Implementing the Tech Startup Model: A Retrospective on Year One

Dr. Kevin Buffardi, California State University, Chico

Dr. Buffardi is an Assistant Professor of Computer Science at California State University, Chico. After gaining industry experience as a usability and human factors engineering specialist, he earned a Ph.D. in Computer Science from Virginia Tech. His research concentrates on software engineering education, software testing, and eLearning tools.

William Zamora, Califonia State University, Chico

William Zamora is a third-year undergraduate student at California State University, Chico and Research Assistant for the Chico STEM Connections Collaborative. He is majoring in Computer Information Systems with a minor in Computer Science. William's interests include Software Engineering and pursuing a career in academia.

Dr. Colleen Robb, California State University, Chico

Dr. Robb is an Assistant Professor of Entrepreneurship at California State University, Chico. She also serves as the Director for the Center for Entrepreneurship.

David Rahn, California State University, Chico

Mr. Rahn is a Lecturer for Strategy and Entrepreneurship and is the Director of the e-Incubator within the Center for entrepreneurship at California State University, Chico. Mr. Rahn has extensive industry background with software and consulting startups and specialized in new product and market development. Following his successful industry career Mr. Rahn transitioned to teaching strategy and entrepreneurship at Chico State. Over the past 16 years Mr. Rahn has developed the e-Incubator at Chico State, as well as created a course called Web-based entrepreneurship which focuses on helping students launch the online portion of their businesses using the Lean Startup approach. In 2016 he published "e-Business for Entrepreneurs," an online course for entrepreneurs building e-businesses.

Implementing the Tech Startup Model: A Retrospective on Year One

Introduction.

Software engineering is a popular career path for students in computer science and closely-related disciplines. The Bureau for Labor Statistics indicates both "software publishers" and "computer systems design" are among the fastest-growing industries and they even project increased demand for software engineers in coming years [1]. Consequently, courses on software engineering may be the most directly-relevant to many students' careers. However, a primary challenge to teaching software engineering is exposing students to a process and environment resembling industry, while restricted to the confines of an academic setting.

Problem-based learning (PBL) is often employed in software engineering courses by teams of students learning from hands-on experience of software development skills and concepts, while working toward a whole-term software project [2]. However, it does not facilitate a realistic industry experience and may even be harmful to students' education when they define their own "toy projects" for the class [3]. In particular, Nurkkala and Brandle [4] observed that, "The most significant gap," between software engineering projects and industry practice, "is that student projects seldom involve a real customer." Real customers have a stake in the quality and timeliness of software deliverables and therefore hold the development team accountable. Likewise, contemporary practices in the software industry have evolved with an emphasis on customer involvement in the development process.

Agile software development emerged as the most prominent approach to software development around the turn of the millenium. Still widely-adopted, Agile contrasts with previous approaches that had rigid planning by emphasizing adapting quickly to evolving customer needs. Agile enables responding to change by operating on short intervals of delivering software to the customer, eliciting feedback, and adjusting accordingly. The Agile principles [5] include: "Business people and developers must work together daily throughout the project, [...] the most efficient and effective method of conveying information to and within a development team is face-to-face conversation, [...and] our highest priority is to satisfy the customer through early and continuous delivery of valuable software." Consequently, students do not experience genuine Agile development without working directly with customers and business people.

However, pairing software engineering students with business customers can be problematic on several fronts. Firstly—although the software industry is growing—not all universities are located near software production firms and therefore, students may be unable to meet face-to-face with business people, as Agile prescribes. Even where there are local software companies, there can be further obstacles including: difficulty recruiting the companies to accept the work of students; non-disclosure and non-competitive agreements limit students' abilities to demonstrate their work in a portfolio; some universities have policies that require students and/or the institution receive compensation, which may discourage some potential industry partners; and industry software development does not typically operate on schedules compatible with the beginning and ending of academic terms. As we discuss in the following section, educators have

attempted different approaches to identifying realistic software projects, but doing so is often at odds with adopting Agile methods.

Meanwhile, students in Entrepreneurship programs also require experiential learning to transition from a "novice entrepreneurship mindset" to practice entrepreneurial processes and develop practical skills [6]. Conveniently, Ries [7] pioneered the Lean Startup approach, which, "brings together the principles of customer development, agile methodologies and lean practices" [8]. Lean Startup is influenced by Agile's incremental approach and benefits from frequent (business hypothesis) tests and corrections. Short iterations of gathering feedback from customers helps mitigate risk by reducing the waste associated with dedicating time and resources to false hypotheses.

Consequently, we introduced the Tech Startup model: an approach to teaching Software Engineering and Entrepreneurship courses in coordination, where students collaborate to create real technology startup businesses (tech startups) [9][10]. The model incorporates experiential learning of Agile software development practices while software engineering students form teams with entrepreneurs who learn Lean Startup in concert. This paper describes the experiences and observations while adopting the Tech Startup model in its first two academic terms. We summarize how the Tech Startup model is applied to a Software Engineering course and reflect on the lessons learned. We supplement the instructors' observations with data collected from surveys on students' attitudes and experiences. We also describe a formative evaluation and compare how our adaptations to the courses' implementation of the Tech Startup model affected students' learning experiences. Finally, we recommend best practices and additional learning activities based on our findings.

Background.

Active learning pedagogy contrasts with traditional lecture format by engaging students directly in activities to learn the material, rather than just receiving information. Comprehensive meta-analysis of comparing active learning to lecture-based lessons found active learning improved performance on examinations and decreased failure rates by 55% in science, technology, engineering, and mathematics classes [11]. Furthermore, Problem-Based Learning (PBL) is an active learning technique that involves students attempting to solve open-ended problems and discovering what they need to learn in order to solve the problem. PBL is often applied to Software Engineering classrooms, where students learn about the software development life cycle while also experiencing the application of contemporary tools and techniques to developing real software products [2].

Nevertheless, strides still need to be made for software engineering class projects to more closely resemble the experience and practices in the software industry. In their study of project realism, Nurkkala and Brandle [4] identified six common shortcomings: the lack of a real product, relatively short duration, high personnel turnover, low sophistication of software, no software maintenance, and no customer. Similarly, Martin recognized that without customers or

other stakeholders, “Students know their code matters only as much as they might find our assignments interesting, or as much as it counts toward their grades” [3].

However, there have been some attempts at improving project realism. For example, involving students in existing free and open source software (FOSS) projects helps address the issues of having a real product and experience maintaining software [12]. Moreover, Ellis and Morelli [13] proposed focusing on FOSS projects with humanitarian purposes to further expose students to social impacts of computing. While these approaches offer unique benefits, FOSS contributor teams are globally distributed and consequently, it is not plausible to adhere to the Agile principle of communicating face-to-face [5]. Likewise, from our observations, the FOSS community has a concentration of software developers and not nearly as many business people, often making it difficult to fulfill the principle of developers working with business people on a daily basis. In the meantime, industry has identified a need for students to gain more “soft skills” including communication across disciplines [14].

To try to reconcile the advantages of FOSS contribution with Agile principles, we founded an open source consortium in our local community to foster students collaborating side-by-side with software professionals [15]. Localized FOSS (LFOSS) projects help students model their communication, behavior, and technical skills from professionals in industry and demonstrated unique advantages over projects with no external stakeholders [16] and more natural adherence to Agile than projects with only remote collaboration [17]. However, organizing a local open source organization may be difficult in some communities, particularly if there is not much of a presence from the technology sector. Likewise, we have found that the professionals mostly volunteer their time for the open source projects so their involvement is susceptible to interruption due to busy weeks at work or other personal matters. The size of our organization can support mentoring a small group of students (usually around four to six) each semester, but it would not scale well to the entire class (usually around thirty to fifty students). Consequently, we needed to explore scalable solutions to supporting multiple teams of students each with their own stakeholders.

Meanwhile, ASEE published a report that called for improving students’ entrepreneurship and intrapreneurship skills [18]. There are also opportunities for real business success, as evidenced by businesses like Facebook [19] and Google [20] that emerged from projects begun by university students. In addition, each year in the United States alone, fifteen to twenty technology companies are founded that eventually surpass revenues of $100 million dollars [21]. Universities with programs both in computing and business (and particularly in entrepreneurship) are in the position to leverage cross-disciplinary projects to improve project realism and foster entrepreneurial success. Accordingly, we introduced the Tech Startup model to facilitate software engineering projects featuring collaboration with entrepreneurship students to pursue viable businesses with real software products.

Tech Startup Model.

The Tech Startup process begins with the onset of the semester, when instructors present a general overview of semester-long projects and charge students to brainstorm project ideas for software to develop. With the popularity of social media, web services, and prevalence of smartphones, most projects proposed involve web and/or mobile applications. In the following lecture, the project proposals are shared among all students and each student expresses which project(s) to which they wish to contribute. Based on their preferences, we form teams of about 5 software engineering students with 1-2 entrepreneurship students. Students are expected to arrange times outside of class to meet regularly to work on their projects as an interdisciplinary team, while they learn Agile and Lean Startup in their respective lectures.

The Software Engineering course includes two, one-hour lectures per week, each followed by one-hour lab meetings. The lectures concentrate on teaching Agile methods as well as relevant topics including: developer operations, version control, unit testing, design patterns, and software metrics for quality assurance. Lab meetings are mostly dedicated to time for development teams to work on their respective projects.

Our course adopts Scrum [22], a development framework that adhere to Agile principles by compartmentalizing intervals of work into one-to-two-week iterations or "sprints." Sprints involve developers focusing on the feature(s) deemed to deliver the most value for the least amount of work. During a sprint, developers each share their progress in "daily stand-up" meetings (or "scrums") at the beginning of lab periods. At the end of each sprint, teams hold sprint reviews to evaluate what they have and have not accomplished, along with what problems need to be addressed for the following sprint. Likewise, the team holds a sprint retrospective which evaluates the methodology and adapts how they are approaching the software development as necessary. The remainder of time is dedicated to designing, implementing, and testing their software. During labs, the instructor supervises and advises teams.

Software Engineering project grades account for a majority of the class grades but students also complete quizzes and in-class assignments individually. Projects are graded based on the product's usefulness, design, and verification, as assessed by the development team's ability to demonstrate the application of concepts from lecture to their projects. This grade includes a team's self-critique of their software's design and verification by using metrics, tools, and best practices. Each member of the development team also completes periodic peer reviews of each team member (themselves included) and the instructor adjusts individuals' project grades based on their observation and feedback from the peer reviews.

The Web-based Entrepreneurship course from the College of Business, has a similar goal of students understanding and experiencing Lean Startup, a methodology in entrepreneurship focused on short product development cycles. Key topics in the course include customer validation, website content design, as well as marketing planning & campaign execution. Within each key topic area, five lessons are provided which deepen and expand the student understanding of the key topic area. For example, the marketing planning & campaign execution topic includes lessons in setting up eCommerce, marketing plan development, social media plan development, search engine optimization, and web analytics. More details about the

entrepreneurship course structure and assignments are available in our previous publications [9][10].

In order to provide the sense of realism we seek in both classes, students are instructed that their goal is to launch a business based upon the work completed during the course of a semester. They are further encouraged to be aware that after the semester, all members of the team are free to continue to pursue the project, and that various forms of support are available, including mentorship and business incubation. We try to make it clear from the start that it is expected that all students will go beyond just seeking a grade and will dedicate effort that empowers their tech startup to eventually realize a profit.

Once students are coming together to form a business for profit, legal implications emerge which must be dealt with in order to avoid issues that may arise. According to the Uniform Partnership Act [23] a partnership is "an association of two or more persons to carry on as co-owners a business for profit." The Act makes it clear that students in this situation must receive guidance in order to navigate this situation appropriately. Most universities—the authors' own university included—provide clear executive memoranda on the ownership of student projects from the perspective of the university, instructors, and students. Typically in the absence of any well-documented arrangement between the university, professors and students co-working on projects, the student owns the entirety of any intellectual property created. However, the ownership relationship *between students who work together* on a project is not always clear. Many universities are becoming aware of this issue and a variety of approaches are emerging to handle the challenges. [24]

Traditional approaches to equity introduce problems. In one common scenario, students create fixed equity arrangements amongst team members at the onset of a project. In the other common scenario, they postpone decision making with the intention of shaping the equity later when more about the value of member contributions is known. Fixed equity splits are problematic in quite a few ways. One of the most problematic scenarios is when a member loses interest and leaves the team but still keeps equity. In most cases, this creates such low morale that remaining members also lose interest and the project dies since no member wants to pull the project uphill with an undesirable equity split. Delaying the determination of equity, or of a plan for earning equity, creates the default situation mentioned earlier - the inadvertent partnership, which results in an equal equity split across all members, regardless of their contributions. This situation falls prey to the same issues as the fixed equity split.

On the other hand, an approach called Slicing Pie [25] has emerged as a viable process in which to dynamically allocate equity. In this arrangement, different types of contributions are defined and categorized, and valuations are assigned to specific contributions. Contributions in the category of work are typically given hourly dollar valuations (based on the going market rates), along with multiplier values based on whether payment is made or not (multipliers are intended to handle the "at-risk" nature of certain contributions.) Contributions in the form of cash, property, or other goods (such as paying for web hosting or other services) are given agreed upon values. As the team moves ahead, it tracks the investments for various contributions; in our collaborations, most contributions involve tracking hours worked for the entrepreneurs and developers. In this approach, each individual is earning future equity. Additionally, the team

decides how to handle adding and removing team members with respect to equity. The team agrees ahead of time what should happen to the Pie of an individual with respect to *the reason why* a member is leaving the project. The team may decide that a member who is fired will see a different outcome of their Pie than someone who needs to quit for valid personal reasons beyond their control.

From the perspective of the entrepreneur, adopting dynamic equity sharing via Slicing Pie enables the project to attract developers without having to put forth significant capital. Since lack of developer resources is a major barrier to launching otherwise promising products to market, dynamic equity sharing is a helpful tool. It is impractical to predict the future needs of a project and this approach enables the team to weather numerous changes, including personnel turnover, along the way to venture success.

By simplifying the legality and logistics of adding new team members in subsequent semesters, projects can mature and also create more realistic software engineering experiences. Nurkkala and Brandle [4] identified team member turnover, short duration of projects, and a lack of maintaining (pre-existing) code all as significant hindrances to software engineering students gaining realistic experience. However, Slicing Pie dynamic equity enables teams to continue working on their product after the academic term finishes because even if team members quit contributing to the project, they can find replacements in the following semester.

When a project continues from one semester to the next, the new developers to the team receive their equity while they contribute to the maintenance of existing code. Maintenance requires developers to: read and understand someone else's code; fix bugs and add features within the confines of the existing software design; and appreciate that their work also has to be easy for other engineers to understand, use, and maintain. Consequently, by adopting dynamic equity sharing (via Slicing Pie) in Tech Startup projects, teams may benefit from continued development and be better prepared to survive team member turnover. Meanwhile, software engineering students who join continuing projects can benefit from more realistic software maintenance.

Formative Evaluation.

For several years, we have been gathering pre- and post-project surveys from students in our Software Engineering class. The surveys capture student attitudes, opinions, and intentions regarding computing with Likert-type scale items, as adopted from the Humanitarian FOSS multi-institutional studies [26]. We collect responses to the pre-project surveys after project ideas have been proposed, so we supplement the existing items with questions for students to express their project preferences. In particular, students identify their top three preferred choices for projects to join.

In addition, they elaborate on their motivations with five-point (1-strongly disagree to 5-strongly agree) Likert-type items to rate their agreement with "My project preference is motivated by..." each of the following: *...programming languages and technologies I am already*

familiar with; *...new programming languages and technologies I want to learn*; *...the potential of the product making money*; *...the opportunities for professional/career networking*; *...the potential of the product becoming well-known and used by many people*; and *...the problem the product addresses*. Based on the pre-project responses, instructors form teams of about 5 developers per proposed project. Occasionally, projects are proposed that do not garner enough interest to create a team, but in nearly every case, each student is assigned to one of their top three choices.

In the post-project survey, we collect follow-ups to how their opinions and intentions may have changed but also gather data on their experiences with the project. Since following the Agile principle of working with business people in face-to-face meetings was a primary motivation to shaping the Tech Startup model, we investigated those interactions in our pilot semester. For insights on the teams' interaction, students answered the questions:

INT01. Outside of lecture and lab, estimate how many hours you spent per week on your project *(free response)*

INT02. I directly communicated with the mentor/customer for my project… *Choose one: {I didn't have a mentor or customer; I had a mentor or customer but only other team members directly communicated with them; Less than once a month; Each month; Each week; Each day}*

INT03. My predominant contact with my mentor/customer was… *Choose one: {I didn't have a mentor or customer; I had a mentor or customer but only other team members directly communicated with them; In person; Synchronous communication (Phone, video messaging, instant messaging, etc); Asynchronous communication (Email, bulletin boards, messaging at different times, etc) }*

INT04. My mentor/customer was prompt in replying to me *(5-point Likert-type item)*

INT05. My mentor/customer's communication was helpful to my progress on the software development project *(5-point Likert-type item)*

INT06. My mentor/customer held me accountable to completing my work well and on time *(5-point Likert-type item)*

INT07. Interacting with my mentor/customer was valuable for my professional networking *(5-point Likert-type item)*

We clarified that for the purpose of these questions, their project collaborators who were not enrolled in the Software Engineering class were to be considered a "mentor/customer." In the case of Tech Startup projects, this included their partners in the Entrepreneurship course while other projects (including from previous semesters) included "mentors/customers" such as localized FOSS (LFOSS) consortium group members, globally-distributed FOSS collaborators, and remote industry mentors.

In preliminary analysis of the pilot semester that compared Tech Startup (n=33) teams to the ongoing LFOSS project and other projects from previous semesters, we discovered that—despite the team members' close proximity to each other—only one team unanimously reported face-to-face meetings as the primary mode of communication [27]. We compared the directness of communication by coding responses ordinally (1-asynchronous, 2-synchronous, 3-in person) and using the conservative Bonferroni method to adjust the critical value α=.017) to account for multiple one-way tests. Wilcoxon-Mann-Whitney tests found that Tech Startup groups (M=2.38, sd=0.86) were approaching significance (p=.06) to be *less* direct as the groups who collaborated with LFOSS (M=3.0, sd=0). However, Tech Startup groups approached significantly *more* direct communication than those mentored for proprietary industry projects (p=0.4, M=2.0, sd=0) and traditional FOSS projects (p=.12, M=2.0, sd=0). We also compared the frequency of communication with the development teams' external collaborators by coding responses ordinally (1-"I had a mentor or customer but only other team members directly communicated with them" to 5-"Each Day"). Tech Startup (M=3.19, sd=.91) groups reported less frequent communication with their clients than those who worked with industry (p<.01, M=4.0, sd=0.0), and LFOSS (p<.017, M=4.0, sd=0.0), while approaching significance in comparison to FOSS (p=.09, M=3.83, sd=0.41).

These preliminary findings suggested that the potential advantage of face-to-face communication in Tech Startup groups was undermined by the students' inclination to settle for more convenient (even if less effective) modes of communication, and doing so less frequently than intended by the Tech Startup model's design. We recognized that students' schedules sometimes deter face-to-face communication but also acknowledged that while other types of projects involved collaboration with professionals, Tech Startup collaboration was with peers. It is possible that students feel less pressure to demonstrate professionalism with their peers than they do for professionals (or those perceived as authoritative). However, we also recognized that our survey only addressed teams' *predominant* form of contact and did not reveal if (or how) students may have used multiple forms of contact. Consequently, we adjusted our analysis for the second semester of the Tech Startup implementation.

For the following semester, we explicitly stated that teams were expected to meet with their collaborators face-to-face *at least* on a weekly basis. To accommodate that mandate, we also coordinated class times so that software engineering and entrepreneurship students would not have conflicting time schedules. The Software Engineering lab period was also made available for entrepreneurs to join for meetings, when possible. With these changes, we found that Tech Startup teams went from typically communicating on about a monthly basis (M=3.19, sd=.91, where 3 corresponds to monthly) to about a weekly basis and with less variance (M=3.95, sd=0.65, where 4 corresponds to weekly) in the subsequent semester (n=39). Similarly, we found modest improvements in directness of communication from the first semester (M=2.38, sd=0.86) to the second (M=2.55, sd=0.75). However, this also revealed that other forms of communication were common. To add more granular insight into their communications, we added questions to the post-project survey for the second semester, where students rated the following statements on an ordinal scale (1-never, 2-less than once a month, 3-each month,

4-each week, 5-each day): How often did you personally communicate with your mentor/client "*...face-to-face,*" "*...remotely at the same time,"* and "*...remotely at different times*?"

We found that students in the second semester of our Tech Startup implementation communicated asynchronously approximately monthly (M=2.9, sd=1.25), chatted remote-synchronously less often (M=2.48, sd=1.3), and communicated face-to-face most often (M=3.23, sd=0.93). While the results show room for improvement still, they indicate a step in the right direction and in the following section, we will discuss future plans to continue improving adherence to frequent face-to-face communication. Meanwhile, we also investigated students' attitudes about their cross-disciplinary communications. On a five-point scale (from 1-strongly disagree to 5-strongly agree), students agreed and showed improvement that their entrepreneurs replied promptly from the first semester (M=3.36, sd=1.25) to the second (M=4.0, sd=1.3). Similar agreement and improvement was reported for helpfulness of entrepreneur communication from the first (M=3.09, sd=1.4) to second semester (M=3.85, sd=1.22).

Students in the first semester averaged slight disagreement (M=2.76, sd=1.25) while those in the second semester averaged slight agreement (M=3.19, sd=1.2) that their entrepreneurs "held me accountable to completing my work well and on time" and similarly moved from slight disagreement (M=2.91, sd=1.26) to moderate agreement (M=3.69, sd=1.26) that the relationship was "valuable for my professional networking." A previous comparison between remote and local FOSS projects suggested an association between more direct forms of communication and feelings of being held accountable [17].

Discussion.

Our preliminary findings from the pilot semester demonstrate that simply introducing students to Agile principles is insufficient and does not guarantee that students will meet frequently and face-to-face with their entrepreneurial partners on their own accord. However, by removing some obstacles and emphasizing our expectations to both the Software Engineering and Entrepreneurship students, we found improved adherence. Even so, we observed a need to encourage closer communication with emphasis on the Agile principle of continuously delivering working software. Anecdotally, we observed that entrepreneurs were too often satisfied with the development team *saying* that they were making progress without actually *demonstrating* the working solution.

Accordingly, some teams showed a tendency to revert to focusing on their long-term goal of having something of value to show by the end of the semester. However, when adopting Agile and Lean Startup methods, teams should focus on short iterations for making incremental progress and receiving quick feedback—including, at times, "failing fast," as the adage has been widely-adopted in industry. For that reason, we plan to adopt a mechanism for holding groups accountable for working incrementally. We propose scheduling periodic "show-and-tell" meetings where both classes meet at once and each group has a brief time to *show* what features work and *tell* the class how it has been adapted to suit what entrepreneurs have discovered about

the business' and customers' needs. In future work, we plan to evaluate whether this activity improves project adherence to Agile and Lean Startup philosophies in the Tech Startup model.

Meanwhile, we also observed expectations and attitudes of students in the Web-based Entrepreneurship class. Many students majoring in entrepreneurship find that they need developers in order to execute their idea, particularly with the current prevalence of social media and web/mobile applications in everyday life. More often than not, when a student or recently graduated entrepreneur seeks out developers on their own, they lack the ability to effectively communicate their needs to the professional.

On the other hand, this type of interdisciplinary program gives entrepreneurship students a unique opportunity to learn the various communication techniques needed for startup collaboration prior to a situation where costly mistakes and missteps may occur. Whether the entrepreneurship student launches a startup or goes to work in a large corporation, their ability to communicate with software engineers puts them in a advantageous position to excel. While some students are able to see the value of this skill prior to the project, most students do not value it until mid-semester, when they start to appreciate the different communication and thinking styles.

There are also some initial hurdles to overcome when convincing entrepreneurship students the value proposition of collaborating with developers. Historically, one of the biggest impediments to entrepreneurs launching businesses has been their inability to attract a technical partner who believes in the product idea as much as the entrepreneur and who is willing to provide the same level of sacrifice, and work towards a launch for some hoped for equity. The Tech Startup model aims to address that. However, without the exposure to what is involved in software development, entrepreneurship students do not always appreciate the skill and time it demands at the onset of the semester.

Anecdotally in the past, some of the more clever entrepreneurship students have reached out to their instructors having found a "How to Learn Programming in 24 Hours" book. They sincerely point out that "if it really is only 24 hours to become a coder, then that is something I can handle!" However, after a little exploring, they usually explain that they started reading and skipped around before conceding, "I was okay up through 'Hello World,' but then I lost it." However, it is not until that experience that some entrepreneurs appreciate the talents of software engineers.

We have also found that entrepreneurs are uniformly surprised when, having had their project chosen, they are given a team of four or more engineers. On the financial front, they are immediately struck by the fact that they will be outnumbered by engineers and they perceive an immediate imbalance in the equity distribution that is pending. Despite being told ahead of time that several developers will be assigned to the team, when they encounter the actual Slicing Pie assignment, they demonstrate some hesitation. Both behaviors represent uninformed thinking and can be rectified by helping entrepreneurs understand the value proposition of a dedicated software development team. It may also help improve the Slicing Pie activity itself if teams are

prompted with a point of discussion of whether they want to value the original idea as a valuable resource in of itself that deserves part of the valuation.

Introducing entrepreneurs to Agile and how it supports the creation of a better product helps them understand the value of development teams. However, to strengthen and encourage earlier appreciation of both disciplines' contributions, more learning activities may be necessary to facilitate useful communication. We hope to better bridge the communication and values gap between Software Engineering and Entrepreneurship students by including an additional lesson that is shared with both groups. User stories following the format: "As a *(user role)*, I want *(goal)* so that *(reason)*" communicate customers' (end users') needs to developers without requiring technical expertise to write them. In previous semesters, we have taught the Software Engineers how to write user stories (and to use them to populate, prioritize, and plan their sprints). However, we plan to include all students in this lesson in the future so that developers can appreciate the insights entrepreneurs discover about the customers, while entrepreneurs gain a better way to communicate the functional requirements of a minimum viable product.

Finally, we observed a trend when proposing projects. Although we give all students (from both classes) the opportunity to propose project ideas, only one of seven Tech Startup proposals came from the Software Engineering students in the pilot semester and one of five the following semester (16.7% overall). We were surprised and discouraged to observe that software engineers were far less likely to propose ideas. We do not attribute this to a lack of ingenuity or motivation to create something of their own. When we examined their responses to the pre-project survey, software engineers indicated they were motivated by (from most to least): languages/technologies they want to learn (M=4.07, sd=0.91), the problem the product addresses (M=3.81, sd=0.9), the likelihood the product will be well-known and widely-used (M=3.78, sd=1.0), languages/technologies they already know (M=3.03, sd=1.19), and (the least by) their expectations for the project to make money (M=2.76, sd=1.33). Consequently, in future work, we will explore how to leverage their interests in learning compelling new technologies to encourage more students in the Software Engineering course to propose their project ideas.

Conclusions.

By design, the Tech Startup model addresses challenges to teaching software engineering by engaging students in partnerships with entrepreneurs as they follow compatible Agile and Lean Startup methods for creating real startup businesses. At the time of writing this paper, two projects from the Tech Startup collaborations are continuing their ventures at a local business incubator while another has reported intentions to legally incorporate. In addition, three of the projects from the first semester continued onto the following semester as the entrepreneur recruited from a new batch of software developers. This continuation is promising as it demonstrates the flexibility that Slicing Pie dynamic equity provides startup projects as they adapt to personnel turnover. Meanwhile, developers still gain recognition for the work they contributed during the semester that could eventually result in a share of future revenue.

Likewise, continuity of projects uniquely provide the possibility for some Tech Startup projects to expose students to the experience of maintaining code.

A reflection over the first year of implementing the Tech Startup model also revealed areas for improvement. Although the approach is designed to foster frequent and face-to-face communication between software engineers and entrepreneurs, students in both disciplines require extra encouragement to adopt these practices. By coordinating class logistics and emphasizing an expectation of frequent meetings, we saw an improvement by the second semester of Tech Startup collaborations. We also propose delivering an additional lesson on user stories to both classes to help facilitate communication and appreciation of what either side of the team offers the other. Similarly, we plan to also examine how periodic "show-and-tell" activities will hold teams accountable to following incremental cycles of development, feedback, and adaptation. Finally, we identified a need to encourage software engineers to propose more of their own innovative ideas by leveraging their interest in exploring new technologies and programming languages.

References.

[1] R. Henderson, "Industry employment and output projections to 2024." *Monthly Lab. Rev. 138(1), 2015.*

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

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

[4] T. Nurkkala and S. Brandle, "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, 2011. pp. 153-158.* doi:10.1145/1953163.1953209

[5] 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, "Manifesto for Agile Software Development." *Agile Manifesto*, February, 2001. [Online]. Available: Agile Alliance, http://agilemanifesto.org/. [Accessed Jan. 31, 2018].

[6] N. Krueger, "What lies beneath? The experiential essence of entrepreneurial thinking." *Entrepreneurship Theory and Practice, 31(1), 2009. pp. 123-138.*

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

[8] S. Trimi, and J. Berbegal-Mirabent, "Business model innovation in entrepreneurship." *International Entrepreneurship and Management Journal, 8(4), 2012. pp 449-465.*

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

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

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

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

[13] H.J.C. Ellis and R.A. Morelli, R.A., "Support for Educating Software Engineers Through Humanitarian Open Source Projects." *Proc. of the 21st IEEE-CS Conf. on Software Engineering Education and Training Workshop. IEEE Computer Society, Washington, DC, 2008. pp 1-4. doi: 10.1109/CSEETW.2008.5*

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

[15] K. Buffardi, "Localized Open Source Collaboration in Software Engineering Education." *Proc. of IEEE Frontiers in Education, 2015.*

[16] K. Buffardi, "Localized open source software projects: Exploring realism and motivation." *Proc. of IEEE International conference on Computer Science & Education, 2016.*

[17] K. Buffardi, "Comparing Remote and Co-located Interaction in Free and Open Source Software Engineering Projects." *Proc. of conf. on Innovation & technology in computer science education, ACM, 2017.*

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

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

[20] Google, "How we started and where we are today" [Online]. https://www.google.com/about/our-story/ Accessed February 2018.

[21] P. Kedrosky, “The Constant: Companies that Matter” *Available at SSRN: 2262948, May 2013. doi: 10.2139/ssrn.2262948*

[22] J. Sutherland, J, "Agile Development: Lessons learned from the first Scrum" *Scrum Alliance, 2004.*

[23] National Conference of Commissioners on Uniform State Laws, "Uniform Partnership Act of 1997," *Annual Conference Meeting in its One-Hundred-and-Fifth Year, American Bar Association, San Antonio, TX, USA, July, 1996.*

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

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

[26] H.J. Ellis, G.W. Hislop, S.M. Pulimood, B. Morgan, and B. Coleman, “Software Engineering Learning in HFOSS: A Multi-Institutional Study.” In *Proc. of the 122nd Annual ASEE Conf. and Exhibition, Seattle, WA, 2015.*

[27] K. Buffardi, C. Robb, and D. Rahn, “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, 2017. pp 28-33*. doi:10.1145/3059009.3059063