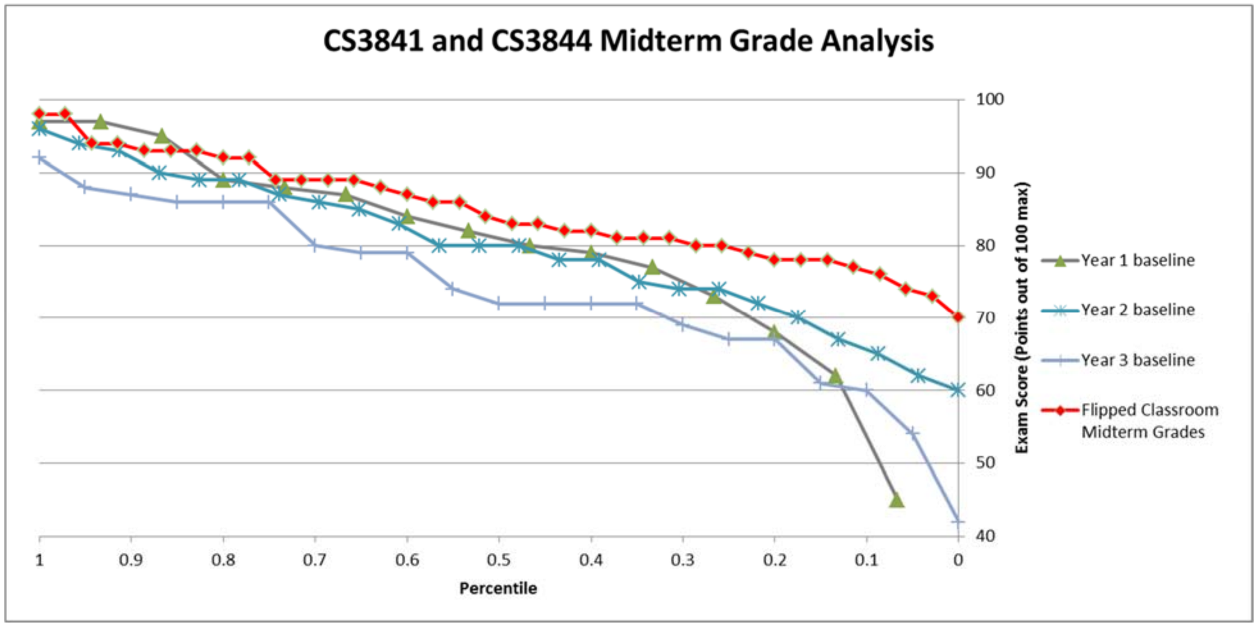

Figure 5: A Comparison of Midterm Exam grades between the Flipped Classroom and the traditional classroom.

Summary and Conclusions

This paper has shown preliminary results of incorporating a flipped classroom approach to teaching a technical topic in the software engineering program. The flipped classroom approach was necessitated by a need for greater teaching efficiency as part of a curriculum change and a 40% reduction in content time for a given course. All in all, the approach was supported by students in the class and resulted in an increase in student's performance as measured by examination.

Bibliography

1. J. Bergmann, Flip Your Classroom: Reach Every Student in Every Class Every Day, International Society For Technology In Education (ISTE), 2012.
2. J. Bishop and M. Verleger, "The Flipped Classroom: A Survey of the research," in ASEE Annual Conference, Atlanta, 2013.
3. "Camtasia - Advanced Features: Adding a Quiz to a Lecture," 3 7 2013. [Online]. Available: http://webs.purduecal.edu/oit/files/2013/09/Camtasia-Lecture-Recording-with-Quizzes-revised-9-5-13.pdf. [Accessed 1 6 2014].