# Improving Retention and Confidence Through Cross-Course Collaborative Project-Based Learning

Jennifer Winikus, Lukasz Ziarek, Carl Alphonce, and Jesse Hartloff
Department of Computer Science and Engineering
University at Buffalo, State University of New York
Buffalo, New York 14260
Email: {jwinikus, lziarek, alphonce, hartloff}@buffalo.edu

***Abstract*—This work in progress introduces cross-course collaborative, project-based learning, or C$^3$PBL. C$^3$PBL builds on established collaborative project-based learning ideas that have been shown to improve retention at various institutions, by introducing collaboration that spans multiple courses in the same term. We discuss how such a collaboration between students in different courses can change the undergraduate educational experience to improve retention, community, and self-confidence. For our work we are focusing on a curriculum based approach, leveraging collaborative project-based learning experiences to affect the mindset and culture of our department. We present lessons learned from an initial pilot of C$^3$PBL deployed over two semesters. The pilot looks at the pairing of introductory level courses with an upper level software engineering class. The collaboration is centered on the idea of experienced students in the upper level software engineering course working with novice students from the introductory courses. The logistics explored has lead to more meaningful interaction and assessment in the second semester offering of C$^3$PBL.**

## I. Introduction

The 2013 Computer Science (CS) Curricula report emphasized as a critical need to prepare CS students for the workforce in a "more holistic way than simply conveying technical facts. Indeed, professional skills (such as teamwork, verbal and written communications, time management, problem solving, and flexibility) and personal attributes (such as risk tolerance, collegiality, patience, work ethic, identification of opportunity) play a critical role in the work place" [1]. To better prepare students for STEM careers, collaborative project-based learning (CPBL) has proven to be an effective pedagogical mechanism for developing both technical and soft skills [2], [3], [4].

Project-based learning is a documented approach that has had positive impacts on retention and diversity. Attrition in undergraduate education is a concern in all disciplines; with students leaving STEM bachelors degrees at a rate of 48%, business at 50%, social and behavioral sciences at 45% [5].

Retention improvements have been observed not only as a result of integrating projects into a curriculum [6], but also with the adoption of peer lead mentoring [7]. Motivation has also been documented to improve with the incorporation of projects [8]. Other approaches which have motivated our pedagogy of cross-course collaborative project-based learning includes leveraging paired programming [9], service learning [10], and collaborative learning [11]. Collaborative project-based learning has a strong presence is STEM courses [12], [13], [14]. The implications of success with this idea have been seen in entrepreneurship programs [15], [16], [17], across disciplines [18], [19], [20], with libraries [21], between special needs and mainstream [22], and between different institutions [23], [24]. Collaborative project-based learning [22] has not, however, been studied across different courses in the same semester.

We build on this evidence-based approach to STEM teaching and learning to increase the effectiveness of instruction in the Computer Science and Engineering (CSE) department at the University at Buffalo. We propose an innovative CPBL curriculum that brings together students from across levels, experiences, and courses, in a given semester. Unlike traditional CPBL which groups students within the same course, our CPBL pairs students in *different* but mutually synergistic courses within the *same semester*. Our version of CPBL will be termed *Cross-Course, Collaborative, Project-Based Learning* or C$^3$PBL. Our vision of C$^3$PBL lies at the intersection of three core design aspects: mentoring, projects, and collaboration (Fig. I). For a successful implementation, we believe those core design aspects all need to be considered and interdependent on each other. We believe finding the proper balance of the design in Fig. I of C$^3$PBL will create influence in retention, engagement, and recruitment that will then (1) improve STEM learning and learning environments, (2) help build the professional STEM workforce for tomorrow, and (3) broaden participation and institutional capacity for STEM learning.

## II. Research Questions

Though our diverse experiences as students and faculty, we considered where we see some weaknesses in the educational experience for the students that we wanted to improve. We identified weaknesses in retention, collaborative skills, inclusivity, and communication skill. These are all aspects which strongly tie to what we view as a solid foundation necessary to support persistence in the computing field. We have developed the following research questions to explore with our approach of C$^3$PBL.

1. How has the collaboration influenced the students attitudes and perceptions in regards to self-confidence, inclusivity, and stereotypes? 2. How has the collaboration enhanced the

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

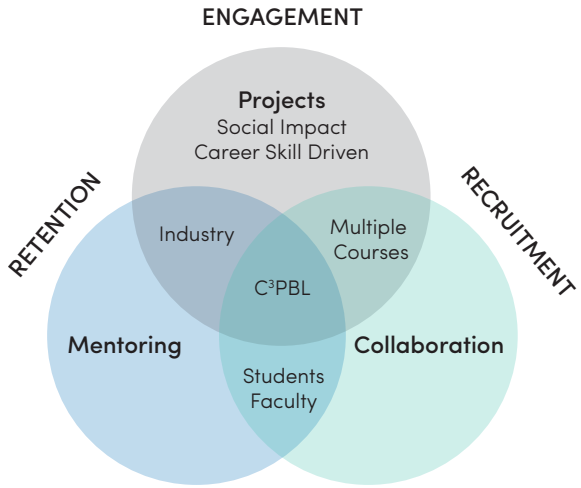


Fig. 1. C$^3$PBL dimensions of influence.

learning objectives for the collaborative project? 3. To what extent are the students able to be successful in project environments that are bigger than the scope of a single class?

These research questions will be evaluated through the use of surveys and the student performance in their projects.

## III. C$^3$PBL Course Pairings

The first step in setting up a cross-course collaboration is choosing courses which can lead to productive interactions. Once a synergistic course pairing is established project details and assessment artifacts can be developed to address the research questions. In our pilot pairing we considered ways in which the pairing would benefit students in both courses.

### *A. Piloting C$^3$PBL case study – Introductory Courses and Software Engineering*

In this cross-course collaboration we paired students from our introductory courses (CSE101 *computer literacy*, CSE113 *introductory programming for non-majors*, and CSE115 *introductory programming for majors*) with students from our senior-level software engineering course. The introductory students play the role of customer for software projects developed by students in the CSE442 *software engineering* course.

In more details, CSE442 students present their semester-long projects to the introductory students, during various phases of development, during lab sessions. The introductory students are tasked with developing a testing procedure to evaluate the software and identifying changes, via a set of requirements, which they believe should be enacted in the software based on their testing. The CSE442 students will leverage the results of the testing and requirements to evolve and maintain their software.

Since the introductory students do not have much experience writing software, playing the role of the customer will help them understand their eventual role as developers and gain insight into the software development process (including but not limited to its collaborative nature and especially soliciting of user requirements), testing of software, as well as reviewing incremental software releases. All of such concepts present transferable skills which can be applied to other majors, creating a meaningful experience for all students. The software engineering students will learn to interact with realistic (technically naive) customers and learn the software development process. They will need to gather user requirements, testing results, and provide incremental software builds for evaluation by the introductory course students.

Through this pairing we have benefits to the introductory students through the introduction of testing concepts, design procedures, experiences to expect as they progress in the degree program, and communication between developers and clients (which is the role that the intro students take). The software engineering students are designing projects that are aimed for general use, so by including the introductory students as clients, they have a population with different perspectives to learn to communicate with and to get genuine feedback.

### *B. Introductory Courses*

We have three introductory courses that are participating in the collaboration. There are two versions of the introduction to programming course, one for students intending to pursue computing majors and the other for the population of students from other majors to learn problem solving skills. The other group is the computer literacy course that explores many facets of technology, including societal aspects.

For the two programming sections, we determined the primary learning objective is the testing behavior. The idea of focusing on an aspect of the project, developing a process to test for that target, and defining an expected outcome. This is woven into the courses with the development of debugging skills, and enhanced with the communication factor from working with real projects.

The literacy course also focuses on testing, but not in the sense of programming, but in the development of websites and interaction expectations. The most important part though for these students is the focus on the communications between developer and client.

### *C. Software Engineering*

Our software engineering course examines in detail the software development process.

> Topics include software life-cycle models; architectural and design approaches; various techniques for systematic software testing; coding and documentation strategies; project management; customer relations; the social, ethical, and legal aspects of computing; and the impact of economic, environmental, safety, manufacturability, and sustainability factors on design. Students in this course participate in a real-world project from conception to implementation.

To be successful with the concept of real-world projects, we believe that a genuine audience from the "real world" offers an enhancement to the project. This then ties into the customer relations learning objective, where testing phases work with the customer for feedback.

## IV. Logistics of Implementation

To develop the implementation structure for the projects, there are some logistics necessary to support the instructional aspect. All classes have structured class schedules and learning outcomes that need to be satisfied. We explore challenges we encountered and some potential initial solutions we have put into practice.

As with many institutions around the country [25], our computing programs have experienced substantial growth. This poses several challenges, including how to manage the 26 teams of students in the software engineering course and over 500 students in the introductory courses participating in the pilot.

In our case, we have various introductory courses participating to allow for insight into the different populations in the interaction. This is important for us as the department is in the process of transitioning to a single introduction to programming course for majors and non-majors as well as evolving the computer literacy course to be more relevant to interactions that a diverse group of majors may have, which in future careers may or may not involve technology.

## V. Activities for the Pilot Pairing

We have identifies three primary means of interaction between the introductory students and the software engineering students.

### A. Initial Documentation Review

At the start of the semester, software engineering students develop documentation which acts as the foundation for the project they will be implementing in the course this semester. This documentation is expected to be user-friendly to a general population; the introductory students review these documents and provide feedback.

### B. Development of Testing Approach

When testing projects, there are many objectives to assess. We use a variation on user stories: more narrowly focused use statements, but less technical than use case statements. In these statements the software engineers present an interaction that the user will expect to have with the system. From these statements the introductory students will distill the statement to identify what the target of the test will be, then a procedure to test the concept, and finally an expected outcome for the test.

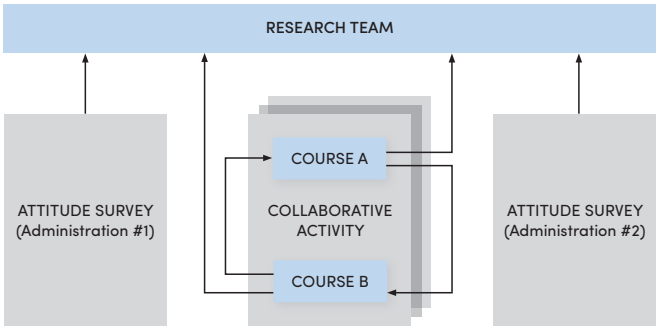


Fig. 2. Evaluation Plan

### C. A variation on Beta Testing

In preparation for the interaction, the software engineering team produced documentation from which the introductory students developed a testing plan. The software engineering teams start a face-to-face session with the introductory students by giving a brief introduction to the project. The introductory students then put their testing plans into action: they use the statements they developed and ones provided by the software engineering students to evaluate the project.

## VI. Assessment

Through the collaborative experience there are three phases of assessment. Fig. 2 shows the administration of student attitude surveys bookending the process, with the collaborative activity and its internal assessments in between.

### A. Attitudes

Using the attitude survey from North Carolina State University [26], we evaluate the student's attitudes towards computing, specifically self confidence and bias. A key aspect of the survey design is the positive and negative connotations of the questions to allow for the interpretation of the sincerity of responses. Students who respond the same answer for a question that is a positive view and a negative view (or the opposite) were not paying attention to the details or just clicking through.

### B. Experience

After the "Beta Test", participants are surveyed on their experiences and how it influenced their learning outcomes. The questions were categorized into experiential impacts on motivation, learning outcomes, and perceptions of community.

Motivation for the upper level students aimed at the importance of collaborating with populations to support the software development process, while the introductory students considered the personal understanding of roles in the computing discipline and their intentions to continue their computing education.

Learning outcome evaluation focuses on how the collaborative experience has influenced their communication skill and project development skills. These skills we examine in the learning outcomes, while the project is software development

focused, are able to transcend disciplines. This transcendence is a key aspect of the feasibility of incorporating these activities into the diverse introductory population.

The survey of the community perspectives asks the students to reflect both on themselves and their community. The traditional demographic questions are asked (gender, race, and age), and we presented the students with several open ended questions. The questions examined the students' definition of diversity, how they choose to self identify, and key experiences during the interaction pertaining to diversity and inclusion. While analysis of open ended questions is challenging, we believe such questions will allow us to better understand the the community as the students perceive it.

### C. Artifacts of Activity

These activities are integrated to the structure of each course, and as such there is a need to evaluate each student's success with the activity to ensure that learning objective are met, regardless of the interaction experience. These artifacts will be the conceptual basis for the development of the desired artifacts for the fall 2018 implementation.

## VII. Lessons from Pilot Development

In fall of 2017, as we started the first pilot implementation of C$^3$PBL, we quickly learned that our formulation needed more depth in the interactions so that the interaction experience would be meaningful to all students. Our initial implementation focused on the beta test interaction, with the software engineering students structuring the interaction during the test. This was found to not be meaningful due to the lack of structure given by the software engineering students. The software engineering students did not know how to efficiently run a beta test and the introductory students did not know how to perform testing to give useful feedback. Additionally, we found our survey questions on the experience did not provide adequate insight into the meaningfulness of the interaction for the learning outcomes.

In the spring 2018 implementation activities which have been completed thus far, we have learned that there needs to be more development of the evaluation artifacts for the initial documentation produced by the software engineering students and the evaluation methods used for the introductory students. While we successfully developed documentation for the creation of testing statements, we still need to improve the structure for the documentation that is used going into it.

## VIII. Future Work

The pilot has provided valuable insight into C$^3$PBL logistics, which we leverage when we implement the new curriculum in the department. We plan to use the outcomes observed this semester to continue to improve C$^3$PBL in our current pairing and to implement the developed pedagogy in additional pairings. The targets for this expansion are a pairing of synergistic courses that are primarily computer engineering with ones that are focused on computer science. We plan this expansion to start with our compilers course and our computer architecture course, and a pairing between our discrete structures course and our digital systems course. The main focus of both of these pairings is on the collaboration between computer engineers and computer scientists, and more professional skill development than the retention driven activities done with the pilot pairing.

## References


[1] "Computer science curricula 2013." http://www.acm.org/education/CS2013-final-report.pdf.

[2] D. R. Garrison and J. B. Arbaugh, "Researching the community of inquiry framework: Review, issues, and future directions," *The Internet and Higher Education*, vol. 10, no. 3, pp. 157–172, 2007.

[3] P. Ram, "Problem-based learning in undergraduate education," *Journal of Chemical education*, vol. 76, no. 8, p. 1122, 1999.

[4] E. Seymour, "Tracking the processes of change in us undergraduate education in science, mathematics, engineering, and technology," *Science Education*, vol. 86, no. 1, pp. 79–105, 2002.

[5] National Science Board, *Science and Engineering Indicators 2016*. 2016. (NSB-2016-1).

[6] D. W. Knight, L. E. Carlson, and J. F. Sullivan, "Staying in engineering: Impact of a hands-on, team-based, first-year projects course on student retention," in *Proceedings, ASEE Conference and Exhibition*, 2003.

[7] V. Pierce, J. A. Kypuros, A. A. Fuentes, H. Vasquez, and S. W. Crown, "Technology-enabled, after-hours, asynchronous, peer-led supplementary instruction and mentoring in engineering gatekeeper courses," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[8] A. Danowitz, "Leveraging the final project to improve student motivation in introductory digital design courses," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[9] A. Y. DiLillo, M. Altebarmakian, and R. Alterman, "The microgenetic analysis of staged peer collaboration for introductory programming," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[10] K. Vernaza and L. Zhao, "Teaching first year engineering students engineering design process and problem solving through service learning projects," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[11] M. Kirk and C. Zander, "Bridging the digital divide by co-creating a collaborative computer science classroom," *J. Comput. Sci. Coll.*, pp. pp. 117–125, 2002.

[12] E. Seymour, "Tracking the processes of change in us undergraduate education in science, mathematics, engineering, and technology," *Science Education*, pp. pp. 79–105, 2002.

[13] P. Ram, "Problem-based learning in undergraduate education," *Journal of Chemical education*, p. p. 1122, 1999.

[14] D. R. Garrison and J. B. Arbaugh, "Researching the community of inquiry framework: Review, issues, and future directions," *The Internet and Higher Education*, pp. pp. 157–172, 2007.

[15] K. E. Hersch, D. Piovesan, and A. Schmitz, "Using external business plan competitions to drive innovation and effective cross-disciplinary collaboration," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[16] R. Shinnar, M. Pruett, and B. Toney, "Entrepreneurship education: attitudes across campus," *Journal of Education for Business*, pp. pp. 151–159, 2009.

[17] Brinkman, T. M. Vitolo, K. E. Hersch, and B. J. Brinkman, "Making the connection: Successful cross campus collaboration among students," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[18] D. Piovesan and N. Morris, "Integrated crayons for adaptive needs," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[19] U. Poerschke, R. J. Holland, J. I. Messner, and M. Pihlak, "Bim collaboration across six disciplines," in *Proc., Int. Conf. on Computing in Civil and Building Engineering*, 2010.

[20] Y. R. Chassiakos, L. Rubino, and B. Freshman, "Collaboration across the disciplines in health care," 2010.

[21] P. McPherson and M. Phillips, "Using everyday objects to engage students in standards education," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[22] A. L. Williams, R. O. Sattler, and D. J. O'Shea, "Collaboration across special education and general education: Preservice teachers' views," *Journal of Teacher Education*, pp. pp. 147–147, 1999.

[23] S. Rouvrais and J. Bennedsen, "Finding good friends to learn from and to inspire," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016.

[24] O. P. Brenton, S. Less, R. Bedson, C. Boldyreff, S. Drummond, P. Layzell, L. Macaulay, and R. Young, "Student collaboration across universities: A case study in software engineering," in *Proceedings. 13th Conference on Software Engineering Education & Training*, 2000.
[25] T. Camp, W. R. Adrion, B. Bizot, S. Davidson, M. Hall, S. Hambrusch, E. Walker, and S. Zweben, "Generation cs: The mixed news on diversity and the enrollment surge," *ACM Inroads*, vol. 8, pp. 36–42, July 2017.
[26] E. Wiebe, L. Williams, K. Yang, and C. Miller, "Computer science attitude survey," *computer science*, vol. 14, no. 25, pp. 0–86, 2003.