Tech Startup Learning Activities: A Formative Evaluation

Kevin Buffardi
Dept. of Computer Science
California State University, Chico
Chico, CA, USA 95926-0410
kbuffardi@csuchico.edu

ABSTRACT

The Tech Startup model is an approach where students in Software Engineering and Entrepreneurship courses form interdisciplinary teams to create businesses based on software products. The model combines Agile software development with compatible practices from Lean Startup to foster collaboration on real technology startup businesses (tech startups). This paper introduces learning activities for use in the Tech Startup model intended to improve adherence to Agile and Lean Startup methodologies.

This study describes a formative evaluation of the learning activities used for: project ideation, project planning, iterative delivery & feedback, and teamwork assessment. The study synthesizes responses to a questionnaire with feedback from three focus groups at the conclusion of an academic term. Based on our findings and observations, we report suggestions for encouraging innovative project ideas from Software Engineering students as well as approaches to holding students accountable for adhering to Agile practices.

CCS CONCEPTS

• **Social and professional topics~Software engineering education** • Social and professional topics~Computing and business • Software and its engineering~Agile software development

KEYWORDS

Entrepreneurship, Agile, Lean Startup, Slicing Pie

1 INTRODUCTION

In 2013, the American Society for Engineering Education (ASEE) gathered leaders in industry, government, and academia to identify and prioritize Knowledge, Skills, and Abilities (KSAs) for transforming the future of engineering education.

SEEM'18, May 27-June 3 2018, Gothenburg, Sweden
ACM ISBN 978-1-4503-5750-0/18/05…$15.00 https://doi.org/10.1145/3194779.3194781

Entrepreneurship was among the fifteen KSAs deemed highest priority, which they described as an: "aggregated trait of several other KSAs: critical thinking, business/economics acumen, and the ability to take risks. It builds […] by expanding on business and economics acumen and enabling students to learn more than economic capitalization, but also the process of starting a business from an idea" [1].

Simultaneously, there is an emerging trend of "millennipreneurs" (a portmanteau of *millennial* and *entrepreneur*) who are starting their first businesses in their twenties, while their counterparts in previous generations launched their first businesses around 35 years old, on average. Social media and eCommerce are both among the top three industries in which millennipreneurs launch businesses [4], following the lead of notable figures like Facebook's co-founder, Mark Zuckerberg [19].

Meanwhile, project-based Software Engineering courses face a common dilemma: when students come up with their own project ideas (or "toy projects"), their experience building the software usually does not resemble professional software engineering. Toy projects are particularly unrealistic because without external pressures for holding the software's value and quality accountable, "Students know their code matters only as much as they might find our assignments interesting, or as much as it counts toward their grades" [13]. Consequently, there is an opportunity to address the challenge of incorporating entrepreneurship into software engineering curricula while also creating more realistic projects by collaborating with Business or Entrepreneurship programs.

In the Fall of 2016, we piloted a cross-disciplinary collaboration that connected entrepreneurship students (from the College of Business) with development teams from the Computer Science department's undergraduate Software Engineering course. While students in Software Engineering learned Agile software development methods and related skills, their partners in the Entrepreneurship program learned compatible business methods as the teams joined efforts in creating technology business startups (tech startups). We proposed the learning framework as the *Tech Startup model* for software engineering education [7].

This paper provides an overview of the Tech Startup model while focusing on lessons learned from specific educational interventions we employed during the Fall of 2017. In previous work [23], we identified early indications of the model's strengths and suggested approaches for overcoming obstacles, such as managing equity and intellectual property. However, we

also previously found that the teams often strayed from Agile principles when working on their projects outside of the classroom [8]. Likewise, we wanted to encourage more software engineers to propose project ideas since we observed most proposals came from entrepreneurship students.

In this paper, we propose active learning approaches to address these shortcomings found in the first two semesters of the Tech Startup model. During the third semester, we introduced a joint-class activity for project ideation with the hope to generate more proposals from software engineers. In addition, we investigated how other activities impacted students' behaviors and adherence to Agile methods.

The additional activities included periodic "show-and-tell" sessions and an end-of-semester showcase to hold teams accountable for delivering working software in short iterations. In addition, the class incorporated student peer reviews with feedback to address concerns about individuals within teams. Finally, the classes placed an emphasis on the expectation that developers and business people meet face-to-face regularly. This paper describes the implementation of each intervention and uses qualitative analysis to provide insights into their effects. As a result, this experience report includes guidance on how to improve implementation of the Tech Startup model.

2 BACKGROUND

Students are sometimes highly motivated to work on a project when they are afforded the opportunity to come up with the project idea themselves. However, such open-ended assignments can be harmful [13] to software engineering education when they lack external pressures and accountability beyond their class grade. Nurkkala and Brandle identified six common gaps between academic and real industry software engineering projects, including: the lack of a real product, relatively short duration, high personnel turnover, low sophistication of software, no software maintenance, and no customer. They recognized that students require visceral motivation from an external pressure with "skin in the game" and that an instructor simply standing-in as a mock client is insufficient [16].

Beyond potential legal and university policy complications, clients are often hard to come by when they have a real business need for software but understand that students usually have less time and less experience to offer than their professional counterparts. Alternatively, some educators have turned to involving students in Free and Open Source Software (FOSS) projects [2]. Contributing to FOSS has unique advantages since it usually has real users and involves existing software in need of maintenance.

However, the FOSS model inherently depends on remote collaboration largely comprised of developers who volunteer their talents. To the contrary, the Agile Manifesto emphasizes principles incompatible with these qualities, namely: "Business people and developers must work together daily throughout the project" and "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation" [3].

In previous work, we attempted to reconcile FOSS with Agile principles by establishing a FOSS consortium in the local community that allowed Software Engineering students to meet face-to-face and work with software professionals and business people while collaborating on open source projects [5]. While we found many advantages to the localized FOSS (LFOSS) approach, it may not necessarily be easy to replicate in all communities and it is difficult to scale to support 30+ students every semester [6].

However, with the current prevalence of web, mobile, and internet-of-things applications, there is no shortage of people with ideas for software products who are in need of developers to create them. While Software Engineering courses are also in need of business-minded stakeholders for projects, students should not necessarily be doled out as free labor. As a result, we formulated the Tech Startup model so that entrepreneurs and software engineers could mutually benefit from each other's efforts while working toward creating a real business based on a software product.

2.1 Tech Startup Model

Students participating in the Tech Startup model form multi-disciplinary teams of entrepreneurs and software engineers. At the beginning of the semester, students propose their ideas for software products and form teams. As the projects commence, students in the Software Engineering course are taught the philosophies behind Agile software development as well as specific practices utilized by the Scrum [24] framework. These methods diverge from the processes students may have grown accustomed to when working on previous programming assignments. Unlike traditional one-to-two week programming assignments with rigid requirements and no maintenance after delivery, following Agile focuses the team on short iterations of developing, receiving feedback, and adapting.

Similarly, students in the Entrepreneurship course learn Lean Startup practices. Lean Startup [22] is based on a philosophy similar to Agile. Lean Startups mitigate the risk of faulty assumptions leading to wasted resources by delivering Minimum Viable Products (MVPs). MVPs are determined by what provides the most value to gaining insights about customers that require the least amount of effort [21]. Accordingly, both Agile and Lean Startup promote iterations of delivery, discovery, and adaptation that embrace a "fail fast" mentality that allows projects to accommodate evolving requirements.

Entrepreneurship students concentrate their efforts on testing business hypotheses and discovering insights into customers and the market. Meanwhile, Software Engineering students concentrate on designing, developing, and testing software that fulfills the customers' needs. The cross-disciplinary interaction provides students with unique opportunities to develop required business acumen [1] and soft skills [18]. However, care must be taken with teams working together on entrepreneurial projects since there is risk of forming inadvertent partnerships with murky legal ramifications [12].

Collaboration between the two disciplines complements each other's unique skill sets. While many students have entrepreneurial ideas and aspirations, studies have observed that

a lack of funding discourages many in the 'Millennial' generation from pursuing risky self-employment [11]. Accordingly, student entrepreneurs usually lack funding to hire developers; software engineers are high in demand and can consequently demand high pay. For these reasons, we would also consider it unfair to expect software engineering students to work for free to benefit someone else's business.

To reconcile the dilemma, we employ the Slicing Pie [14] approach to establishing a dynamic equity agreement. Slicing Pie creates a flexible relationship where the team agrees to pre-defined values for different contributions (such as implementing a certain feature or spending an amount of time on a particular activity) along with rules for how to retain "slices of pie" that have already been earned when there is personnel turnover later on (which is often the case with student-run projects, in general) [20]. Before beginning any of the project work, teams write and sign a Slicing Pie agreement. While the efforts put into the project do not immediately reap financial payback, this approach empowers students to earn a percentage of future business revenue to compensate them for their contributions.

2.1 Preliminary Findings

We adopted the Tech Startup model starting in Fall 2016, centered on collaboration between undergraduates in the Computer Science department's *Software Engineering* class and those in the Entrepreneurship program's *Web-Based Entrepreneurship* course. We scheduled the two courses to coincide so they could occasionally hold joint meetings. Although, the majority of class meetings are held separately so each class could concentrate on their respective topics. Details of the Entrepreneurship course are available in our previous publications [7, 23].

The Software Engineering course introduces students to Agile software development and particularly to the Scrum framework. Accordingly, the course has twice-a-week lab periods that are dedicated to students practicing Scrum, beginning with daily (or in our case, twice-weekly) standup meetings [24]. The course also introduces relevant skills and topics including: collaborative version control, unit testing, development operations/continuous integration, software design patterns, and metrics for evaluating software quality.

The grade for the course includes 75% weight for the team project. Projects are evaluated for their product's usefulness, design, and verification, including the team's ability to demonstrate accurate self-assessment of those qualities by applying the concepts and techniques covered in class. For example, teams are expected not only to test their software but also to demonstrate (using tools and metrics like code coverage) how thoroughly they have tested.

For the most part, we found the Tech Startup model showed advantages over other approaches in its pilot semester. However, we also discovered that when students worked on the projects outside of the classroom, the team's interactions often strayed from the Agile principles. We were disappointed to find that—although the teams were co-located and were taught to meet frequently face-to-face with their teams' business people—they often resorted to less frequent, online communication [8]. We also anecdotally observed that computer science students had a history of imagining creative and innovative projects, but in our first two semesters of the Tech Startup model, only three (of 100) students from the Software Engineering class even proposed an idea for others to consider. Consequently, the vast majority of the team projects originated from entrepreneurs' proposals. This paper describes interventions we implemented in the Fall 2017 semester to address these concerns.

3 INTERVENTIONS

The Tech Startup model resembles a variation of problem-based learning [9] since lessons in the Software Engineering course are contextualized in how they can be applied to the projects. There is also considerable evidence that suggests that, in comparison to traditional lecture approaches, active learning techniques improved educational outcomes in science, technology, engineering, and mathematics classes [10]. Accordingly, we designed active learning lessons to encourage greater participation in project ideation from software engineers as well as improved adherence to Agile principles.

3.1 Ideation

In the first two semesters of implementing the Tech Startup model, we announced the project to students in both courses and encouraged them to come up with ideas before the following class meeting, where they could propose their idea to all enrolled students. However, since that approach yielded an imbalance with many proposals from entrepreneurs and few from software engineers (despite Software Engineering also having larger rosters), we designed an ideation lesson. Instead of ideas just being proposed on the second day of class, we scheduled both classes to meet together and complete a brainstorming activity during class, followed by time to propose ideas at the end of the meeting.

In the brainstorming activity, students were instructed to list their individual hobbies, interests, talents, and achievements. Subsequently, they identified common themes in their list as well as any possible areas that involved an intersection of multiple interests. Next, they considered problems that impact them and people close to them as well as what might help resolve those problems. After reflecting over the answers, we encouraged students to deliberately explore what unique solutions software might contribute to addressing the identified areas and problems.

Finally, the remainder of the joint class meeting was dedicated to providing each student who wanted to propose an idea to give a brief summary of it to the class. Students came to the front of the class to share their idea and the instructors took note of each idea. All students indicated their top three choices for projects that interested them on an online survey conducted as homework.

In previous semesters, we taught the principles of Agile and Lean Startup that emphasized frequent face-to-face meetings. However, we found that students did not follow through when

outside of the classroom. Our simple intervention was to state explicitly to both classes that they were expected to meet with their respective partners face-to-face at least once per week. Entrepreneurs were also invited to join the Software Engineering lab periods (following the Scrum daily standup meetings) if it worked with their schedules.

In order for a proposed project to be approved, the student who proposed it was required to agree to the (at least) weekly meetings. Once teams were formed, they were also directed to find common times where they could meet outside of class. Expectations for face-to-face meetings were not unique to our third semester. However, the more overt emphasis on requiring teams to meet in person each week was an attempt to improve student adherence to Agile principles when they were not under direct supervision.

3.3 Show-and-Tell

We were also concerned that teams might divert from planning and executing short iterations for feedback and adaptation. To hold teams accountable for making incremental progress, we held two joint meetings of the classes with five weeks in between. During these show-and-tell meetings, each team was required to *show* what was working in the software and *tell* the audience what value it was offering the customers.

Although this practice itself is not necessarily an Agile method, it was designed as an educational intervention for holding teams accountable for being able to explain what the customer needs and to have corresponding working software. It is emphasized that only working features can be shown and the teams are not to talk about features that are not yet working. In other words, it promotes a value system of: "*If it is not a delivered feature, it does not yet exist,*" that emphasizes delivering working software instead of just reporting works-in-progress to something that might work in the future.

3.4 Peer Reviews

Both instructors observed team interactions and took note of individuals' behaviors and contributions. In previous semesters, we also required students to complete periodic peer reviews of each member of the team (themselves included) to supplement the instructors' observations. However, manually collating responses to provide students with aggregated feedback was a laborious and time-consuming process.

In the Fall 2017 semester, we replaced the peer reviews we had been using in favor of CATME [17]. CATME is an online tool that provides peer evaluations with the ability to automate feedback to each student with insights on their own teamwork. We hoped the rapid feedback from a validated instrument for assessing teamwork would provide an improved reflection process that could complement Scrum retrospectives—a meeting at the end of a sprint to adapt the team's organization and methods for the next sprint [24].

3.5 Tech Showcase

Finally, at the end of the semester, we organized a "Tech Showcase" where each team was expected to set up a booth to demonstrate their product. The showcase was open to the general public. To add extra incentives for teams to impress the audience, we also solicited judges (and donations of small prizes such as corporate merchandise, water bottles, and Raspberry Pi mini computer kits) from local industry to award teams who had the most viable business (Tech Startup Award) and who had the most innovative and well-implemented software (Innovation Award). This intervention was not new to the Fall 2017 semester. However, since we were evaluating our interventions, we sought feedback on the showcase as well.

4 FORMATIVE EVALUATION

At the end of the semester, the Software Engineering instructor offered the students an opportunity to participate in focus groups to provide formative feedback on their experiences on the team project. Participation was voluntary and students were offered food and refreshments as well as a credit to drop their lowest (non-project) assignment grade. Seventeen students, representing seven of the eight projects from the Fall 2017 semester, volunteered and participated across three separate focus group meetings and responded to questionnaires.

4.1 Focus Groups

Each focus group followed a common procedure but allowed for flexibility to ask follow-up questions when the moderator sought further insight on specific comments. A focus group moderator's guide was designed to elicit experiences and provide feedback on how to improve learning activities for future semesters. Participants within the same project teams were spread across multiple focus groups as much as possible so that no individual focus group would be disproportionately influenced by a single team's experience.

Focus groups can sometimes suffer from social pressures for conformity or dominance by individuals with strong opinions. However, we followed best practices from focus group methodology to encourage all students to share their thoughts and avoid groupthink [15]. The moderator emphasized to participants that all feedback was useful, including if it dissented from what others' shared. To avoid the discussion being dominated by more assertive participants, the moderator specifically asked less vocal participants to share their thoughts when they had not yet done so. By conducting multiple, independent focus groups, we also lessened the risk of arriving at conclusions based on the social dynamics of one group and instead were able to observe common themes across multiple groups.

The focus groups were held after the course was over, but participants were reassured that the content of their responses—both positive and negative—would have no influence on their grades. Similarly, while we took notes about participants' responses, those notes never included personally identifying information and so students were assured of their anonymity.

The topics discussed in the focus groups addressed activities from the class, in chronological order of when they were introduced. The primary topics addressed were: brainstorming (ideation), proposing projects, choosing projects, planning sprints, face-to-face meetings with clients, show-and-tell meetings, CATME peer reviews, and the tech showcase event. At the conclusion of the focus groups, each participant completed an anonymous questionnaire about their personal experiences.

4.2 Questionnaire

The questionnaire was designed to compare experiences with the different learning activities in regards to objectives for the course. The questionnaire began with open-ended responses to the questions: *Q01: What is your major? Q02: Which project did you work on?* and *Q03: Outside of [the Software Engineering course] students, who was involved in this project and what role(s) did they play in the project?*

The remaining questions were Likert-type items on a five-point scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The following five items directly regarded interactions with their client and planning their project: *Q04: I had a client who helped write user stories, Q05: I had a client who provided guidance on prioritizing deliverables, Q06: I had a client who provided feedback on deliverables on at least a weekly basis, Q07: Project planning activities were valuable to the project's success, and Q08: Project planning activities were valuable to my personal learning.*

Then, the course's primary learning activities (show-and-tell, face-to-face meetings, scrum meetings, CATME peer reviews, and Tech Showcase) were each rated on the same five-point scale to indicate degree of agreement with each of the following statements: "*{Name of Activity}* ...helped the team focus on continuously delivering working software," "...helped the team develop the product incrementally," "...helped the team get valuable insight from outside the team," "...was valuable to the project's success," and "...was valuable to my personal learning."

Although the questionnaire had more participants (n=17), we excluded some students' responses from our analysis to only represent those whose teams adopted the Tech Startup model (n=12). One other team followed the LFOSS model [5]. The Fall 2017 semester also had an uncharacteristically low enrollment in the Entrepreneurship course; so another team was permitted to pursue a "toy project" despite not having a business collaborator. Those students were included in the focus group and questionnaire to provide equal opportunity to the incentives but their data was excluded for this study's formative evaluation.

4.3 Lessons Learned

After the ideation activity, students who wanted to propose their ideas as projects were each given about a minute to explain the product before all students were given time to talk to the proposers. Two (of eight) project proposals came from students enrolled in Software Engineering. A quarter of the proposals coming from software engineers is consistent with previous semesters (which had a combined three proposals over two semesters), but still reflects an imbalance between who proposes Tech Startup projects. The ideation exercise alone was inadequate to encourage more proposals from software engineers.

During the focus group, we asked students about the brainstorming activity. For the vast majority of the participants who had not proposed an idea, we also asked for them to explain why they did not. The overall consensus was that the Software Engineering students wanted more time to think about ideas and some participants (n=3) even remarked that they come up with ideas "too late," after the class had already formed teams. These students also conceded that without a clear proposal, they decided to settle on joining a team whose project was more carefully thought through. It was observed that several of the entrepreneurs seemed to have thought of their ideas before the semester began and had better-rehearsed proposals, sometimes in the format of an elevator pitch.

Upon receiving this feedback, the moderator probed further by asking what would have made the students more likely to propose their own ideas. The students explained that they were unconfident in what made for a good project and all three focus groups suggested providing examples of previous, successful projects. In addition, one participant suggested using the brainstorming activity as a homework assignment to allow for more time to think about possible projects. Likewise, a consensus in each focus group wanted more time to discuss the proposed ideas before selecting which one(s) interested them.

A student elaborated that they chose a project because it was described as a web application, but was disappointed to discover upon being assigned to that team that the proposer had a specific platform in mind (Ruby on Rails) when the student was hoping to use a different one (Django). That student explained that more time for discussion could have clarified that misconception and he would have chosen different projects on the survey as a result. This student's motivation to work on a specific technology is common since our previous study indicated that Software Engineering students are more likely to base their project preferences on technologies they want to learn than the project's purpose or perceived likelihood to make money [6].

Consequently, there are multiple ways we plan on improving the likelihood of Software Engineering students to propose projects. As recommended in the focus groups, we plan to introduce an overview of the projects and assign the ideation activity as homework (to be completed and turned in before the second class). The next class meeting will be combined Software Engineering and Entrepreneurship and will *begin* with brief project proposals. Following proposals, all students will be provided more time to talk with the proposers and discuss details of the idea as well as any specific expectations (such as technologies used).

However, our observations of the ideas proposed over the last three semesters of the Tech Startup model reflected the insights from the focus groups: Software Engineering students are often most motivated by technology and technical challenges while Entrepreneurship students are more driven by business viability and clarity of the project's vision. To elicit more proposals from technically minded students, it might be effective to combine the

initial project overview with brief summaries of emergent technologies to lead students to consider applications of cutting edge technology to address real life problems. For example, summarizing challenges and opportunities for Internet-of-Things products may motivate students to generate ideas that utilize that technology while they brainstorm for homework.

We also discovered areas for improvement after the ideas had been proposed and teams formed. In particular, focus group participants reported that the teams' respective entrepreneurs on their teams struggled to provide clear business needs. As an exception, both participants from one team (in different focus groups) reported that their entrepreneur excelled at communicating the desired features. However, that entrepreneur was, "Always happy with whatever we did, even when we did not finish everything we said we would" so there was not much accountability.

In addition, students' responses to Q04-Q06 suggested that most entrepreneurs did not help prioritize work to be done (M=2.58, sd=1.44) nor did they provide useful feedback each week (M=2.58, sd=1.67). While students reported that the Software Engineering lesson on user stories and sprint planning was moderately helpful to their projects' success (M=3.58, sd=1.16) and was valuable to their learning (M=3.75, sd=1.06), they reported that their entrepreneurs did not provide help writing user stories (M=2.33, sd=1.23).

Likewise, students reported that their face-to-face meetings with entrepreneurs did not help the team focus on continuously delivering working software (M=2.82, sd=0.98); did help incremental development (M=2.73, sd=0.90); did not provide valuable insights (M=2.64, sd=1.03); did not promote the project's success (M=2.45, sd=0.93), and were not valuable to their learning (M=2.63, sd=0.92). From our observations and these results, we found a clear disconnect between the disciplines with a need for improved communications.

Communication across disciplines is a challenging task and even one faced in industry. However, there are techniques that can be incorporated into the Tech Startup model to foster more effective communication. We anecdotally observed that some entrepreneurs thought that providing less guidance was doing the software engineers a favor by giving them more freedom, when instead it left developers with a lack of direction. Meanwhile, our Software Engineering students typically only have previous experience in programming for very detailed and rigid programming assignments. Therefore, learning how to better navigate requirements gathering is a useful skill to cultivate. We had taught the Software Engineering class about user stories and how to use them to help construct a product backlog and to plan for sprints. However, the students did not consistently adopting the method with their entrepreneurs nor customers outside of class.

On the other hand, user stories may be an effective tool to help bridge the communication gap. Since user stories are situated in the context of the end-user's (customer's) needs, they do not require technical expertise to write. Moreover, since the Entrepreneurship course focuses on discovering more about the customer, it should be beneficial for both classes to learn how to incorporate user stories into project planning and adapting to evolving requirements. Consequently, we plan to update the lessons on user stories and sprint planning to have both classes meet together so teams can practice the activity together while receiving guidance from the instructors. With that initial interaction and guidance, we expect that teams will be able to communicate more effectively with a focus on discovering and adapting to customer needs.

When reviewing our show-and-tell meetings, we were satisfied with the technique. Although those meetings take class time, we considered it a good opportunity to hold teams accountable for demonstrating real progress on their software products. Similarly, the Software Engineering students shared in the focus groups that the periodic, brief presentations helped keep them on track. In each of the three focus groups, at least one participant commented that show-and-tells, "Burned a fire under our [seats]" and the consensus was that teams worked extra hard before those deadlines because they did not want to be embarrassed in front of the whole class. Accordingly, the questionnaire responses indicated that show-and-tells helped teams focus on delivering working software (M=4.17, sd=0.94); helped incremental development (M=4.08, sd=0.79); helped the team's success (M=4.00, sd=1.13); and moderately agreed that it helped gain outsiders' insights (M=3.58, sd=1.16) and contributed to their learning (M=3.83, sd=1.03).

Similarly, the end of semester 'Tech Showcase' event went well and received positive feedback from students. Questionnaire responses indicated that the showcase helped the teams focus on delivering working software (M=4.08, sd=0.79); helped gain outsider insights (M=4.33, sd=0.89); supported team success (M=4.25, sd=0.62); was valuable to learning (M=3.92, sd=1.16); and was moderately helpful to motivate teams to work incrementally (M=3.58, sd=1.08). The focus group participants expressed particular appreciation for interacting with judges and the general population. They commented that it was a challenge to communicate the value of their product to someone completely unfamiliar with it, but that it was reassuring to receive positive feedback and constructive conversations with peers and professionals alike.

On the other hand, the focus groups each provided mixed feedback on using CATME for peer reviews. Most students reported that the peer reviews did not have much of an impact on how their teams interacted and performed. However, at least one participant in each of the three focus groups expressed some appreciation of the automatic feedback the tool provided. More so, several participants commented that being able to provide honest, anonymous reviews was cathartic and, as one student said, "Helped me get some issues off my chest, which relieved some stress. I got to spill my frustration about [my teammate's] slacking off, while I knew that [the teammate] would hear it without me having to start a yelling match." Similarly, as instructors, the ratings helped supplement our observations of the teams and intervene when necessary.

Finally, we compared the five principle interventions on how they impacted students' perceptions of their project outcomes. With regards to facilitating delivery of working software,

students indicated that the show-and-tells (M=4.17, sd=0.94), tech showcase (M=4.08, sd=0.79) and scrum daily standups (M=3.58, sd=1.08) all helped, while peer reviews (M=2.83, sd=1.03) and face-to-face meetings (M=2.81, sd=0.98) did not. The ratings are illustrated in Fig. 1.

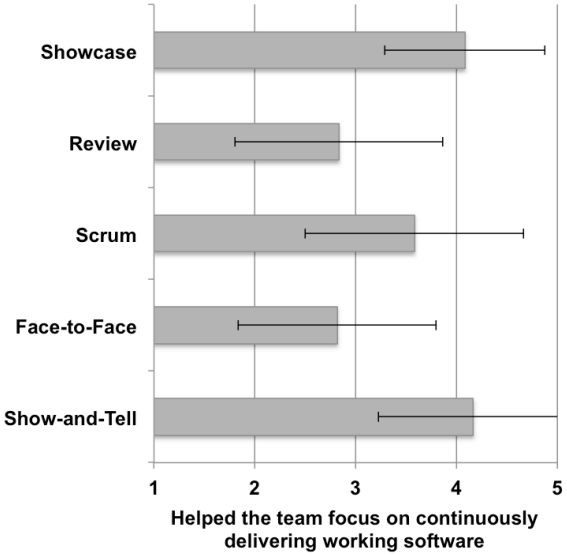

Figure 1: Learning Activities' impact on delivering working software, from 1 (Strongly Disagree) to 5 (Strongly Agree)

Surprisingly, the tech showcase was perceived as nearly as helpful as the show-and-tell meetings, even though it only occurred once at the end of the semester. However, students perceived it as a major point of accountability. These findings particularly illustrate the need to improve team interactions in face-to-face meetings so that accountability is driven by a desire to meet the business' and customers' needs at least as much as the desire to impress their peers and instructors.

Fig. 2 demonstrates similar perceptions about the activities' influences on working incrementally. Show-and-tell (M=4.08, sd=0.79), scrums (M=3.91, sd=0.90) and the showcase (M=3.58, sd=1.08) all helped the teams develop their software products incrementally. Meanwhile, students rated the tech showcase (M=4.33, sd=0.89) as most valuable for gaining external insights about their products. Responses also indicated moderate agreement that the show-and-tells (M=3.58, sd=1.16) provided valuable insights while none of the other activities did, as shown in Fig. 3.

5 CONCLUSIONS

The third semester of implementing the Tech Startup model in an undergraduate Software Engineering class showed some potential for exposing students to more realistic development experience, especially with the challenges associated with working on cross-disciplinary teams with entrepreneurs. However, instructor observations supplemented by student feedback via focus groups and questionnaires found areas that require improvement. In particular, our formative evaluation revealed that the cross-disciplinary team interactions outside of class did not properly facilitate Agile and Lean Startup principles of "failing fast" and adaptations to new discoveries.

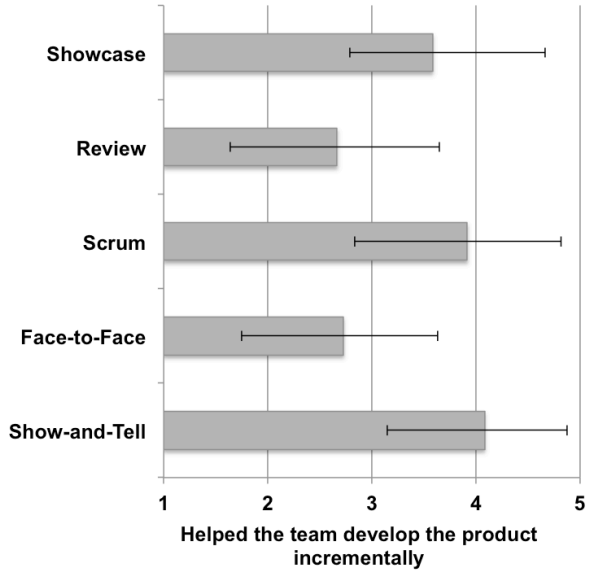

Figure 2: Learning Activities' helpfulness on supporting incremental development, from 1 (Strongly Disagree) to 5 (Strongly Agree)

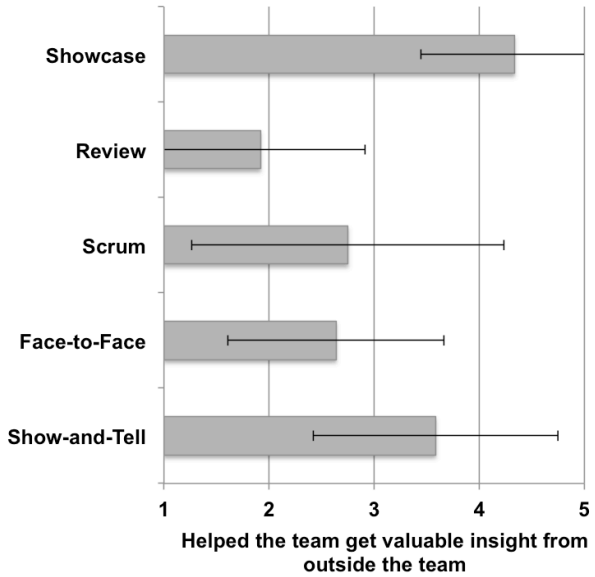

Figure 3: Learning Activities' helpfulness on supporting incremental development, from 1 (Strongly Disagree) to 5 (Strongly Agree)

Student feedback made it evident that the two disciplines were not adequately prepared to communicate effectively to make good use of their face-to-face meetings. We also found that—perhaps as a consequence of poor communication—the teams did not take full advantage of each other's unique talents to find out more about customers and build software that adaptively meets their needs. We propose that incorporating a joint lesson with both classes learning how to write user stories to describe customer's needs should help facilitate better communication.

By learning about user stories and sprint planning activities, entrepreneurs should gain better appreciation of how their feedback can help developers understand a shared, clearer vision of the product. Once the teams experience the activity under supervision, we hope they will be able to integrate it into their outside of the classroom interactions. As a result, both disciplines should demonstrate techniques that more closely resemble professional Agile and Lean Startup practices.

Meanwhile, we integrated two new and effective learning activities into the Tech Startup model. Periodic show-and-tell meetings where teams briefly demonstrate what is working on their software and discuss how it provides value to their customers helped motivate teams to deliver working features incrementally. Similarly, students particularly appreciated the opportunity to demonstrate the value of their work and receive feedback from a diverse audience in a showcase at the end of the term. Although neither activity is inherent to Agile or Lean Startup, they both helped foster a sense of accountability and provided insights from people outside of the teams.

We found that despite our efforts to promote equitable number of proposed ideas across disciplines, Software Engineering students were more reticent to share their ideas. Based on focus group feedback, it appears that providing example projects and allowing for time outside of class may help software engineers feel more confident in proposing project ideas. An introduction to preparing elevator pitches might also improve the proposal presentation. Since software engineers tend to be motivated by the excitement and challenge of learning new technologies and programming languages, we also suspect that encouraging students to consider emergent technologies might help provoke their creative interests as well.

At the conclusion of the Fall 2017 semester, two different teams brought it to our attention that they are continuing their pursuit of their respective tech startups. One project (proposed by an entrepreneur) plans to continue development as they use the services of a local business incubator and hope to launch their web application publicly soon. The other (proposed by a software engineer) decided to slightly pivot the vision of their product to accommodate a different market; two of the team's developers are in the process of registering a business as they make the appropriate adaptations to their product.

Our implementation of the Tech Startup model similarly requires some iteration and revision to better support adherence to Agile methods. However, it shows promise to support *millennial* students' entrepreneurial aspirations. Despite a shortage of funding and likelihood of personnel turnover from student projects, the model equips students to create tech startups while experiencing how to work within multi-disciplinary teams and apply Agile methods.

REFERENCES

[1] American Society for Engineering Education. 2013. "Phase I: Synthesizing and Integrating Industry Perspectives Meeting Report 2013" *Transforming Undergraduate Education in Engineering*, Workshop Report, May 2013.

[2] L. Auer, J. Juntunen, and P. Ojala. 2011. "Open Source Project as a Pedagogical Tool in Higher Education," *Proc. of the Intern'l Academic MindTrek Conf.*, ACM, 2011. 207-213.

[3] M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, K. Schwaber, J. Sutherland, D. Thomas. 2001. "Manifesto for Agile Software Development." http://agilemanifesto.org/ Accessed Feburary, 2018

[4] BNP Paribas. 2015. "The Emergence of the 'Millennipreneur'," *BNP Paribas Global entrepreneurs Report 2016*, BNP Paribas: November 2015.

[5] K. Buffardi. 2015. "Localized Open Source Collaboration in Software Engineering Education." *Proc. of IEEE Frontiers in Education*, El Paso, TX. doi: 10.1109/FIE.2015.7344142

[6] K. Buffardi. 2016. "Localized open source software projects: Exploring realism and motivation." Proc. of IEEE International conference on Computer Science & Education, Nagoya, Japan. 382-387. doi: 10.1109/ICCSE.2016.7581611

[7] K. Buffardi, C. Robb, and D. Rahn. 2017. "Tech startups: realistic software engineering projects with interdisciplinary collaboration." *Journal of Computer Science in Colleges*. 32(4), 93-98.

[8] K. Buffardi, C. Robb, and D. Rahn. 2017. "Learning Agile with Tech Startup Software Engineering Projects." *In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. ACM, New York, NY, USA. 28-33. doi:10.1145/3059009.3059063

[9] S. C. dos Santos, M. da Conceição Moraes Batista, A. P. C. Cavalcanti, J. O. Albuquerque and S. R. L. Meira. 2009. "Applying PBL in Software Engineering Education," *22nd Conference on Software Engineering Education and Training*, Hyderabad, Andhra Pradesh, 182-189. doi: 10.1109/CSEET.2009.39

[10] S. Freeman, S.L. Eddy, M. McDonough, M.K. Smith, N. Okoroafor, H. Jordt, M.P. Wenderoth. 2014. "Active learning boosts performance in STEM courses." *Proceedings of the National Academy of Sciences*, 111 (23), doi: 10.1073/pnas.1319030111

[11] M. Hollenhorst. 2016. "Millennials want to be entrepreneurs, but a tough economy stands in their way," *North State Public Radio*, September 26, 2016.

[12] J. Litton, R. Patterson, and A. Little. 2014. "Business organization legal issues arising from ideas generated by university students." *Southern Law Journal*. 24(2), 267-280.

[13] F. Martin. 2006. "Toy projects considered harmful." *Commun. ACM* 49(7), July, 2006. 113-116. doi:10.1145/1139922.1139958

[14] M. Moyer. 2012. *Slicing Pie: Funding Your Company Without Funds*. Lake Shark Ventures.

[15] J. Nielsen and R.L. Mack. 1994. *Usability Inspection Methods*. John Wiley & Sons, New York, NY.

[16] T. Nurkkala and S. Brandle. 2011. "Software studio: teaching professional software engineering." *In Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11)*. ACM, New York, NY, USA. 153-158. doi:10.1145/1953163.1953209

[17] M.W. Ohland, M.L. Loughry, D.J. Woehr, C.J. Finelli, L.G. Bullard, R.M. Felder, R.A. Layton, H.R. Pomeranz, & D.G. Schmucker. 2012. "The comprehensive assessment of team member effectiveness: Development of a behaviorally anchored rating scale for self and peer evaluation." *Academy of Management Learning & Education*, 11 (4), 609-630.

[18] M. Orsted. 2000. "Software development engineer in Microsoft. A subjective view of soft skills required." *Proc. of the Intern'l Conference on Software Engineering*, IEEE.

[19] S. Phillips. 2007. "A brief history of Facebook," *The Guardian*, 25(7).

[20] D. Rahn, T. Schakett, and D. Tomczyk. 2015. "Building an Intellectual Property and Equity Ownership Policy for Entrepreneurship Programs." *Journal of Entrepreneurship Education*, 49(1), 55-70.

[21] E. Ries. 2009. "Minimum viable product: a guide." Startup Lessons Learned.

[22] E. Ries. 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

[23] C. Robb, D. Rahn, and K. Buffardi. 2017. "Tech Startups: A Model for Realistic Entrepreneurship & Software Engineering Project Collaboration." *United States Association for Small Business and Entrepreneurship. Conference Proceedings*; Boca Raton, 1280-1294.

[24] K. Schwaber and J. Sutherland. 2017. "Scrum Guide." https://www.scrumguides.org/. Accessed: February 2018.