Experiences translating project-based Software Engineering courses into online courses

Srividya K. Bansal[1], AjayBansal[2]

[1,2]School of Computing,Informatics,and Decision SystemsEngineering, Arizona State University, Mesa, TX, USA
[1]srividya.bansal@asu.edu
[2]ajay.bansal@asu.edu

Abstract: Online education has seen an enormous growth in the last few years. In this paper we describe the design of three project-centric Software Engineering courses in an undergraduate software engineering program to be delivered online. The study examined three courses offered in face-to-face and online environments. The course goals, structure, learning objectives, and assessments were exactly the same. The courses were designed with hands-on and project-based activities in a cooperative learning environment using a Software enterprise pedagogical model. The paper presents a comparison of student performance on various course outcomes, working in teams, success/failure rates and lessons learned from translating a face-to-face course into an online course. These results can be useful to other educators and institutions in how to improve student learning outcomes and learner satisfaction in online environments and further improve quality of online course offerings.

Keywords: Online education; learn-by-doing; project-based learning; software engineering education.

Srividya K. Bansal[1],
[1]SCIDSE, Arizona State University
[1]srividya.bansal@asu.edu

1. Introduction

Software Engineering (SE) profession is a fast-growing profession in the nation's economy thereby adding pressure on SE educators to produce industry-ready graduates (Phase & others, 2005). There has been a surge in enrollment in undergraduate computer science and software engineering programs. In addition, online education has seen an enormous growth in the last few years that bring new challenges in providing high-quality online education. SE education is rapidly evolving and technologically challenging discipline. Research shows that learners are more engaged in hands-on and active learning environments as opposed to traditional lecture-oriented classes. Sheppard et al. suggest that engineering curricular design should move away from a linear, deductive model and move instead toward a networked model: "The ideal learning trajectory is a spiral, with all components revisited at increasing levels of sophistication and interconnection" ((Sheppard et al., 2008) p. 191.

A project-centered undergraduate B.S. in Software Engineering degree program was implemented in our institution in 2009.The principal design feature of the program is a realization of a *professional spine*(Sheppard et al., 2008) through a culture of project-based learning (PBL) (Gary, Lindquist, Bansal, & Ghazarian, 2013). In 2013, our administration asked that the B.S. in Software Engineering be offered online. The translation of the face-to-faceprogram to online is happening one major year at a time, so as of Fall 2015 the junior year is just being rolled out. In 2 short years the B.S. in Software Engineering has grown to an enrollment of 450 students, the second largest program within our institution. Translating our current program for full online delivery poses certain challenges in the development and delivery of these courses. This paper describes the translation of three core Software Engineering courses, the delivery of these courses for the first time as asynchronous online courses, and the results obtained.

Our Software Engineering courses were designed with hands-on and project-based activities in a collaborative learning environment using a Software enterprise pedagogical model. Studies show that online learners are getting frustrated more and more with collaborative learning (Capdeferro & Romero, 2012). The study shows that the factors leading to the frustration are time pressure, time zone differences, communication issues, delayed

feedback, and commitment imbalance. Another study lists factors that drive successful e-learning as learner computer anxiety, instructor attitude toward e-learning, e-learning course flexibility, e-learning course quality, perceived usefulness, perceived ease of use, and diversity in assessments being the critical factors affecting learners' perceived satisfaction (Sun, Tsai, Finger, Chen, & Yeh, 2008). The goal of this paper is to advance the community's understanding of the process of translation of a traditional face-to-face course into an asynchronous online course, how teaching and learning innovations may be delivered in the context of project-centered curricular structures and demonstrate their adaptability. The above listed factors for successful e-learning will be taken into consideration in course design.

A. Research Objectives

The specific research questions of this study are as follows:

(i) how does the translation of traditional face-to-face software engineering course into online course affect student performance?
(ii) how effective is the online software engineering course in a project-centered curricular context?
(iii) what are the challenges in delivering a hands-on software engineering course online?

B. Paper Outline

The remainder of the paper is organized as follows: related work and background is presented in Section 2. Section 3 presents the course description and design of three Software engineering courses used in this study. Section 4 presents the translation of course design for online delivery. Results are presented and discussed in section 5 followed.

2. Related Work and Background

Online courses and degree program offerings are becoming widespread and controversial. Faculty opinions and research exist both for and against the online model. There are varying interpretations of online as well, from "blended" models to new "massively open online courses" (MOOCs). In this study we consider online delivery to be a course whose experience is delivered entirely online with zero in-person interaction (instructor-to-student and student-to-student), but it may allow for synchronous or asynchronous activities and communication. For Software Engineering courses, the challenge is facilitating team-oriented practice, project activities and ensuring students stay in synch in what is a time-sensitive delivery model.

Cognitive Apprenticeship Model (CAM) presents a theory that assumes people learn from one another, through observation, imitation, and modeling(Sun et al., 2008). CAM describes four dimensions that constitute a learning environment: content, method, sequencing, and sociology. Our course design takes these four dimensions into consideration. Burge presents her reflection of MOOC's experience and describes how taking several MOOCs has given her a better understanding of student motivation, time commitment issues, and student perception of grading. She describes how she is using those insights in her regular classroom teaching and describes how this knowledge can be used back into our own courses (Burge, 2015). Studies have been conducted on instructional activities that have online discussions on social networking sites and concluded that the characteristics of learners and their individual differences have to be taken into consideration during the design of an instructional activity (Lin, Hou, Wang, & Chang, 2013). An investigation on online collaborative learning experiences identified important factors that were crucial for building teamwork trust (Tseng & Yeh, 2013). Relationship conflicts, lack of communication, and low levels of individual accountability seemed to be the top factors for virtual teams failing to work collaboratively. We have attempted to address these factors in our course design.

A. Background

The Software Enterprise is a blended pedagogical model that combines traditional lecture with newer, forward-thinking techniques in problem-based and project-based learning (Cary, 2008). This model leads students through a modular series of lessons that combine foundational concepts with skills-based competencies. The Software Enterprise has been in use at the author's institution since 2004, and now forms the "project spine" that runs the full four years of B.S. in Software Engineering degree program(Gary et al., 2013). The primary goal of the Enterprise is to move students rapidly from foundational concepts to industry best practices, so students completing the degree program are prepared to meaningfully contribute to the profession without any additional training. The Enterprise pedagogy takes studentsfrom introduction of a concept to scalable practice in a real project in the span of a two-to-three week "sprint." This is in contrast to typical degree programs that introduce a concept with toy problems in one course and then expect students to synthesize the multiple concepts in a capstone project (culminating experience) in a different course (perhaps years later).

B. Software Enterprise Delivery Model

The Software Enterprise is an innovative pedagogical model for accelerating a student's competencies from understanding to comprehension to applied knowledge by co-locating *preparation*, *discussion*, *practice*, *reflection*, and *contextualized learning* activities in time (Cary, 2008). In this model, learners prepare for a module by doing readings, tutorials, or research before a class meeting time. The class discusses the module's concepts, in a lecture or seminar-style setting. The students then practice with a tool or technique that reinforces the concepts in the next class meeting. At this point students reflect, which helps to internalize the concepts and elicit student expectations, or *hypotheses*, for the utility of the concept. Then, students apply the concept in the context of a team-oriented, scalable project, and finally reflect again to (in)validate their earlier hypotheses. These activities all take place in the span of a single two-three week *sprint*, resulting is a highly iterative methodology for rapidly evolving student competencies (Fig 1).The Software Enterprise was derived from existing theoretical model called Kolb's Experiential Learning Cycle represented as a four-stage cycle (Concrete experience, reflective observation, abstract conceptualization, active experimentation) where learner is

exposed to all stages (Kolb, 2014). Software Enterprise was derived from Kolb's learning cycle by assembling best practices such as preparation, reflection, practice (labs), and project-centeredlearning in a rapid integration model that accelerates applied learning.

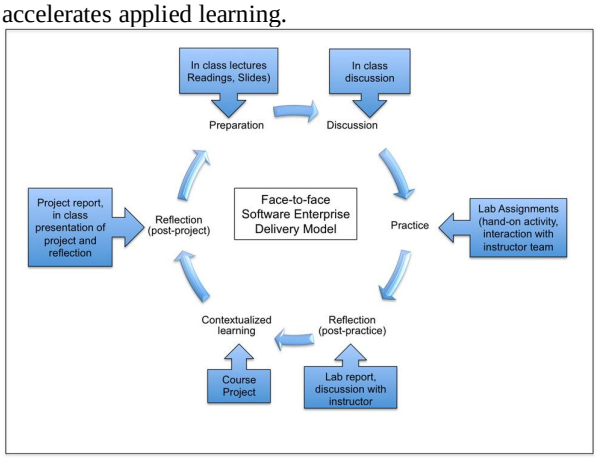

Figure 1:Software Enterprise Delivery Model

3. Course Description and Design

In this project we studied three courses from our B.S. in Software Engineering program. The first course is an introductory programming course, that introduces problem-solving to freshman (1st year) students, titled "CST100: Object-Oriented Software Development". The second and third courses are core Software Engineering courses, introduced during the sophomore (2nd) year of the degree program, titled "SER215: Software Enterprise I" and "SER216: Software Enterprise II". All of these courses have been developed for online delivery in addition to in-person delivery. This section describes the course goals, learning objectives (designed using an outcomes-based education model called PC^3 model(Andhare, Dalrymple, & Bansal, 2012; Bansal, Bansal, & Dalrymple, 2015; Bansal, Dalrymple, & Bansal, 2015; Mager, 1997)),course structure, and design.

A. CST 100: Object-Oriented Software Development

Object-Oriented Software Development (CST100) is a freshman course in the Software Engineering program that introduces problem solving with a state-of-the-art programming language. Expressions, statements, basic control flow, and methods are the broad topics introduced to students. Students are also exposed to data, data aggregation, and usage. This course uses a structured personal software development process to implement solutions representative of common computing applications. Development kits are used for some of the course activities. Basic concepts of object-oriented analysis, design, and programming using Python are covered. The students in the class study basic Python variables, expressions, arrays, statements, loops, functions, methods, and classes. Game development using a Python development kit called Pygame was introduced. A project-based pedagogical model is used for delivery of all our courses in Software Engineering program. Students in this course worked on a game project using Pygame. The learning objectives are as follows:

- *CO1*: Apply the concepts of sequence, selection, and iteration by constructing algorithms and formal code for problem solving
- *CO2*: Use variables and composite data structures to store and manipulate data by constructing algorithms and formal code to solve problems
- *CO3*: Use modular programming techniques such as functions and objects by constructing algorithms and formal code to solve problems
- *CO4*: Understand concepts of objects and types
- *CO5*: Apply a disciplined problem solving process to the construction of algorithms and formal code to solve problems
- *CO6*: Configure a software development environment for the construction of formal code to solve problems

B. SER 215: Software Enterprise I

Software Enterprise I: Personal Software Process (SER215) is designed to expose students to practical, real-world considerations in software development. It is a required/core course for B.S in Software Engineering program. Students learn in a hybrid lecture-lab-project environment, which exposes them to concepts and accelerates conceptual understanding in a project context. Lab assignments provide an opportunity for students to develop and enhance a defined process for their own work. Students are introduced to Software Engineering, software development life cycle (SDLC) models, object-oriented programming, personal software process, effort estimation and tracking, Defect estimation and tracking. The learning objectives are as follows:

- *CO1*: Design a software solution using object-oriented design principles of encapsulation, information hiding, abstraction, inheritance, and polymorphism.
- *CO2*: Develop a software solution in an object-oriented programming language employing standard naming conventions and making appropriate use of advanced features such as exception handling, I/O operations, and simple GUI.
- *CO3*: Use object-oriented design tools such as UML class diagrams to model problem solutions and express classes and relationships such as inheritance, association, aggregation, and composition.
- *CO4*: Use personal software process for individual development productivity through time estimation and tracking.
- *CO5*: Use personal software process for individual development quality through defect estimation and tracking
- *CO6*: Demonstrate teamwork

Students in this course worked on a java-based game project that involved building a board game using object-oriented design and personal software process.

C. SER 216: Software Enterprise II

Software Enterprise II: Testing and Quality (SE2) is designed to expose students to software testing and quality in software engineering. Students learn concepts, tools, and methods in testing and quality management; various testing

activities including unit, integration, system, and acceptance testing; white box and black box testing; code coverage; creation of a Software test plan; teamwork and communication in a hybrid lecture-lab-project environment. Projects are team-based and include multiple deliverables and presentations, with a specific emphasis on testing, validation, and quality assurance. The learning objectives are as follows:

- *CO1*: Define basic terminology of Software testing (e.g., fault, error, failure, debugging, test case) and activities in testing lifecycle (e.g., unit, integrations, system, acceptance testing)
- *CO2*: Compare various approaches for Integration testing
- *CO3*: Model Use case, Class, State, and Activity diagrams using UML
- *CO4*: Create a software test plan in the IEEE format and conduct testing for a given Software product
- *CO5*: Evaluate various Software testing tools
- *CO6*: Communicate effectively in writing a technical document, evaluating, and presenting software testing tools, and project presentation
- *CO7*: Demonstrate working effectively in small teams

Students in this course worked projects that involved working withexisting java codebase with a focus on Software testing. They created design models using UML, software test plan, executed their test plan to identify defects and enhancements, fixed the source code and implemented enhancements in the existing codebase.

4. Description of the Study

The design for the online courses was kept the exact same as the face-to-face courses. The Software Enterprise pedagogical model was used. As the delivery mode was completely online, the *preparation* phase had to be done completely by the students themselves using the materials provided. Discussion boards replaced in-class discussions as shown in figure 2.

A. Course Tools

The tools used for course delivery online were as follows:

- *Blackboard*: was used as the primary learning management system through which all interactions happened and course materials were shared.
- *Blackboard Discussion forums*: were used to discuss various aspects of the course.
- *CATME*: software tool was used to help with team formation through their team-maker survey as well as for peer review for projects.
- *Facebook Group*: was used informally by the students to discuss the course and socialize. The instructor team did not monitor this page.
- *Google Hangout*: was used by teams to meet and communicate with each other. This was also used to hold office hours during the 7.5-week semester.
- *GitHub*: was used by some teams to share their repository and code among team members.
- *Respondus Lockdown browser*: was used to administer the exams.

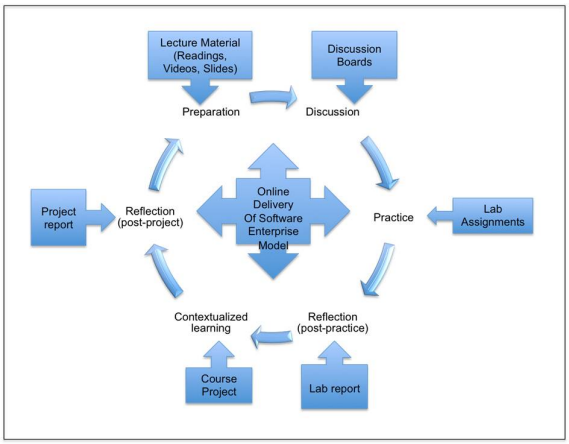

Figure 2: Software Enterprise pedagogy for online delivery

B. Setup of the Study

The study was conducted with CST100, SER215 and SER216 that were offered face-to-face and online. Number of students in the online course was double than that of the face-to-face with the exception of SER216 - follow-on to SER215 (shown in table 1). The number of online students reduced in SER216 and mostly the A and B grade scorers of SER215 returned to SER216 the subsequent semester. Others did not take SER216 immediately and instead registered during a later semester. The courses were taught with the exact same materials and assessments in both sections (taught by the authors). The same instructor and teaching assistant were assigned to both courses. The one big difference was the course duration. Online courses are 7.5-week courses while face-to-face are 15-week courses and hence double the material is taught to students every single week. Project sprints had to shrink down to 1-week sprints instead of being 2-3 week sprints.

Table 1: Experiment Design

Course	Semester	Number of students	Duration	Number of project teams
CST100 (face-to-face)	Spring 2015	74	15 weeks	n/a
CST100 (online)	Spring 2015	209	7.5 weeks	n/a
SER215 (face-to-face)	Fall 2014	44	15 weeks	15
SER215 (online)	Fall 2014	94	7.5weeks	30
SER216 (face-to-face)	Spring 2015	65	15 weeks	22
SER216 (online)	Spring 2015	36	7.5 weeks	12

C. Data Collection

The data collected for this study included the following:

- *Student Performance Data*: The grades of the students for each learning objective of the course were collected for both face-to-face and online. The format used for this is the faculty course assessment report (FCAR) (Estell, 2009). FCAR presents a methodology that allows instructors to write assessment reports in a standardized format for use in both course and program outcomes assessment.

- *Course evaluation data*: the institution collects this data from students in an anonymous manner to evaluate faculty and make improvements to future.
- *Course failure and dropout rates*: For each course in this study, the number of students who passed, failed, and withdrew was tracked.
- *CATME Peer review*: Students were sent out a survey to review their peers on their project teams. This review was incorporated into their grades for the project.
- *CATME Team-maker survey*: A survey was sent out to get student demographic information, their working style, location, background, etc. in an anonymous manner. This data was used by CATME to form teams.

5. Results And Discussion

Results from the data collected are shown in tables below.

- Figure 3 shows the success rate, failure rate, and dropout rate for all 6 courses. Both online and in-person offerings of CST100 have a 10% dropout rate. Being the first programming course a number of students are deterred by the cognitive nature of this course. Online offering of SER215 has a failure and dropout rate of around 10% much higher than in-person. This is the first core course in the Software enterprise sequence where students are exposed to teamwork for the very first time in the curriculum. Online students most likely don't understand the amount of work involved in computing and software engineering courses. For all our SE online courses the syllabus clearly states that about 18 hours per week of work is expected. But most students have full time jobs and families and are enrolled in multiple 3-credit courses during the same semester thereby resulting in a lower success rate. The drop and failure rate is much lower for the remaining courses.

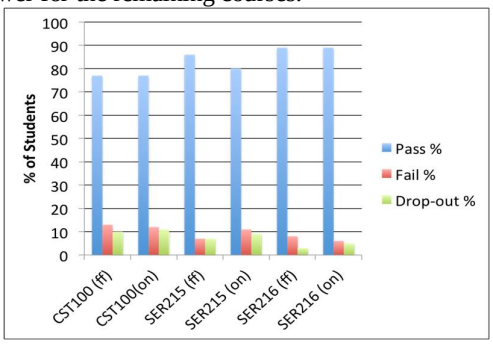

Figure 3: Course success, failure, dropout rates

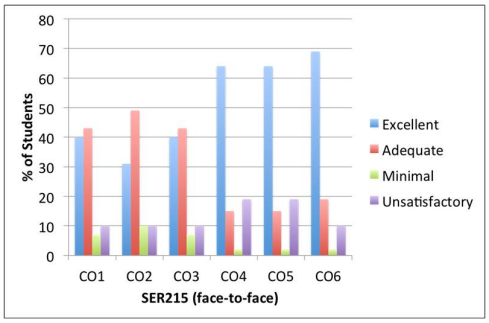

Figure 4: SER215 (face-to-face) Student performance

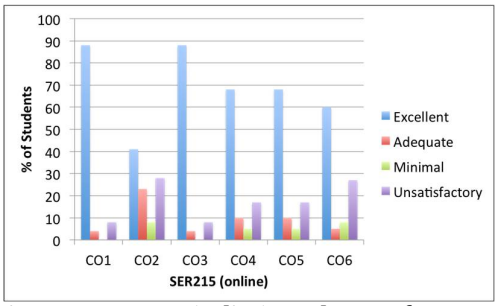

Figure 5: SER215 (online) Student performance

- Figures 4 and 5 present the student performance data for SER215 face-to-face and online delivery. The results show percentage of students who fell under the excellent, adequate, minimal, and unsatisfactory levels. The four categories are defined in the FCAR (Estell, 2009). The results indicate that for objectives 4, & 5 the results are almost similar for both course offerings (objectives related to PSP). Learning objectives 1 and 3 were mostly related to Technical outcomes of the program and results show that online students performed better. This raises concerns about the validity of these scores and the possibility thatstudents might be getting external help for assignments and exams. Another reason could be that a large part of the online student population are already working in the industry and have programming experience and are in this program to get a formal degree. Objective 6 associated with teamwork has better performance in f2f than online and poses challenge of how to effectively conduct projects in online courses.
- Figures6 and 7 present student performance data for SER216 face-to-face and online delivery. Success rate for course objectives involving teamwork are lower for online whereas students performed better on the technical objective similar to SER215.
- Table 2 shows course evaluations. The face-to-face courses have higher ratings than online in spite of having the exact same material and instructor for SER 215 & 216 that involve teamwork. Ratings dropped in SER215 (online) but got better in SER216 (online) – the follow on course which most likely indicates that students got used to the pedagogical model, instruction style, and project-based learning by the time they got to next semester.

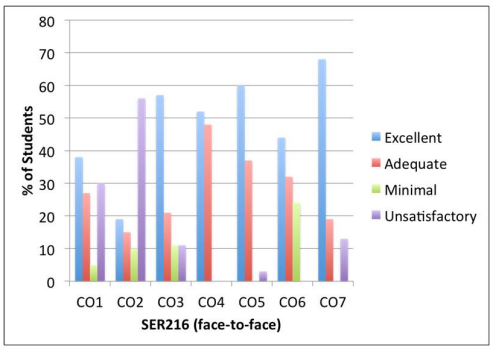

Figure 6: SER216 (face-to-face) Student Performance

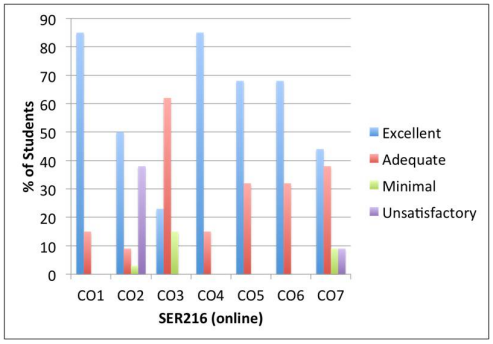

Figure 7: SER216 (online) Student Performance

6. Challenges and Lessons Learned

The main challenges identified with translation of course design for online education can be categorized as follows:*Assessment of teamwork*: Implementing a project-based course online is not trivial because of the nature of collaborative work that relies on people working together. It becomes much easier when students are in front of each other in class rather than having to synchronize meeting times and meet with team members virtually to work on the project. Through the team maker survey we found that a number of students had a full schedule and hardly had enough time for the course and to interact with the team. The student population taking online courses is different from those in f2f courses. So a detailed analysis of student population and learner behavior in the online collaborative learning environments might bring new insights.

Table 2: Course Evaluation data

Course	Semester	Overall Evaluation	Evaluation of Course	Evaluation of Instructor
CST100 (face-to-face)	Spring 2015	4.81	4.59	4.8
CST100 (online)	Spring 2015	4.42	4.53	4.67
SER215 (face-to-face)	Fall 2014	4.46	4.63	4.4
SER215 (online)	Fall 2014	3.88	3.42	3.03
SER216 (face-to-face)	Spring 2015	4.24	4.36	4.46
SER216 (online)	Spring 2015	3.93	4.27	4.21

Communication: All communication is virtual through email, discussion boards and occasional hangout/skype office hours. It is easier to motivate students to perform on the team and complete their tasks when instructor meets them in class. In online courses students can chose to interact or not on discussion boards unless discussions are also part of the assessment vehicle. In order to have good interaction in an online class, it might be useful to enforce participation in discussion boards in some way.

Student engagement: The level of student engagement and motivation is low for many students in online courses probably because of a number of commitments and full time job. Students take multiple courses during the same semester that makes it worse. The course is already at a fast pace because of the 7.5-week duration unlike the face-to-face course. So for future offerings of the online course, instructor has to explore new innovative ways of keeping online learners engaged in class.

Instructor satisfaction & Student Self-Efficacy: As part of this study data was not collected on instructor satisfaction with the delivery of the course and course outcomes as well student self-efficacy of the concepts that they have learnt. As future work we will implement these to learn more on what works in the online environment and what does not.

Teacher-Student interaction: Tracking teacher-student interaction and correlating it with student success rate might also be useful to explore as a future direction. It can be done through Blackboard learning management system.

7. Conclusions and Future Work

This paper presented the authors experiences translating project-based Software Engineering courses into online courses for complete asynchronous delivery. This study confirms that the exact translation of a traditional course to an online course can help maintain the quality of course and student performance. Challenges arise when instructional activities more hands-on training and team-based projects. Effective ways for administering team-based projects online, improving student teacher interaction, keeping students motivated and engaged are to be explored.

References

Andhare, K., Dalrymple, O., & Bansal, S. (2012). Learning Objectives Feature for Instructional Module Development System. Presented at the PSW American Society for Engineering Education Conference, San Luis Obispo, California.

Bansal, S., Bansal, A., & Dalrymple, O. (2015). Outcome-based Education Model for Computer Science Education. *Journal of Engineering Education Transformations*, *28*(2&3), 113–121.

Bansal, S., Dalrymple, O., & Bansal, A. (2015). Building Faculty Expertise in Outcome-based Education Curriculum Design. In *Proceedings of Frontiers in Education Conference (FIE)*.

Burge, J. (2015). Insights into Teaching and Learning: Reflections on MOOC Experiences. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 600–603).

Capdeferro, N., & Romero, M. (2012). Are online learners frustrated with collaborative learning experiences? *The International Review of Research in Open and Distributed Learning*, *13*(2), 26–44.

Cary, K. A. (2008). The software enterprise: Practicing best practices in software engineering education. *International*

Journal of Engineering Education, *24*(4), 705.

Estell, J. K. (2009). Streamlining the assessment process with the faculty course assessment report. *International Journal of Engineering Education*, *25*(5), 941.

Gary, K., Lindquist, T., Bansal, S., & Ghazarian, A. (2013). A project spine for software engineering curricular design. In *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on* (pp. 299–303).

Kolb, D. A. (2014).*Experiential learning: Experience as the source of learning and development*. Pearson Education.

Lin, P.-C., Hou, H.-T., Wang, S.-M., & Chang, K.-E. (2013). Analyzing knowledge dimensions and cognitive process of a project-based online discussion instructional activity using Facebook in an adult and continuing education course. *Computers & Education*, *60*(1), 110–121.

Mager, R. F. (1997). Preparing Instructional Objectives: A critical tool in the development of effective instruction 3rd edition. *The Center for Effective Performance, Inc*.

Phase, I. I., & others. (2005). *Educating the Engineer of 2020:: Adapting Engineering Education to the New Century*. National Academies Press.

Sheppard, S. D., Macatangay, K., Colby, A., & Sullivan, W. M. (2008). *Educating engineers: Designing for the future of the field* (Vol. 2). Jossey-Bass.

Sun, P.-C., Tsai, R. J., Finger, G., Chen, Y.-Y., & Yeh, D. (2008). What drives a successful e-Learning? An empirical investigation of the critical factors influencing learner satisfaction. *Computers & Education*, *50*(4), 1183–1202.

Tseng, H. W., & Yeh, H.-T. (2013). Team members' perceptions of online teamwork learning experiences and building teamwork trust: A qualitative study. *Computers & Education*, *63*, 1–9.