A Connected Course Approach for Introduction to Engineering Problem Solving

Dr. Anthony Ferrar, Temple University

Tony Ferrar is obsessed with student success. He focuses on preparing students for rewarding careers through pedagogical innovation and incorporating professional development into educational experiences. Anthony received his BS, MS, and PhD in mechanical engineering from Virginia Tech, where his research revolved around air-breathing propulsion. As a graduate student he contributed to Virginia Tech's Graduate Education Development Institute, Faculty Development Institute, and Networked Learning Initiatives. After graduating in 2015, he joined the BEARS Lab (B&E Applied Research and Science) in the nuclear engineering program at the University of Florida as postdoctoral researcher where he investigated spent fuel storage and cancer treatment. Throughout his graduate and postdoctoral experiences he participated in teaching, student mentorship, and faculty development as an instructor and advocate for learning innovation. He joined the Temple University faculty in 2015, where he focuses on Engineering Entrepreneurship, Social Networking and Connections in Higher Education, Peer-to-Peer Mentorship, and Open and Inclusive Education.

Pete Watkins, Temple University

Pete Watkins is an Associate Director in the Center for the Advancement of Teaching where he facilitates faculty development programming on a range of topics including a focus on online learning. In addition, he is a faculty member in the Teaching in Higher Education Certificate. He has worked in higher education since 2000 and has extensive teaching experience, both face-to-face and online, as well as expertise in course design, curriculum development and accreditation. He earned a bachelor's degree in psychology from the University of Pennsylvania, a master's degree in social work from Temple University and is currently pursuing a Ph.D. in educational psychology from Temple University. He has a particular interest in using cognitive science to inform and improve college teaching. Pete believes that excellent teachers continually grow, develop and learn from one another.

A Connected Course Approach for Introduction to Engineering Problem Solving

Introduction

This Work in Progress paper summarizes preliminary results of a new college-wide first-year engineering problem solving course. Using computer programming as a vehicle, the goal of the course was to enhance students' ability to solve novel real-world problems. Students learned design-thinking, opportunity recognition, and customer discovery in collaboration with the Innovation and Entrepreneurship Institute, housed in the College of Business. Students identified an issue, empathized with potential customers through interviews, defined a specific problem that could be solved with software, brainstormed solutions, created prototype smartphone apps, and tested them on intended customers.

Key Constraints

This course had no prerequisites and was one of two required college-wide courses for first-year engineering students. The challenge was to prepare students for success in all of the engineering disciplines offered by the college without requiring background in math or physics. This course is a companion to another first-year course called Introduction to Engineering. The two courses are taken in either order as students encounter the other courses typical of a first-year experience (calculus, physics, sciences, general education).

Learning Objectives

The course focused on three key learning objectives which were selected to be supportive of all of the engineering disciplines offered within our college. The first revolved around Engineering Problem Solving Skills: *"A successful student leaving this course will identify a human need that can be met with code, and design a functioning solution."* Students learned to recognize opportunities, identify goals, find relevant missing information, implement design thinking, and formulate a solution. Coding was merely the vehicle for teaching the widely-applicable creative problem-solving framework.

The second learning objective addressed Engineering Programming Fundamentals: *"A successful student leaving this course will develop a working Android, iOS, or Web-based App."* Students learned ten key programming skills including Variables, Data Types, Console I/O, Functions, Debugging, Operators, Conditional Code, Flow Control, Loops, Objects. In addition, students employed integrated development environments for their final projects. Students were given a choice between these three platforms based on interest and resource availability (for example, developing for iOS requires access to an Apple computer).

The third learning objective focused on developing the soft skills required to flourish as engineers: *"A successful student leaving this course will create a personal learning network."* Students learned to work effectively in teams, communicate clearly, consider complex perspectives and tradeoffs, and how to seek information outside of their experience. The topics of self-discipline, time management, self-paced learning, and persistence were also emphasized.

Guiding Philosophy
Taught in a single section with up to 240 students enrolled, our goal was to leverage the diversity of perspectives and interests to create an experience that was enhanced by its large size. The course involved a team of 40 Undergraduate TAs (UGTAs) who served as mentors and facilitators to the first-year students as they navigated this self-paced class. UGTAs were selected over Graduate TAs due to two factors. First, we wanted the students to identify with their mentors. Second, we needed to scale: the job requires connecting with individuals regularly. This was more readily accomplished by a large team working a few hours rather than a small team working a large number of hours. The philosophy behind the course design stemmed from Seth Godin's apt words on experiential education: "The process involves selling the student on the mission, providing access to resources, and then holding her responsible for an outcome that works. And repeat. And repeat."[1]

Course Content
To emphasize the difference between language-specific syntax and general concepts involved in problem-solving with code, the students learned three different languages. The 10 key concepts were taught first in JavaScript and then repeated in Python. By repeating the same challenges in two languages, students were encouraged to focus on learning concepts rather than syntax.

JavaScript was chosen as an effective introductory language because it requires no new software to write and evaluate code – all computers can create using this language via a text editor and a web browser. Python added the need to install a new language and development environment, but maintained the scripted (vs compiled) nature of JavaScript to limit the number of changing variables. Students ultimately learned a third language which they selected to complete their project. A Core Skills Demonstration served as a midterm exam in which students demonstrated proficiency in both JavaScript and Python.

Students next moved into project-specific learning where student teams of approximately 10 members proposed projects as the UGTAs and faculty provided guidance and scope. Group size was chosen as a compromise between budget and meeting students' needs. In future iterations, this is a variable that should be explored intentionally. The projects began with student teams interviewing customers and creating value proposition statements. They used this information to create a required feature list for the Minimum Viable Product (MVP) version of their app. Next, they created a paper prototype as a way to quickly test their GUI design. The design phase concluded with laying out the GUI using the development studio for their project languages.

The next phase of the project involved writing the code required to make the MVP features functional, and adding these functions to their GUI. They moved on to debugging and revising the project, while adding any additional features that were identified as opportunities along the way. Finally, the project concluded with a fully-functional GUI, customer testing and feedback, and a live demo of the app during a pitch competition. The pitch competition was developed in partnership with the Innovation and Entrepreneurship institute and invited a panel of entrepreneurial judges to evaluate students' pitch presentations.

Selected Sample Projects
The project requirements were to determine a need that could be met via software, select the best platform for the solution (Android, iOS, or Web App), and create a functioning prototype as a group. Table 1 summarizes three examples of projects which students created.

Table 1. Sample projects and languages used.

Platform	**Language**	**Project description**
Android Mobile	Java	An app that enables a user to enter their refrigerator and pantry inventories and then display recipes which could be made using these ingredients. Advanced features include inventory tracking (deducting quantities when a recipe was actually used to cook a dish), and “one more item” feature which suggests recipes if the user went to the store for one more item. *Note about scope: the app produced for the course required users to enter their own recipes into a database. A full-featured version would include an aggregator feature which searches the web for open recipes.*
iOS Mobile	Swift	An app that enables users to enter their class schedules and post times and locations that they plan to study for courses. The app assists users in finding study groups and locations on campus. *Note about scope: the app for the course utilized a pre-built database of simulated users, schedules, and locations. The full-featured app would connect to a server which would accept users input and access to actually create the platform.*
Web App	Python, HTML, CSS, JavaScript	An app enabling users to hire other users to do their laundry, similar to peer-to-peer ridesharing services. A customer would hire a “washer” to pick up their clothes, wash them (using the “washer’s” equipment), and return them to the customer. *Note about scope: the app for the course utilized a pre-built database of simulated users and “washers” without actual connectivity. A full-featured app would connect to a server to actually facilitate transactions.*

Course Structure
Several key features of the course structure enabled an experience of this scale to take place.

In the Classroom
This course was taught in a large lecture hall with 140 students enrolled in the first offering, and 240 in the second. Students were divided into teams of 10 and each team was assigned a UGTA team leader who sat with their students to offer individual attention and facilitate activities. Class sessions blended lectures, demos, and group activities. Every Friday the teams worked to solve Hackathon problems - challenges which required teamwork and creativity to succeed. The Hackathons helped the team leaders to identify which topics were most challenging for students,

provide one-on-one instruction, and give the course instructor a summary of key topics to highlight in future class sessions.

The Friday Hackathon problems were based on the 10 key concepts mentioned above, and could be solved in any language – simultaneously solidifying concepts and preparing the students for their final projects. Teams used a web-based code evaluation platform called HackerRank to solve these problems and obtain real-time feedback.

Class time was also used for team-building and guest talks from industry partners and the Innovation and Entrepreneurship Institute who provided real-world coding challenges as well as teaching students about how to succeed in engineering.

Out of the Classroom
The majority of the coding-specific learning occurred out of the classroom as students worked through open-source competency-based online training modules where students unlocked the next challenge by completing the current one. Their coding assignments were housed in an automatic evaluation and feedback system called HackerRank. This tool, and others like it, enable common errors to be identified and specific corrective feedback offered instantaneously. Students were given unlimited attempts on the assignments, and allowed to work at their own pace.

Students were not expected to work completely independently. The UGTAs held walk-in office hours, responded to email inquiries, and utilized social media to provide tips and tricks, and give demonstrations when common issues were encountered.

Tying Together
The students performed three main tasks during the course: online self-paced programming lessons, Friday Hackathons, and the final project. The self-paced programming lessons comprised the bulk of the first half of the semester, with the Friday Hackathons acting as anchor points to encourage students to stay on schedule and form effect teamwork within their groups. After the midpoint of the semester, the efforts shifted from self-paced programming lessons to group work on the final project.

Measuring Student Success
Student success was measured across the three key activities. The self-paced programming lessons ended with graded competency-based assignments in which students would solve coding challenges related to the 10 key concepts. Students were given unlimited attempts on these assignments, which they completed as individuals. The Hackathons were scored as a group, and anyone who contributed to the effort in a substantive way was given credit. Lastly, the projects were group assignments in which all contributors received credit. In future iterations, it is suggested that students be given individual scores for these assignments based on TA observation and peer reviews (perhaps using a system such as CATME [2]).

Challenges

With the multitude of moving parts, we experienced several key challenges in developing and implementing this course.

Self-Paced Learning
Because students were able to go at their own pace, the online learning modules needed to be live on the first day of class. This made iterating and adjusting as the course progressed difficult. We addressed this by tasking the UGTAs with a substantial role in the development of robust test cases for each coding assignment, and in the debugging of the assignments as students encountered them.

The other issue with self-paced learning revolved around trying to synchronize students so that they could work on the semester project as a team. If a student struggled with a topic, they found themselves sinking into deeper and deeper holes while their teams left them behind. We addressed this in the second semester by using the midterm assignment as a deadline for the learning modules and increased monitoring from the UGTAs. While the results improved, this is an area which requires further development.

College-wide Considerations
Another challenge in developing this college-wide course was creating an experience that served all of the disciplines. It was difficult to map certain disciplines to typical beginner-level coding challenges, which led to some students feeling unmotivated. It is important to build consensus among the faculty and administration from all departments regarding the learning objectives and major topics of the course. One possibility would be to launch the course as a smaller pilot elective to debug its major features before making it a required course for all engineering students.

Team Building
The biggest challenge with getting something of this magnitude up and running is forming the staff team. The second semester consisted of 240 registered students served by 40 UGTAs. The large team was needed to cover the various support roles during the week, though most of the UGTAs worked only a few hours. This large team required leadership, management, and support. In the future, certain roles could be combined to streamline the effort.

First-year student demands
The first-year engineering curriculum is extremely demanding and this course increased the pressure on an already challenging semester. Reconsideration of the number of languages, the scope of the projects, and the schedule of the course need to be made to target the learning objectives more efficiently. This is especially important in addressing the self-paced learning issues in which some students fell very far behind and gave up on the course altogether.

Lastly, messy real-world problems can be intimidating for students. The challenges they faced in this course were (intentionally) open-ended. Many of the assignments were ill-posed and asked the students to first answer the question “what really matters?” While this is an important aspect of problem solving, careful mentorship by the UGTAs is needed to reduce anxiety while maintaining the effectiveness of the learning experience. The goal is to avoid contrived problems and leverage faculty and TA guidance to keep things messy but manageable.

Outcomes
One question this course explored is whether large class size can be an asset. Typically, as class size increases, the quality of the students' experience decreases. This was a first experiment into using the largeness as an asset: interdisciplinary ideas, collaboration, teamwork, peer-to-peer mentorship, and a unifying first-year experience in which student feel they are part of an event, a cause, and a movement. In addition, we created a peer-mentoring scheme comprised of value-added experiences for both the students and the UGTAs. Students received individual attention. UGTAs learned leadership, teaching, and communication skills that prepared them for their careers with transferable skills.

Anecdotally, the peer mentoring model was the most impactful aspect of the course design. UGTA's reaped many benefits ranging from an increased sense of ownership and belonging as engineering students to developing the often-overlooked "soft" skills needed to thrive as professionals. This model is now being explored for use in the companion course, Introduction to Engineering, which students take in the alternate freshmen semester (the sequence can be taken in either order).

Perhaps the most important result can be found a year later in our student study spaces, dorms, and off-campus housing: the teams formed in this course can be found living and working together as friends and colleagues after having forged meaningful connections during their time in Introduction to Engineering Problem Solving.

References
[1] S. Godin, *Stop Stealing Dreams (What is School For?),* 2012 [E-Book]
[2] Catme.org. (2019) *CATME Project*. [online] Available at: https://www.catme.org/login/index [Accessed 24 Apr. 2019]