Developing and Teaching a Hybrid Software Engineering Introductory Course

Anna Koufakou[1]
Florida Gulf Coast University

Abstract

This paper summarizes the author's experiences in developing and teaching for the first time a hybrid (or blended) undergraduate Software Engineering (SE) fundamentals course for junior SE students at Florida Gulf Coast University (FGCU). This course traditionally includes lectures of SE concepts and techniques, while students are required to apply the material from the lectures to a software group project working mostly off-classroom. The hybrid course model replaces some of the face-to-face classroom meeting time with online learning activities. Successful hybrid courses have been shown to promote student learning, increase flexibility for students and for faculty, as well as improve classroom space utilization for institutions. The hybrid implementation of this course received a positive student response, further supported by student performance data. Experiences and lessons learned are also discussed in the paper.

Keywords

hybrid/blended course; software engineering

Introduction

Blended or hybrid learning and course delivery has increased in the last few years in higher education. Scholars have predicted that hybrid teaching will become the "new normal" in higher education course delivery[1]. The hybrid course model replaces some of the face-to-face in-classroom meeting time with online learning activities. In this way, hybrid courses can combine the best of both worlds, traditional and online learning[2]. Successful hybrid courses can promote student learning, increase flexibility for students and for faculty, as well as improve classroom space utilization for institutions[2, 3]. There is great variation in what is considered a hybrid course and how it is assessed or supported by various institutions[4]. The literature also reports challenges and obstacles faced by the faculty developing hybrid courses, for example: faculty workload, complexity of the hybrid course instruction, and lack of institutional support, among others[5, 6].

This paper presents the author's experiences in designing and teaching for the first time a hybrid Software Engineering introductory course. Traditionally, this course consists of in-class lectures, along with a semester-long project that requires students to develop software in teams of 3-5 students. This project is time consuming as it includes coding in a much larger scale than what most students have experienced so far, plus documentation, presentations, and maintaining a website. In the traditional course delivery, lectures utilize a large fraction of the class meeting

[1] U. A. Whitaker College of Engineering, Florida Gulf Coast University, 10501 FGCU Blvd S, Fort Myers, Florida 33965, akoufakou@fgcu.edu

© American Society for Engineering Education, 2015

time, while students report that they spent a lot of additional time off-class to work on their project. The hybrid course model was chosen to increase flexibility and convenience, especially for students to work on their project, while also promoting student learning.

The following sections describe the SE course, CEN 3031 - Software Engineering Fundamentals, followed by the design of the hybrid course. The assessment of the hybrid course delivery versus the traditional is presented next. Finally, the paper discusses challenges and opportunities, and provides concluding remarks.

Course Description

Course Outline: CEN 3031 - Software Engineering Fundamentals is a 3-credit hour undergraduate course without a laboratory component. It is a required course for all SE students at Florida Gulf Coast University (FGCU). The course is the first in a series of Software Engineering (SE) courses offered in the new SE degree program at FGCU established in 2011. The sequence of SE courses is: Software Engineering Fundamentals (the course in this paper), Software Specifications, Software Architecture & Design, and Software Testing (each a 3-credit hour course). There is no graduate SE degree offered by the school so there is no graduate teaching assistantship available.

Students arrive at this course with intermediate knowledge of programming. This course aims to introduce SE concepts to students, including the importance of requirements engineering, design, and maintenance. Lessons include terms and tools that are new to the students, such as software development life-cycle models, software cost estimation models, Unified Modeling Language (UML), etc. The course material primarily focuses on software process, requirements, design, testing, and maintenance. A list of topics and the corresponding timeline is shown in Table 1.

Table 1. List of Course Topics (Fall 2012 and Fall 2013)

#	Topic	Timeline
1	Overview of Software Engineering	Week 1
2	Software Process	Week 2
3	Software Development Life Cycle (SDLC) models	Week 2-3
4	Object Oriented Concepts, Unified Modeling Language (UML)	Week 3-4
5	Software Requirements	Week 5-6
6	Project Planning and Estimating	Week 5-6
7	Software Testing	Week 7
8	Object Oriented Analysis and Design	Week 9-10
9	Implementation: Integration and Testing	Week 12-13
10	Post-delivery Maintenance	Week 14

Team Project: An important part of the course is that it uses a semester-long software group project where students apply the material they learned. Students are required to design, develop, and test a software application in teams with 3-5 students per team. The team project has several

© American Society for Engineering Education, 2015

deliverables which are shown in Table 2. Student teams receive fewer points for completing early deliverables such as forming a team and submitting their proposal (5%), and more points for advance deliverables such as detailed design (15%). A large part of the project grade (65%) comes from the final deliverables and presentation at the end of the semester.

The traditional course (Fall 2012) includes three presentations: Presentation 1 is for the teams to introduce the class to their project topic, specifications, and plan; Presentation 2 is on design; and Presentation 3 (final) is to demo their product and summarize their results. The hybrid course (Fall 2013) has Presentations 1 and 2 combined into one presentation due to hybrid class meetings outlined in the next section of this paper.

Table 2. Project Deliverables and timeline (Fall 2012 and Fall 2013)

Deliverable	**Timeline**
Project Proposal	Week 3
Project Management Plan, Software Requirements Specifications (SRS)	Week 6
Detailed Design	Week 12
Source Code, User Manual, Web page	Before final presentation
Presentations	*Fall 12:* Weeks 6, 12, and 16 *Fall 13:* Weeks 12 and 17

Student Assessment: In addition to the team project described earlier, both courses include two exams, and two individual assignments. Assignments are on UML diagrams and programming with Java. The hybrid course (Fall 2013) also includes quizzes that count towards 5% of the total grade. The role of the quizzes is to assess student completing the assigned reading and exercises before the face-to-face meeting in class.

Hybrid Course

Motivation. As mentioned earlier, the main motivation for redesigning the course was to increase student engagement, promote active learning, and improve student performance. After having taught this course the traditional way a few semesters, the instructor observations supported by student feedback in end-of-semester evaluation were that students found much of the lecture subject matter boring and tedious. On the other hand, the students regularly observed that they learned quite a lot in their team project; however, they protested that they had to do much of the work outside the class as the class time was mostly used for lectures and exams. Another observation made by the instructor was that as the traditional course did not include quizzes, several students tended to cram right before exams rather than study and participate throughout the semester.

One way to address these issues is to move some of the terminology and what could be termed ‘basic’ or ‘straight-forward’ concepts to assigned readings and activities that the students would complete before class. Additionally, a part of the class time would be spent on hands-on learning, as well as project activities, for example the team could spend time in class to work on their SRS

© American Society for Engineering Education, 2015

or discuss their design. This way, students would engage in active learning during class time. This way of teaching could be described as an implementation of the *flipped classroom* model.

According to the literature [2,3,9] as well as discussions with other faculty, a blended or hybrid course would take advantage of technology to remove the space and time class-meeting constraint and thus offer more flexibility: in such a course, "students can get on with their everyday life, without having to adapt systematically to a specific space and time, as they are obliged to do in face-to-face […] All this motivates the students' interest in the subject, which encourages learning and leads to better outcomes" [3]. In the case of our course, for example, the project teams could collaborate via online meeting mechanisms and work on shared online documents or code.

In summary, a hybrid course was selected for CEN 3031 as it still includes face-to-face time for lectures on complex concepts, hands-on activities, and guided project time, while it allows students to organize their project meetings and individual studying with more flexibility.

Hybrid Course Implementation. The traditional course offering in Fall 2012 included two meetings of 1.5 hour each per week (total of 3 hours). The Fall 2013 hybrid course described in this paper met once a week for 1.5 hour. The off-classroom portion of the class included assigned readings and exercise activities to be completed before class. Face-to-face meetings incorporated discussion and hands-on application of the new material, as well as project-related activities.

Specifically, a list of assigned readings and activities was posted on the course Learning Management System LMS (Canvas) the week before the class meeting. This list also included a list of objectives and items on which the students needed to focus. The readings were assessed by a quiz, mostly with multiple choice questions. The goal of the related questions on the quiz was not to have students understand material on their own, but rather to familiarize students with the basic concepts, terminology, and formulas before working on an actual estimation problem in the classroom.

An example follows for the "Software Planning and Estimating" module. Questions on the quiz were related to terms or formulas, for example on the Constructive Cost Model (COCOMO), an algorithmic software cost estimation model [7]. An example question is given in Figure 1. Again the goal of this question is for the students to be familiar with COCOMO terminology, terms, and formulas before the face-to-face meeting.

In Intermediate COCOMO, applying the following formula:

$$\text{Nominal effort} = 2.8 \times (\text{KDSI})^{1.2} \text{ person-months}$$

means that the software product development mode was estimated to be:

a) Organic or straightforward
b) Semidetached or medium
c) Embedded or complex
d) Real-time

Figure 1. Example question on quiz related to "Software Planning and Estimating" in the hybrid course (Fall 2013)

© American Society for Engineering Education, 2015

The weeks that included an exam, project deliverable, or homework assignment did not include a quiz. During the class meeting, a brief lecture overview of the assigned readings and the quiz was given, including questions raised by the students. This was followed by giving a hands-on activity, completed by students usually in groups. After the activity, the results of the group work were discussed with the class, and solutions were demonstrated and explained.

Regarding the example quiz described earlier (see Figure 1), during the subsequent class meeting, after a quick overview lecture of "Software Planning and Estimating" including COCOMO, students were given an actual problem with numerical data for which they were instructed to use COCOMO in order to estimate software development effort. After the activity, solutions were presented along with a discussion on questions as well as critique of the estimation model. During the remainder of the class meeting, the students worked on their own team project plan and schedule with the assistance of the instructor.

Assessment

The hybrid course assessment versus the traditional course is based on student performance as well as student feedback in end-of-semester evaluations.

Student Performance

Software Engineering at FGCU has set achievement standards in junior level courses on a program level to target 40% of students in a course achieving at the 85th percentile or above, 70% of students achieving at the 70^{th} percentile or above, and 80% of students achieving at the 65^{th} percentile or above. Table 3 displays the performance achieved by the students in both traditional and hybrid course for each target level for specific course objectives. For example, given the first Course Objective and Exam Questions in Table 3, 91% of students in the traditional course (Fall 2012) achieved at the 70^{th} percentile or above versus 94% of students in the hybrid course (Fall 2013). Both courses surpassed the target set by the program, which is "70% of students achieving at the 70^{th} percentile or above".

As shown in Table 3, there was improvement for almost all of the items assessed for the different course objectives for the hybrid course versus the traditional course delivery. At the end of the semester, the project final deliverables were overall of similar quality and students achieved all target levels for both courses, even though there were more students and project teams in the hybrid course: specifically, there were 23 students and 6 project teams in 2012 (traditional) versus 34 students and 8 project teams in 2013 (hybrid). Additionally, the target levels for the new component introduced in the hybrid course, the quizzes, were also achieved. As a point of reference, the *overall* average for the courses were 90% for the traditional course versus 89% for the hybrid course.

© American Society for Engineering Education, 2015

Table 3. Target Performance achieved by students on course objectives for traditional (2012) and hybrid (2013) courses -*bold numbers indicate equal or larger result than traditional course*.

Course Objective	Item	Target Performance Level	Fall 2012 Results (n=23)	Fall 2013 Results (n=34)
Learn software engineering principles, concepts, methods, and techniques	Exam Questions	40% of students at 85th percentile or above	74%	62%
		70% of students at 70th percentile or above	91%	**94%**
		80% of students at 65th percentile or above	96%	**97%**
	Quizzes	40% of students at 85th percentile or above	N/A	**68%**
		70% of students at 70th percentile or above	N/A	**76%**
		80% of students at 65th percentile or above	N/A	**82%**
Construct UML diagrams	UML Assignment	40% of students at 85th percentile or above	65%	**88%**
		70% of students at 70th percentile or above	96%	**100%**
		80% of students at 65th percentile or above	100%	**100%**
Work in a group to perform activities & provide documentation related to all phases of software development	Detailed Design	40% of students at 85th percentile or above	83%	**100%**
		70% of students at 70th percentile or above	100%	**100%**
		80% of students at 65th percentile or above	100%	**100%**

Student Evaluations

Students evaluate course instruction by completing an FGCU survey at the end of semester. This survey includes both Likert scale questions as well as open-ended questions. Specific statements from the survey were selected to gauge the student response to the hybrid course, e.g. improved learning or simulation of interest.

Table 4 shows the selected statements from the FGCU survey, along with a high-end rating/response for each statement, and the percentage of students from each course that gave that rating for the statement. The scale on each of these statements ranged either from *Strongly Disagree* to *Strongly Agree*, or from *Poor* to *Excellent*. As shown in Table 4, the responses from the hybrid course were improved compared to the traditional course. For example, 93% of students who took the survey in the hybrid course agreed or strongly agreed that they learned a great deal about the subject versus 78% in the traditional course.

© American Society for Engineering Education, 2015

Table 4. Percentage of students that gave a high rating on selected statements in the FGCU student survey for traditional (Fall 12) and hybrid (Fall 13) courses.

Survey Statement	Rating	Fall 12 (n=18) Responses	Fall 13 (n=29) Responses
"I learned a great deal about the subject"	Agree/Strongly Agree	78%	93%
"I was always prepared for class"	Agree/Strongly Agree	66%	86%
"Facilitation of Learning"	Very good/Excellent	83%	90%
"Stimulation of interest in the course"	Very good/Excellent	72%	76%

Finally, the comments below show a variety of student responses from the open-ended part of the end-of-semester student survey in Fall 2013 (hybrid course):

- "I feel the days the class met was just the right amount of time for class material. If the class met two days a week, I feel there would be too much time for class material and not enough for project material. If the future classes do meet twice a week, perhaps one of the days should be used for in class project help/work."
- "I felt we were given enough time for the project, but it was extremely difficult to coordinate a project of this size with the amount of work given by other classes that I have to spend time on. Having more milestones set up might help students prioritize their time more to the project."
- "I found that the quizzes and tests were well prepared, and that the time frame for the tests were well done."
- "Meeting once a week did not work out as expected. People do not show up on Thursday unless they're forced to."
- "The class experience was very good. The time given to us by just having only one meeting a week was used mostly for other classes but we did use it for this class also especially during the end of the project since we had to finish it before the final presentation."

Discussion

Overall, the hybrid course was received well by students and was shown to help increase student learning and performance as presented in the previous section. The instructor also observed increased attendance by students, which could be attributed to having face-to-face meetings only once per week. Another reason might be that having assigned readings and activities plus quizzes before class meetings led to students feeling more motivation to be on top of their studying and participate in class every week. An increase in student engagement and participation in class was indeed witnessed by the instructor for a large part of the class body.

The instructor also observed, as stated by Kaleta et al.[2], that in this course, students were challenged to take more control of their learning. It certainly seemed that there might have been students who thought that a class that meets once a week will be easier than a traditional class, or students who felt that they should/could use the "free time" to study for other courses (see student survey comments in previous section). The instructor does need to frequently remind

© American Society for Engineering Education, 2015

students of online deadlines as well as of the importance of their online tasks. On average, approximately 10% of the students missed a quiz (3.5 students out of 34) which might be attributed to the fact that quizzes were only 5% of semester grade. Additionally, it could be because, as reported in literature[2], students feel that the in-classroom component is the "real" class and they find it hard to keep track of additional deadlines online. In general, as also stated by Nel and Wilkinson[8], the instructor must emphasize to students the consequences of lack of planning and participation in their blended course in order for the students to know what to expect.

As was anticipated, the instructor must do a lot of work to redesign the course for some components to be online as well as to prepare online material and quizzes. For example, some concepts might need more face-to-face time than others, or splitting material into an online component and an in-class component might not make sense. In order for this to work, online components need to make sense to students, online material should not be duplicated in in-class lectures, and the hybrid class should not result in a lot of additional work for students compared to the traditional course [9].

Scheduling a course to meet once a week also has some complications. For example, class presentations did not fit in one class session as there were 8 teams to present with an expected time of 10-15 minutes. The instructor opted to schedule the class to meet twice in Week 12, which was not possible with a couple of students' schedules. Additionally, some groups might face problems with disappearing team members, which could be due to reduced face-to-face interaction and smaller perceived accountability by students.

Finally, ideas are presented to overcome these limitations. First, an instructor might choose to develop a hybrid course with a slightly smaller off-classroom component, e.g. 35% instead of 50%. This way, more project activities could be scheduled in class with the guidance of the instructor and senior SE student assistants if possible. Another avenue is for the instructor to incorporate online (e.g. chat) meetings with the groups. Also, quizzes and in-class activities could count more towards the student grade to give students a higher incentive to complete the quizzes and to participate in class.

Conclusions

This paper presented the author's experiences in designing and teaching a hybrid Software Engineering introductory course for the Software Engineering program at FGCU for the first time. Overall, the hybrid model was beneficial: the course was received well by students, increased flexibility for students, and was shown to improve student engagement and promote learning. Challenges included increased workload and complexity for the instructor as well as reduced face-to-face interaction with student teams.

References

1. Norberg, A., Dziuban, C. D., and Moskal, P. D. "A time-based blended learning model", *On the Horizon*, 19(3), 2011, pp. 207-216.
2. Kaleta, R., Skibba, K., and Joosten, T. "Discovering, designing and delivering hybrid courses". In A. G. Picciano & C. D. Dziuban (Eds.), *Blended learning: Research perspectives, Needam, MA: The Sloan Consortium*, 2007, pp. 111-143.

© American Society for Engineering Education, 2015

3. Alonso, F., Manrique, D., Martinez, L., and Vines, J.M., "How Blended Learning Reduces Underachievement in Higher Education: An Experience in Teaching Computer Sciences," *IEEE Transactions on Education*, 54(3), 2011, pp.471-478.
4. Graham, C., Woodfield, W., and Harrison, B., "A framework for institutional adoption and implementation of blended learning in higher education," *The Internet and Higher Education*, 18, 2013, pp. 4-14.
5. Ocak, M., "Why are faculty members not teaching blended courses? Insights from faculty members," *Computers & Education*, 56(3), 2011, pp. 689-699.
6. Oh, E., and Park, S., "How are universities involved in blended instruction?", *Educational Technology & Society*, 12(3), 2009, pp. 327–342.
7. Boehm, B., "Software Engineering Economics", *Englewood Cliffs, NJ Prentice-Hall*, 1981.
8. Nel, L. and Wilkinson, A. "Enhancing collaborative learning in a blended learning environment: applying a process planning model," *Systemic practice in action research*, 19, 2006, pp. 553-576.
9. Hoic-Bozic, N., Mornar, V., and Boticki, I., "A blended learning approach to course design and implementation," *IEEE Transactions on Education*, 52, 2009, pp. 19-30.

Anna Koufakou

Dr. Koufakou is an Assistant Professor in Software Engineering in the U.A. Whitaker College of Engineering at Florida Gulf Coast University. Dr. Koufakou received a B.Sc. in Computer Informatics at the Athens University of Economics and Business in Athens, Greece, and a M.Sc. and a Ph.D. in Computer Engineering at the University of Central Florida. Her research interests include mining of large datasets, outlier detection, and frequent itemset mining. Educational areas of interest are promoting student engagement via techniques such as hybrid teaching, flipped classroom and problem-based learning.

© American Society for Engineering Education, 2015