ASEE CoED Journal
Published 1 March 2024

Using Active Learning to Connect Entrepreneurial Mindset to Software Engineering

Ben Tribelhorn[1*], H.E. Dillon[2], Andrew Nuxoll[1], and Nicole Ralston[3]

[1]School of Engineering, University of Portland
[2]School of Engineering & Technology, University of Washington Tacoma
[3]School of Education, University of Portland

ORIGINAL

Abstract

The purpose of this research was to develop classroom project modules that supported students in developing an entrepreneurial mindset in the context of software engineering. The modules connect the software development life-cycle from beginning to end including user focused requirements elicitation and evaluating quality attributes. The modules were implemented in a junior level software engineering course, and three modules were surveyed in 2019 as part of a school-wide effort to embed entrepreneurial mindset into engineering curriculum. An IRB approved, student survey was developed and measured student perceptions of learning objectives that tie directly into ABET accreditation outcomes. Students reported they found the activities most helpful for designing, building, and testing real world systems.

The development of these modules was a component of increasing the process focus of the software engineering course by implementing a novel version of agile software development with active learning techniques. The practical experiments in this course taught from 2017 through 2021 allow us to report extensions and variants for adapting this design to existing software engineering courses at other universities. Among these variants we propose adopting class-wide teams which is atypical at other universities for junior-level project courses.

Qualitatively, we found that the student work completed in these modules to be higher quality than similar work submitted in prior years. Exam scores were improved when measuring students ability to create use cases, especially clarity and completeness. Student performance was greatly improved when writing use cases, especially clarity and completeness which was reflected in improved projects. Quantitatively, the same mindset objectives were assessed in other course modules as part a larger curriculum wide effort in Engineering. The numerical results indicate that the modules in this course outperformed other modules in the curriculum for most of the mindset objectives. Ultimately, the results indicate these types of modules may play an important role in entrepreneurial mindset development for computer science students.

Keywords: Software Engineering, Active Learning, Entrepreneurial Mindset, ABET, Computer Science

OPEN ACCESS
Volume
14

Issue
1

Corresponding author
tribelhb@up.edu

Submitted *6 Sept 2021*

Accepted *8 May 2023*

Citation
B. Tribelhorn, H.E. Dillon, A. Nuxoll, N. Ralston. Using Active Learning to Connect Entrepreneurial Mindset to Software Engineering. Computers in Education Journal, vol. 14, no. 1, 2024.

1 Introduction

This paper describes a set of modules designed to develop an entrepreneurial mindset (EM) in computer science students. It also contrasts these specifically detailed modules to an engineering-wide initiative to improve EM across the curriculum at the University of Portland. An entrepreneurial mindset is defined by the Kern Entrepreneurial Engineering Network (KEEN) as supporting students in developing advanced skills and mindsets to equip students to create personal, economic, and societal value (KEEN). Traditional computer science education often focuses on technical and collaboration skills. The mindset is a critical skill to develop in a software

engineering course as many students can rapidly create novel applications as demonstrated by the rise and success of mobile apps. This class was redesigned to build a set of skills and mindsets to focus the students on creating value.

In the context of this work, Entrepreneurial Mindset is not about creating businesses or pushing students into working for small businesses or startups. Instead it is a set of important mindsets that are critical to success as an engineer. Substantial prior work has shown that this mindset is important for helping engineering and computer scientists to create value through their work (Bosman et al., 2018). As so much of software engineering is focused on creating value for the customer, this mindset is well aligned to typical instructional goals.

The modules described in this paper, and contextualized in purple in Figure 1, have been placed in a junior level Software Engineering course that is required for all majors and commonly taken by minors. The prerequisite courses are Data Structures and an Object Oriented Design course that includes a large semester long project. This course widely covers the software design life-cycle (SDLC) including requirements, project management, design methods, integration, quality assurance, testing, maintenance, and tools. It is also a key course used for ABET assessment. Our approach to software engineering is to teach the agile process using Scrum. We describe the implementation of the top six agile techniques used in industry (daily standup, sprint planning, retrospectives, sprint review, short iterations, planning poker) which focuses the learning experience on the most important components of agile development in addition to including top engineering practices used in industry.

Agile software development is widespread in industry, however it is not widespread in computer science education. According to a corporate survey, the 13th Annual State of Agile Report (agi, 2019), “97% of respondents report their organizations practices agile development methods.” This development process is a missing skill set for most computer science undergraduates entering industry. Agile is a very general set of guidelines encouraging iterative software development, and Scrum is a variant of agile that aims to be lightweight and is therefore appropriate in the context of a course.

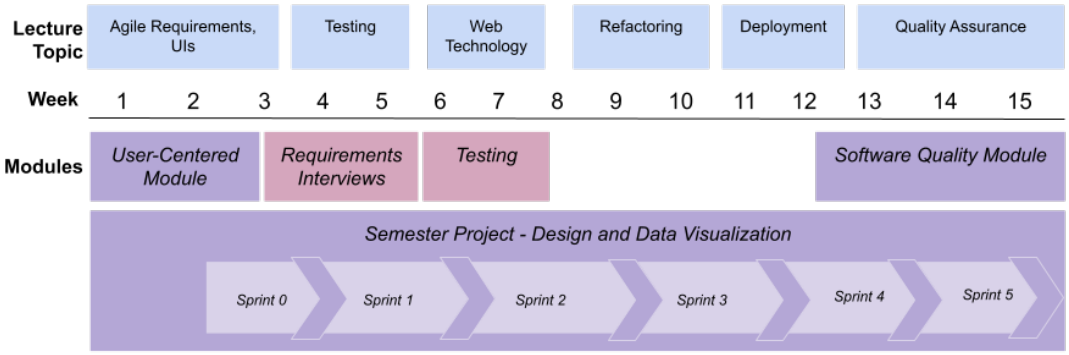

Figure 1. Summary of the surveyed EM modules (purple) and other active learning based interventions (pink) discussed in this paper, in the context of the course timeline.

The modules developed had several learning objectives focused on the entrepreneurial mindset:

- Students will identify and describe links between course knowledge and real world systems. [connections]
- Students will improve their ability to self-reflect and evaluate preconceived ideas, thoughts, and accepted solutions to recognize opportunities. [curiosity]
- Students will increase their ability to understand the ramifications of design decisions. [connections]
- Students will integrate engineering solutions by creating use cases [connections]

- Students will appreciate the value of quality in software development. [creating value]
- Students will design, build, and test real world engineering systems. [creating value]
- Students will increase their ability to identify and evaluate sources of information. [connections]

The modules also contribute to a set of technical course objectives:

- Students will understand that to be a professional software developer, one must always be learning new technologies, quickly.
- Students will learn real world development technologies.
- Students will experience first-hand a test-driven, medium-sized software development project.
- Students will learn and apply the software development life-cycle.
- Students will reflect on professionalism in software development.
- Students will increase their ability to work in teams and communicate technical information.

Research Question: The research goal of the project was to determine if a set of structured modules in a computer science or engineering course could help students enhance EM skills and mindsets. Many of the learning objectives are important for computer science students, and tie directly to accreditation outcomes. The software engineering course is used to measure performance indicators for program outcomes #2 and #5. Although not used directly for ABET assessment, this course also strongly contributes to outcomes #1 and #3 while contributing to a smaller extent to #4.

Computer Science ABET Student Outcomes:

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts.
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

This course modification is also part of a larger effort at the University of Portland to embed the entrepreneurial mindset across the curriculum (Farina et al., 2020; Dillon et al., 2020; Dvorak et al., 2020; Poor et al., 2020; Bieryla et al., 2020; Tribelhorn et al., 2021). Over the course of two years more than 30 different EM modules were tested in computer science, electrical engineering, mechanical engineering, and civil engineering classrooms. Many of the EM modules across the university used the same set of learning objectives for module design, allowing a larger sample for comparison with this class project.

2 Background

The entrepreneurial mindset (EM) has been studied by prior research teams. Some faculty have worked to build EM into each part of a curriculum, program (Blake Hylton et al., 2020), or university (Secundo et al., 2016; Ridley et al., 2017). Other groups have focused on understanding how students learn the entrepreneurial mindset and meta-cognition (Haynie et al., 2010; Bosman and Fernhaber, 2017). Antonaci et al. (Antonaci et al., 2015) used advanced computer science tools to develop games that support the development of the entrepreneurial mindset in students.

Embedding a module focused on entrepreneurial mindset within an existing course has been explored by others using different tools (Harichandran et al., 2018; Bosman and Fernhaber, 2017). A summary of prior work on entrepreneurial mindset in the computer science classroom is shown in Table 1. We have found that although a lot of prior work has focused on EM in engineering courses, there hasn't been as much focus in computer science specific courses. This is a concern as computer science students will be contributing to much of the future digital world and would be well served by this mindset.

Table 1. Summary of literature on entrepreneurial mindset modules in computer science classes.

Citation	Year	Course Type	Assessment	Methods and Topics
Bowers and Yerion (Bowers and Yerion, 2013)	2013	Intro to CS	Pre/Post Survey	Programming robots
Stanchev (Stanchev, 2015)	2015	Database Systems	Student Survey	Project-Based Learning
Estell et al. (Estell et al., 2016)	2016	Intro to Engineering	Student Survey	Software design
Tabrizi (Tabrizi, 2017a)	2017	Digital Systems	Student Survey	Jigsaw
Salas (Salas, 2017)	2017	Software Entrepreneurship	Student reflections	Software design
Tabrizi (Tabrizi, 2017b)	2017	Computer Architecture	Student Survey	Customer Model
Santiago and Guo (Santiago and Guo, 2018)	2018	Digital Communication Systems	Student Survey	EM report
Present Work	2021	Software Engineering	Student Survey	Problem-based learning, Jigsaw

Variants of agile development have been slowly integrated with traditional approaches into courses, most commonly in capstone courses and project management courses. (e.g., (Baird and Riggins, 2012; Landry and McDaniel, 2016; Laplante, 2006; Ramakrishnan, 2009)). However, it is rarely a focal point as most textbooks provide minimal content on Agile (May et al., 2016). There is a need to incorporate Agile into both the content and pedagogy of courses (Sharp and Lang, 2018). Agile has been combined with service-learning (Robinson and Hall, 2018), but there has not been a lot of work integrating Agile with active learning techniques. A specific implementation of agile that is inherently and especially active is Scrum. Scrum or a hybrid Scrum are the most common versions of applied Agile "methodologies" (agi, 2019). Scrum is used in some courses, but often not applied holistically and in the manner of industry (Jimenez and Cliburn, 2016; May et al., 2016; Robinson and Hall, 2018). The authors posit that the difficulty of replicating a complete set of agile practices in the form of a process such as Scrum limits the adoption of this pedagogical focus in curricula outside of software engineering programs. Many adaptions are needed to approximate these techniques in an undergraduate classroom setting. Recently, efforts have focused on making these adaptations for an undergraduate classroom setting (Tribelhorn and Nuxoll, 2021).

Scrum is a variant of agile software development that aims to be lightweight and subscribes to the standard agile practice of breaking up the work schedule into blocks, called sprints. These sprints are typically units of time ranging from one to four weeks. At the end of each sprint, the development team should be delivering some incremental value to the end customer (Cohn, 2005). Within Scrum, the sprint is managed with a number of key process-based events. Each sprint begins with a sprint planning meeting where the entire Scrum team generates tasks in various forms, often in the form of user stories (user-centered statements of requirements and acceptance criteria) which are placed in the product backlog. These are refined by the development team and then confirmed for the sprint and added to the sprint backlog. Each future work day within a sprint includes a short stand-up or daily scrum meeting where each team member addresses three issues: recent work, immediate work plan, impediments. These meetings are time-limited so issues are addressed by smaller groups and not the entire team. At the end of a sprint, the work is reviewed, often via demos, and finally a retrospective meeting is completed. In the retrospective, the team reflects on how to improve the application of the specific agile process. Scrum specifies three roles: product owner, Scrum master, development team. An important distinction between Scrum and other agile methodologies is that it has no particular software emphasis which allows for individual teams to specify specific engineering practices. This also makes it particularly suitable to academic settings. Table 2 summarizes the key components of the Scrum process.

Table 2. Summary of Scrum Components

Roles	Events	Artifacts
Product Owner	Sprint	Product Backlog
Scrum Master	Sprint Planning	Sprint Backlog
Development Team	Daily Scrum (standup)	
	Sprint Review	
	Sprint Retrospective	

The modules described in this paper extend the prior work in several ways. The modules have been scaffolded in a single course to provide nuanced skills for the students. The methods include overlapping learning objectives between multiple modules for a rich student exploration.

3 Methods

Our Software Engineering course is scheduled as a semester long (15 week) course as shown in Figure 1. There is a single semester long project that the students complete in groups of four to five students using a variant of the Agile development process. The large group size mirrors the larger teams found in industry and encourages development of communication skills. The course project has included various primary programming languages including Java, Python, and JavaScript. Past course projects include creating a web browser, an e-sports video game league client, creating web-based applications including: discovering and rating art installations in Hawaii, displaying comparative statistics about clean/dirty energy generation by city or region, visualizing various statistical information juxtaposing states.

Implementing an agile process has many variations and we use the active learning focused methods presented in (Tribelhorn and Nuxoll, 2021). Within each sprint, we take care to focus on stand-up meetings, process focused retrospectives, and sprint planning. In most offerings the assigned work has been setup with increasing requirements provided by the instructor in addition to students self-directing the final product based on a written requirements specification and successive sprint planning. These modules reinforce the learning outcomes discussed previously. In order to help shift the mindset of the students we have experimented with a number of interventions. Two modules presented were only assessed quantitatively within the context of the course long project (given their focus on requirements elicitation and software testing).

In this section we present our the individual interventions, in chronological order as visualized in Figure 1), for connecting entrepreneurial mindset, the assessment methods implemented, and finally course variations. The times are estimated for a course size of about 20-25 students. Additional information on materials and handouts can be accessed by request to the corresponding author.

3.1 Connecting Entrepreneurial Mindset

The first module, “Design and Data Visualization" is a framework for the course project and an application of Agile development into problem-based learning. It spans the entire semester and is used to inspire the students to focus on non-technical components of a software development project such as value creation. The other modules highlight the value creation aspects of software engineering. This entrepreneurial mindset motivates the students to think of the course project as more than an assignment but as a creative endeavor. The creation of these activities serves as a continuous reminder of the value that they will create in their own careers.

3.1.1 Assessed Module - Design and Data Visualization (Semester Project)

In this semester long project students work in teams to develop a software product over the course of a semester using techniques from Agile and SCRUM. The teams implement a complete software product, as well as develop a viable business plan for the product. The project culminates with an end of semester team presentation of the design, product, and the

management process implemented. Students share lessons learned including both successes and failures along the way.

Scenario: *You were hired by the Mayor as the new director of informatics for the City of Portland. The mayor tells you that she wants to improve the city functions using the data that they collect. She is frustrated because she can't get the city council to agree to some of the changes she wants to make because they don't understand the data. You and your team are given much of the data collected by various city agencies. The Mayor suggests that you should make a web-based application so that she can easily show the city council visuals of the data that you will be processing. She tells you that you'll get a bonus for every efficiency or insight that you identify.*

Sprint Information: Each sprint runs approximately two weeks and sprint planning is done during class on first day of the sprint (last day of prior sprint). Students take tasks (product features) from the backlog or requirements document and add to their current sprint board. These tend to be large and vague, so they have to work to define each task better and then assign a work amount and a person to complete the work. This is how the students apply systems thinking to complex problems and how they begin to assess and manage risk. Assessing each task, estimation is done: this is a group based *Think-Share Inquiry and Debate*: What does it take to build this feature? Which features have the most value? Students typically rate work as a number of hours to complete and can use their hands rock-paper-scissors style to reveal their best guess for each task before then discussing to agree on an amount.

At the end of each sprint each team presents a live software demo and reviews their testing code. They always must answer what value did you create? Following the demos, retrospectives are conducted within each team. Students work to improve the process by reflecting on what when well, what did not go well, and what to do differently next sprint.

3.1.2 Assessed Module - User-centric value creation

Students design use cases for a real world software application to make a project more profitable for a fictional employer. The introduction to uses cases lecture includes an informal activity where the students create use cases for a simple product, such as an ATM machine. This lecture period is used to focus on the less obvious details of working with use cases, such as non-functional requirements (NFRs), pre-conditions, alternate flows, and refactoring use cases. In later class periods the following scenario is presented.

Scenario: *You work for a well known conglomerate. Your product manager comes in and tells you that your department is losing funding to self-driving car software because your product is not exciting upper management. You won't be getting any bonuses or raises this year. Your manager thinks that you should create some new and exciting features for your software to get upper management to allocate more to your product. In twos or threes, brainstorm as many Use Cases as you can to extend your software. Do not create any use cases for basic functionality.*

10-15 min - The software here is the course project, but it could be an existing software product if students don't have an applicable project. Carefully separate pairs from their other team members to avoid early ideation convergence. A template for a Use Case is projected. The scaffolding here is to focus them on interesting functionality for version 2 of the software. There is a temptation to rehash basic functionality that they have already implemented.

20 min - Have pairs present to the other pair/triplet in their team and consider the following: The median software developer earns about $100,000/yr, which is about $50/hour. *How much does each use case cost to implement? How much value does the use case represent? (State your assumptions.) How to value differentiation or quality?*

20-30 min - Next each full team presents to the class and the whole class discusses the value of each use case in terms of both the cost to develop and the "value" (differentiation, profits, etc.). Are any of the use cases worth developing? In this discussion, it is helpful to force each team to give actual numbers even if they have no idea how to estimate. The instructor can offer suggestions of approximate numbers in the cases where they are struggling to come up with

reasonable estimates. This helps the students to focus on important details that impact design including user numbers, subscription fees/ad revenue, hosting and delivery costs, etc.

3.1.3 Module - Value Creation via Interviews

As most of our projects are student-driven we need to encourage the students to generate a wide and deep set of project requirements. To achieve this we have the students complete an interview activity about a theoretical business idea.

Scenario: *You and four friends are business majors and you have a great new idea for a mobile application (or other product) that will make you all filthy rich. Unfortunately, none of you know how to create an app. Luckily, you each have a friend in computer science (or discipline that relates to the product) who might be able to help. You plan to take this idea to your friend and have them build it for a small share of the profits. In your group, you will research and determine a single application that represents your idea. You may use existing applications or games as inspiration for your new venture's idea. Be sure that everyone in your group knows and agrees on the functionality of your proposed software.*

Process: Assign groups of N for an even number of teams, each group researches and defines a common application (game, banking app, etc.). This is assigned as homework. Starting in class, spend 10 min making sure this team agrees on the specification.

For each set of two teams pair up the members individually with a member of the opposite team. It's recommended that these pairs are separated in the classroom so that they cannot talk easily with other team members. Each pair will interview for requirements for each other's expert focus. (5 min/ea for 10 min). (Example: if teams are 4 people, each of those 4 will pair one on one with team members of the opposite team). We provide a "Generic Context-Free Interview" slide and handout.

Return to original groups of N, and generate a single list of requirements for the product of the opposite team. (10-15 min). Scaffolding: About 5 minutes in, remind students of the goal of the "business" students: to make big profits.

Expert groups grade each other on the accuracy of their proposed requirements AND the value they think it is worth. Each team presents their view of the product for 3-4 min (approx. 20 min total).

In order to help teams grade their development team you can ask them to answer (at least) the following questions to the rest of the class: *What is the value creation that the requirements gatherers generate? How much of the profits would the proposing team be willing to share with the developers?* Most students find that this activity improves their ability to elicit requirements and integrate information from multiple sources.

3.1.4 Module - Inspiring Software Testing

Unit testing is the number one engineering practice cited according to the 13th Annual State of Agile Report(agi, 2019) used by 69% of teams and Continuous Integration is third with 53% of teams reporting its use. We find that it helps to show students the value of testing to increase their commitment to these industry standard processes.

Hook: *Tell students about the state of phone technology in the early 1990s. Ask them to think about how would you reach someone? Most people had phones only in common rooms, i.e. kitchen and living room. How would you talk to friends outside of school? The internet didn't exist, so no online ordering of goods or pizza. You'd have to call in orders. How would you call 911 from the middle of a street? (Could instead have them do activities like order a pizza, but wait you don't have your cell phone or the internet, etc.)*

We use a day of class to read (for approx. 10 min) a short real case study of an incident where there was economic damage (& possibly loss of life) from a failure to run tests before deploying software.(Zubairi, 2002) Students are assigned away from their project teams to new teams to represent different groups 1) Legal/Public Relations for the company, 2) Developers, 3)

Customers/Users, 4) Government. They are asked to prepare a statement that might address one or more of the following: lessons learned, responsibility, actions, ethical concerns, impacts. These groups then present informally this statement or set of statements and the instructor is sure to bring up in discussion issues including: who was hurt, litigation, liability, job security, how it happened, what it means to be a whistleblower, governmental regulation or policy interventions. After all teams have presented they return to their project teams to discuss and decide the likely outcomes in terms of fault, liability, costs, etc. based on what each group presented. Students can present this to the class or just present to the instructor.

Finally the instructor leads a whole class discussion of the value of testing, and possible ethical issues related to their course project. This can focus on the tradeoffs in terms of costs and risk. This activity can be compressed into a single 85-minute period, or two 55-minute periods. After completing this activity, we have found that students are more motivated to actually write unit tests and develop acceptance criteria.

3.1.5 Assessed Module - Software quality

This project examines software in terms of how well it is designed. Students work in expert groups to evaluate software quality attributes with respect to value creation using a modification of the powerful active learning Jigsaw technique (Aronson et al., 1978). This set of activities takes at least two class periods to complete.

5 min - This begins after a lecture and discussion of software security. Each student starts with ranking an instructor provided list of software programs based on their perception of its security. Any quality attribute or non-functional requirement (NFR) could be ranked; security is a good choice due to its importance and nebulous nature.

5 min - Combine students into pairs and have them re-rank the list. Pairing students in twos and threes within their project team works well.

10-15 min - Finally, the whole project team re-ranks the software list. The students write their ranked list on the board upon completion. Larger groups can really stretch this time with great discussions.

2-5 min - Instructor leads a brief discussion of how different groups perceived security, highlighting differences between groups. This leads to the introduction of valuation of a NFR.

20+ min - Students set their lists aside and each project group assigns each team member to a separate expert group. Each expert group is assigned to a different NFR such as: usability, reliability, availability, or performance. Expert teams must determine how to value that NFR. The output of each group are written criteria. This requires some guidance, after a few minutes hints about desiring criteria specified in a numeric form and weighted criteria can be provided. This can be finished as a homework assignment.

45 min - Students return to their project teams and try to value two specific software programs (assigned by instructor) from the list with respect to all of the expert group NFRs using the criteria from each expert group and generating at least some informal criteria for security. Can they put a monetary value on it?

Students struggle to develop numeric outputs for each of the topics. The authors find that using lots of scaffolding to slowly move them towards actual numbers is required. Optionally a handout with hints about user numbers, subscription fees, ad revenue, etc. could be provided. After about 20-30 min, the instructor will need to request most groups revise their numbers because they “assume" that each NFR is equally important. Finally, encourage the students to commit to a real number for both pieces of software. How much of a subscription fee is attributable to its software quality? How many users are attributable to quality?

15+ min - Each project group presents their estimated value for the two pieces of software.

10 min - Final discussion: How much of software value is from NFRs? How much value have they added in NFRs to their course project? An optional writing assignment can be added for

them to reconsider their own course project. The authors have the students edit their README file with a ranked list of the project's top NFRs achieved.

3.2 Assessment Methods

The overall results of this research involved two parts, an external observation and an optional student survey (IRB approved). The survey asked students to self-assess the specific learning outcomes on a five point Likert scale and write an open-ended response about each learning objective. The survey questions included the five learning objectives that mapped to the larger university project, allowing us to compare to other courses. The complete survey is included as an Appendix.

An example of the format for this survey question is shown in Figure 2. As these course projects are the primary output showcasing a range of student skills (design, reflection, teamwork, implementation, etc.), exam scores are not reflective of student work or knowledge gain. The authors present these results having seen a impressive boost in the quality of student projects from implementing the modules described.

1. For the learning objective listed below, please rank the extent to which your capacity increased **during this class**. (circle one) Then, please elaborate with a specific example.

Question	Not at all	Very Little	To a Small Extent	To a Moderate Extent	To a Great Extent
To what extent has your ability to **design, build, and test real world engineering systems** increased during this class?	1	2	3	4	5
Describe a specific example of how your ability to **design, build, and test real world engineering systems** increased in this class.					

Figure 2. Example of student survey questions where students were asked to consider each learning objective on a Likert scale.

After the students completed the survey, the results were combined for two sections of the class, including two different instructors. The essay results were analyzed using a natural language processor in the R statistical program (CRAN, 2019) and using standard qualitative methods. We used the standard statistical tests in R to perform a t-test between the survey data from this class and the larger collection of student responses across the university.

We used the text mining framework documented in more detail by Feinerer et al. (Feinerer et al., 2008). We converted the text to lower case, removed special characters and numbers, and we removed the white-space. After the text was cleaned the software performed a text frequency analysis to determine the most common terms in the student comments. The results were displayed as a word cloud with the size of the word representing the frequency of occurrence in Figure 3. The larger the word the more occurrences that appeared. Qualitatively, it is clear that the students found value in the content and there is an alignment to Bloom's Taxonomy of some higher level skills from words like *design, testing, decisions, able*.

3.3 Variations

Many variations are possible with the structure of this course. Below is a brief summary of some that we have used successfully.

3.3.1 International Collaboration

In the Spring 2019 semester, we collaborated with Kaunas Technological University (KTU) in Kaunas, Lithuania. The student team was larger: consisting of six students from one university and five from the other. Both universities used the same high-level structure for their courses though individual assignments were different. This approach presented some predictable pros and cons.

Figure 3. Student natural language summary of the open-ended essay questions.

- Pro: Students learned about the difficulties associated with remote collaboration
- Pro: Students gained exposure to another culture
- Con: Communication and collaboration were much more difficult both for cultural and logistical reasons. The different time zones were particularly problematic.
- Con: The difference in semester schedules meant that one university had to begin work a full three weeks before the other which, subsequently, continued for four weeks after the first. The universities also had mid-semester breaks at different times. The net result was that collaboration could only occur during half the 15-week semester.

Many technology companies use distributed teams, and with recent changes in work from home policies, this real-world experience is key for students to appreciate the value of development processes and tools. The students that participated were too few to be surveyed separately, but the authors found that they brought a mindset shift with them into future endeavors and that the experience helped them inspire similar attitudes in their peers.

3.3.2 Class-Wide Project

One of our most unique contributions was to experiment with large groups as most reported agile project based learning courses tend to utilize small teams(Salza et al., 2019; Lutz et al., 2014). In the Fall 2019, Spring 2020, and Fall 2020 semesters, we experimented with assigning a single project to the entire class. Additional roles were assigned to students in order to better approximate an actual software company: a CEO (the instructor), CTO (student elected), Scrum masters, software engineers and quality engineers. These roles were rotated on a sprint-by-sprint basis. This makes a number of differences compared to smaller teams, mostly leading to a better immersive experience:

- When the team is a least 15 students, communication/coordination suddenly gets a lot harder. The value of tools like Trello boards and stand up meetings become much more apparent. We require the student teams to meet once per week outside of class but we find that they often voluntarily meet more frequently. Certainly the leadership team (CTO and team leads) meets even though it is not required.
- Students get taste of being a mid-level manager (the “CTO" role). They are elected to this position by their peers and not allowed to write any code while they are in the role. Even so, they quickly realize that it’s a lot of work. This is much more of a true leadership position than just a team lead. The CTO selects sprint goals and team composition. “I was class CTO" looks good on a resume too.

- Progress is faster. Having one team of 20 isn't four-times faster than one team of five would be because of the social/learning overhead. Still, they make progress about twice as fast. So, about twice as much functionality is implemented in a semester.
- Since there is a different CTO every sprint who must shuffle the teams, they effectively get to experience multiple re-orgs.
- The team changes mean they spend a lot of time eating their own dog food (e.g., adding features and fixing bugs in code they didn't write). So, the value of comments and good code design becomes much more apparent. The value of code reviews becomes very apparent too.
- It's much easier to find a real customer for the team since the instructor only need to find one (see next section). That customer is often particularly engaged and, due to Scrum, comes to class often (about once every 2-3 weeks).

This approach created more realism, and a real entrepreneurial focus, at the expense of adding additional communication and organizational hurdles for the students. In both cases, the instructor anecdotally observed a much stronger sense of camaraderie and enthusiasm for the project. This class-wide approach allows the instructors to stay truer to Scrum within a classroom setting as all of the roles can be filled especially when we recruit a real world client.

3.3.3 Real-World Client

We have also experimented with introducing a real-world client in the Fall 2019, Spring 2020, Fall 2020, and Spring 2021 semesters. Our university has an entrepreneurial certificate program wherein students learn to conceive, assess, create a business plan for and pitch an idea. By partnering with this program, we were able to recruit students whose business ideas require a web application. In the Scrum fashion, this client becomes part of the team.

This approach gave the students an opportunity to understand how a software product fits in a business context. Students got more exposure to the reason for an agile software development model: the desired product requirements change as the project progresses. These customers change their minds and change is a critical thing to cope with. The value of agile methodology is clearer in this context. The students had to identify, negotiate and, ultimately, adjust their design for changes in client needs. Finally, the fact that the product had a real user in mind meant that students were less inclined to compromise on the quality of their work.

Instructors who try this variant should be mindful of intellectual property. In the Spring 2020, we introduced a legal agreement where the students agreed to assign their intellectual property to the customer. However, this increases the students' awareness of value creation, which is a key component of the entrepreneurial mindset.

4 Results and Discussion

The results for the interventions were successful and suggest a few ways to improve the modules over time.

4.1 Module Results

Qualitatively, we found that the student work completed in these modules to be higher quality than similar work submitted in prior years. Exam scores were improved when measuring students ability to create use cases, especially clarity and completeness. This qualitative improvement was also noticed by the instructors of our senior capstone course. The module on quality attributes noticeably increased student commitment at the end of the project attributable to the perspective it provides the students. Projects from the course most recently have included web-apps for discovering and rating art installations in Hawaii, displaying comparative statistics about clean/dirty energy generation by city or region, visualizing various statistical information juxtaposing states, and an e-sports video game league client. The course long project module was particularly helpful in getting the students to generate the requirements and milestones

for their projects with reduced guidance from the instructors. Finally, in terms of assessment, we believe that these modules greatly assisted in achieving the highest level of outcome in Bloom's taxonomy, "create," for the ABET outcome #2: "Design, implement, and evaluate a computing-based solution[...]" as the process of truly evaluating software outputs is the most difficult component implement in an educational setting.

4.2 Survey Results

The student survey was completed by 33 students over two sections, representing 75 percent of the possible respondents. Most of the respondents were computer science majors and four were double majors or CS minors. The summary of the results is shown in Table 3 and Figures 3-4.

The small samples size resulted in no statistically significant comparison based on a t-test. However the same mindset objectives were assessed in other course modules as part of the larger curriculum wide effort. These averages are also shown in Table 3, indicating that the modules in this course outperformed other modules in the curriculum for the mindset objectives except "Identify and evaluate sources of information."

Overall, the university sample for all learning objectives tied to entrepreneurial mindset had a mean of 3.58 and standard deviation of 1.07 from over 2,526 student responses. This confirms that the rated the learning objectives in this module higher than most other objectives we studied. The statistical analysis of all the university modules and objectives did find a statistically significant difference in the instructors, which may indicate some student bias in the survey, or a different level of instructor proficiency in implementations of the modules.

Table 3. Summary of survey results for each learning objective in the course, n=33 students. On the right, the mean and standard deviation for the curriculum wide effort is shown.

Learning Objective	Course Modules		Full University Sample		
	Mean (n=33)	**Standard Deviation**	Mean	Standard Deviation	Sample Size (n)
Understand ramifications of design decisions	4.06	0.86	3.63	1.05	136
Identify and evaluate sources of information	3.16	1.39	3.38	1.12	88
Self-reflect on your own ideas	3.50	1.08	3.37	1.19	126
Design, build, and test real world engineering systems	4.09	0.68	3.52	1.00	58
Identify links between course knowledge and real world systems	4.00	0.94	4.04	0.79	416

The student essay portion of the survey was analyzed using a natural text processing algorithm as shown in Figure 3, and qualitatively as summarized below.

Learning Objective 1: Understand ramifications of design decisions

- Students described how the KEEN module helped them understand considerations in design decisions prior to beginning a project (65%; n = 21 students).

  > "This class makes you think a lot more about how to do things and why before diving into something that may be a bad idea."

- Students described how the KEEN module helped them understand ramifications in design decisions (59%; n = 19 students).

  > "In our project we've had to make design choices, some have been bad. For example, we decided to structure our database a particular way and ended up having to restructure. This leads to knowledge about choices in the future."
  > "I got to see firsthand the good and bad outcomes of design decisions in my project."

- Students reported how the KEEN module helped them understand that their own decisions can have a serious impact (53%; n = 17 students).

> "I understand better how an application should work and how design really affects how other people view it and how that affects their usage of it."
> "I was already afraid of making bad decisions in the workforce but now I know people can actually die as a result."

Learning Objective 2: Identify and evaluate sources of information

- Students reported that the KEEN module helped them identify resources (47%; n = 15 students).

> "I quickly learned how to identify information that would be useful."

- Students described how the KEEN module helped them evaluate sources of information (38%; n = 12 students).

> "A lot of the actual code learning in this class was based on researching for ourselves. I've learned to better recognize the more reliable sources."
> "Trying to solve problems led me to having a better idea of what sources are good are what are not."

- Students described how the KEEN module helped them learn independently (41%; n = 13 students).

> "Since a lot of the coding in the class was self-taught it forced me to go out and find the information myself which is very important to me and I found very valuable."

Learning Objective 3: Ability to self-reflect on your own ideas

- Students reported that the KEEN module helped them self-reflect on their own ideas (63%; n = 20 students).

> "I was able to see how crucial my input is."
> "I feel I had to go back and re-do things I did as a result of necessary changes, which made me reflect on the quality of what I did originally."

- Students described how the KEEN module helped them develop and create new ideas (34%; n = 11 students).

> "This course has helped me gain confidence in my abilities. As a result, it's helped me feel more open to idea creation because these ideas seem more reachable now."

- Students described how the KEEN module helped them learn the value in critiquing the ideas of their peers (56%; n = 18 students)

> "Being judges and having peers rely on you is something I think is accurate to the real work team and having that experience does a great job in forcing you to reflect on yourself."

Learning Objective 4: Design, build, and test real world engineering systems

- Students described how the KEEN module helped them understand the whole process of designing, building and testing engineering systems (50%; n = 12 students).

> "I learned a lot of what it takes to engineer a whole system, different methods, teamwork, etc."
> "We built a website and I had absolutely no clue how to do that before. Now I can build one, test one, and improve one along with being able to work with a team."

- Students described how the KEEN module helped them specifically understand the importance of testing engineering systems (33%; n = 8 students).

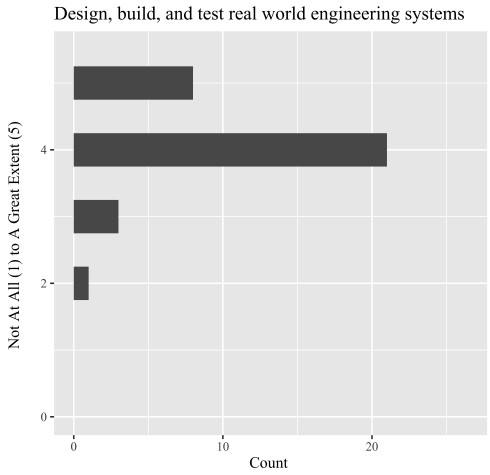

Figure 4. Student responses to the learning objective, “Design, build, and test real world engineering systems."

> “My increased knowledge in the testing filed makes me better understand the importance of reliability and solidly structured code.”
> “Now I know how important testing is to the whole process.”

- Students reported that the KEEN module helped make a connection to real world engineering systems (29%; n = 7 students).

> “I now know how to create a web application. Before this class most of the stuff I knew was not very important to real world stuff.”

Learning Objective 5: Identify links between course knowledge and real world systems

- Students described how the KEEN module helped make connections between the content and the real world (38%; n = 12).

> “This course really incorporated real world situations and was one of the first classes to put me in a ’work’ type setting.”
> “I saw a lot of connection between the types of tasks we learned about in class and real-world testing.”

5 Conclusion

A new project was developed for computer science students to help them develop specific skills to support an entrepreneurial mindset. Multiple new modules were incorporated in an existing course. The modules were well received by students and the projects developed demonstrated the students commitment to creating value in addition to improved technical outcomes. The qualitative improvement in the course projects completed by the students contrasted with prior years also supports the improvement in design thinking described in the results section. Based on these results, the authors recommend that other instructors incorporate the entrepreneurial mindset into their projects, possibly by adapting these modules to their existing curricular structure. Capstone projects are a particular opportunity, since many institutions have businesses act as a client for these projects. This hasn’t been widely studied.

In the spring of 2019, students in a junior level computer science course were surveyed to self-assess learning objectives. Students reported that the module had been successful for most of the learning objectives, particularly for designing, building, and testing real world systems. This emphasis on real world systems encourages reflection on entrepreneurship and ultimately helps with a shift in mindset from a narrow coding focus to a broader, value-focused mentality. This is critical to student preparation given the ease of creating modern software as successful

software is set apart when it provides excellent value to the end user. The authors believe that this work provides a novel focus to CS education research by focusing on mindset to achieve technical outcomes.

Although achieving mindset shift is a challenging endeavor, the qualitative results show that a number of curricula choices can help achieve this learning outcome. Although, too small to separately report quantitatively, the students that worked on the international project were particularly positively impacted by the experience and reported an appreciation for the value of their efforts and creation. The authors found implementing this variant challenging logistically, but a key area for future efforts given its outsize return on mindset.

This work presented a variety of curricular interventions which are highly flexible and applicable at other institutions. In addition to the possible course variations presented, we found that the choice of the primary programming language(s) has not impacted the learning outcomes; although when introducing a new languages the initial progress towards the stated project goals is slower. Ultimately, this course achieves its learning outcomes as an ABET benchmark course for for program outcomes #2 and #5, and students respond positively to this real world experience. The immersive quality of both an agile process and a plethora of active learning modules gives the students a high level of an industry-like experience. Students find that their initial software designs are not well thought out, yet using the agile process enables them to improve iteratively unlike traditional course projects. We believe that the course project described, itself an application of the agile process, counteracts the common waterfall paradigm used by students due to the limited scope of most undergraduate course assignments.

Future work will include adjusting small elements of the project to further improve student ability to learn independently and integrate information from multiple sources. Other researchers may find that studying other areas of mindset shift can use these techniques. More broadly, computer science education is still ripe with opportunity for additional studies of other methods to prepare students. The interventions described could also be implemented at other institutions to validate that they do encourage an entrepreneurial mindset. The authors also think that as the technology industry changes, revisiting the classroom implementation of Agile methodologies will be important to keep students engaged with EM.

Acknowledgments

Funding for this work was provided by the Kern Entrepreneurial Engineering Network (KEEN) as part of a grant to the University of Portland and by the Donald P. Shiley School of Engineering. Special thanks to Jeff Welch and Rebecca Levison for data support. The authors would like to thank Dr. Tomas Blazauskas (KTU) for support of the international development project and Ignas Paplauskas of Devbridge for working as a client and mentor of the international project.

References

KEEN, “KEEN - The Framework.” [Online]. Available: https://engineeringunleashed.com/mindset-matters/framework.aspx

L. Bosman, S. Fernhaber, and S. O. service), *Teaching the entrepreneurial mindset to engineers*. Springer, 2018.

“13th Annual State Of Agile Report.” 2019. [Online]. Available: https://www.stateofagile.com/

J. Farina, H. E. Dillon, R. D. Levison, and N. Ralston, “Increasing Student Curiosity with Cooling Systems,” in *American Society for Engineering Education Annual Conference*, Montreal, Canada, 2020.

H. E. Dillon, J. M. Welch, N. Ralston, and R. D. Levison, “Creating an Engineering Action Plan for Ethics,” in *American Society for Engineering Education Annual Conference*, Montreal, Canada, 2020.

R. Dvorak, H. E. Dillon, N. Ralston, and J. M. Welch, “Exploring Ethical Hacking from Multiple Viewpoints,” in *Education Annual Conference*, Montreal, Canada, 2020.

C. J. Poor, H. E. Dillon, J. M. Welch, and N. C. Ralston, “Implementation of real-world class activities in an Introduction to Environmental Engineering Class,” in *American Society for Engineering Education Annual Conference*, 2020.

K. Bieryla, N. A. Schulz, R. D. Levison, and H. E. Dillon, “Play-Doh and pendulums : making mass moment of inertia fun Play-Doh and pendulums : making mass moment of inertia fun,” in *American Society for Engineering Education Annual Conference*, 2020.

B. Tribelhorn, H. E. Dillon, A. Nuxoll, and N. Ralston, “Connecting Entrepreneurial Mindset to Software Development,” in *American Society for Engineering Education Annual Conference and Exposition*, Virtual Conference, USA, 2021.

J. Blake Hylton, D. Mikesell, J.-D. Yoder, and H. LeBlanc, “Working to Instill the Entrepreneurial Mindset Across the Curriculum,” *Entrepreneurship Education and Pedagogy*, vol. 3, no. 1, pp. 86–106, jan 2020. [Online]. Available: http://journals.sagepub.com/doi/10.1177/2515127419870266

G. Secundo, V. Ndou, and P. Del Vecchio, “Challenges for Instilling Entrepreneurial Mindset in Scientists and Engineers: What Works in European Universities?” *International Journal of Innovation and Technology Management*, vol. 13, no. 5, oct 2016.

D. Ridley, B. Davis, and I. Korovyakovskaya, “Entrepreneurial Mindset and the University Curriculum,” *Journal of Higher Education Theory and Practice*, vol. 17, no. 2, apr 2017. [Online]. Available: https://articlegateway.com/index.php/JHETP/article/view/1569

J. M. Haynie, D. Shepherd, E. Mosakowski, and P. C. Earley, “A situated metacognitive model of the entrepreneurial mindset,” *Journal of Business Venturing*, vol. 25, no. 2, pp. 217–229, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jbusvent.2008.10.001

L. Bosman and S. Fernhaber, *Teaching the entrepreneurial mindset to engineers*. Springer International Publishing, aug 2017.

A. Antonaci, F. M. Dagnino, M. Ott, F. Bellotti, R. Berta, A. De Gloria, E. Lavagnino, M. Romero, M. Usart, and I. Mayer, “A gamified collaborative course in entrepreneurship: Focus on objectives and tools,” *Computers in Human Behavior*, vol. 51, pp. 1276–1283, oct 2015.

R. S. Harichandran, N. O. Erdil, M.-I. Carnasciali, J. Nocito-Gobel, and C. Li, “Developing an Entrepreneurial Mindset in Engineering Students Using Integrated E-Learning Modules.” *Advances in Engineering Education*, vol. 7, no. 1, 2018.

S. Bowers and K. Yerion, “Programming Personal Robots within an Introductory Computer Science Course for Engineering Majors,” *J. Comput. Sci. Coll.*, vol. 29, no. 1, pp. 133–139, oct 2013.

P. L. Stanchev, “Teaching the entrepreneurial Mindset in Database Class,” *Computer Science and Education in Computer Science*, vol. 11, no. 1, pp. 85–93, 2015.

J. K. Estell, D. Reeping, and H. Sapp, “Curiosity, connection, creating value: Improving service learning by applying the entrepreneurial mindset,” *ASEE Annual Conference and Exposition, Conference Proceedings*, vol. 2016-June, 2016.

N. Tabrizi, “Fostering an entrepreneurial mindset in "Digital systems" class through a jigsaw-puzzle model,” in *Proceedings - Frontiers in Education Conference, FIE*, vol. 2017-Octob. Institute of Electrical and Electronics Engineers Inc., dec 2017, pp. 1–8.

R. P. Salas, “Teaching entrepreneurship in computer science: Lessons learned,” in *Proceedings - Frontiers in Education Conference, FIE*, vol. 2017-Octob. Institute of Electrical and Electronics Engineers Inc., dec 2017, pp. 1–7.

N. Tabrizi, “Fostering an entrepreneurial mindset in ’computer architecture and organization’ class through a producer-customer model,” in *IEEE Global Engineering Education Conference, EDUCON*. IEEE Computer Society, jun 2017, pp. 1843–1850.

J. M. Santiago and J. Guo, “Developing an entrepreneurial mindset using the KEEN framework for a digital communication system course,” *ASEE Annual Conference and Exposition, Conference Proceedings*, vol. 2018-June, 2018.

A. Baird and F. J. Riggins, “Planning and sprinting: Use of a hybrid project management methodology within a cis capstone course,” *Journal of Information Systems Education*, vol. 23, no. 3, pp. 243–257, 2012.

J. P. Landry and R. McDaniel, “Agile preparation within a traditional project management course,” *Information Systems Education Journal*, vol. 14, no. 6, pp. 27–33, 2016.

P. A. Laplante, “An agile, graduate, software studio course,” *IEEE Transactions on Education*, vol. 49, no. 4, pp. 417–419, 2006.

S. Ramakrishnan, “Innovation and scaling up agile software engineering projects.” *Issues in Informing Science and Information Technology*, vol. 6, pp. 557–575, 2009.

J. May, J. York, , and D. Lending, “Play ball: Bringing scrum into the classroom,” *Journal of Information Systems Education*, vol. 27, no. 2, pp. 87–92, 2016.

J. H. Sharp and G. Lang, “Agile in teaching and learning: Conceptual framework and research agenda,” *Journal of Information Systems Education*, vol. 29, no. 2, pp. 45–52, 2018.

S. Robinson and M. Hall, “Combining agile software development and service-learning: A case study in experiential is education,” *In SIGCSE ‘18: 49th ACM Technical Symposium on Computer Science Education*, 2018.

O. Jimenez and D. Cliburn, “Scrum in the undergraduate computer science curriculum,” *J. Comput. Sci.*, vol. Coll. 31, no. 4, pp. 108–114, 2016.

B. Tribelhorn and A. Nuxoll, “Using Agile and Active Learning in Software Development Curriculum,” in *American Society for Engineering Education Annual Conference and Exposition*, Virtual Conference, USA, 2021.

M. Cohn, *Agile estimating and planning*, 1st ed. Prentice Hall, 2005.

J. A. Zubairi, “To Test or Not to Test: A Case Study on Ethics in Computing ,” 2002.

E. Aronson, N. Blaney, C. Stephin, J. Sikes, and M. Snapp, *The Jigsaw Classroom.* Beverly Hills, CA: Sage Publishing Company., 1978.

CRAN, “R Software,” 2019. [Online]. Available: https://cran.r-project.org/

I. Feinerer, K. Hornik, and D. Meyer, “Text mining infrastructure in R,” *Journal of Statistical Software*, vol. 25, no. 5, pp. 1–54, 2008.

P. Salza, P. Musmarra, and F. Ferrucci, “Agile Methodologies in Education: A Review,” in *Agile and Lean Concepts for Teaching and Learning*, 1st ed., K. M. David Parsons, Ed. Springer, Singapore, 2019.

M. J. Lutz, J. F. Naveda, and J. R. Vallino, “Undergraduate software engineering,” *Commun. ACM*, vol. 57, no. 8, p. 52–58, Aug. 2014.