Abstract—In this paper we discuss a Software Engineering Without Programming course we taught in the Winter 2020 quarter at Cal Poly, San Luis Obispo. We designed this course to target students outside of computing majors, and taught it with the purpose of getting them familiarized with the software development process. To provide software development support, we taught this course in concert with a traditional Software Engineering I course for Software Engineering majors.

In this paper we concentrate on reporting our experiences related specifically to the teaching early stages of software development: development of software product vision, and requirements gathering. We discuss the topics covered in the course, the activities students participated in and the artifacts they created. Additionally, we report the results of two surveys (one conducted before the course started, and one conducted during Week 8 of the course) detailing student attitudes towards software development and the course material.

I. INTRODUCTION

These days, one does not need a CS degree to come up with an idea for a mobile application, or to work as a product manager in a large software company. The number of software product stakeholders without formal computing education is as large as it has ever been, and is growing. But unlike software developers who almost all take at least one Software Engineering course in their undergraduate study, these stakeholders have never had any educational experiences related to software engineering.

Yet, requirements elicitation, and other project-related communication is a two-way street, and if only one party to the process is well-prepared, well-trained, and understands what needs to be done, this process is not as efficient as it can be, and it can be frustrating and counter-productive.

This paper details our experience piloting a quarter-long Software Engineering without Programming course at Cal Poly, San Luis Obispo. This course targeted non-computing majors, and was paired with a section of the Software Engineering I course offered by the CSSE Department to students majoring (specifically) in Software Engineering. Our approach to teaching the two courses follows the in-concert teaching model pioneered by Dekhtyar and Goodman[1]. In this model, two classes have separate lectures specific to a given audience, but share laboratory time to work in cross-disciplinary teams on shared projects. We use the idea of mutual benefit to facilitate joint work of the students from two different courses.

In this paper, we discuss the learning objectives, and the structure of the two courses, the topics discussed in class, the projects, and the course activities. We primarily concentrate on early stages of the SE lifecycle, specifically, on the joint efforts to develop product vision statements, and determine project requirements. We present some preliminary results of self-assessment, based on entrance and mid-term surveys we conducted in both courses.

II. RELATED WORK

The approach of teaching software engineering courses following a project-based learning approach applying agile practices is not new. There are several reports on teaching software engineering classes with real external stakeholders such as in [2]. Also, Péraire [3] reported on a dual track version of a software engineering course addressing the divide between software engineering and human-computer interaction through an integrated approach to requirements engineering and interaction design. Again, students had available real external stakeholders to communicate with as in a typical industry driven software engineering capstone course.

In another approach reported [4], authors designed a software engineering course as a mentorship experience, in which one college student mentor is connected to one high school student mentee. College students learn to communicate programming concepts and software design to work with colleagues with very different levels of software engineering knowledge, and to overcome problems related to visual accessibility.

Portugal et al. [5] presented a course design to teach requirements engineering based on role playing having CS graduate students and instructors playing the role of consultants and stakeholders and the undergraduate students in the course playing the role of team members.

None of these studies have reported the configuration we have designed and measured in our research. To the best of our knowledge, there is no study reported on teaching a software engineering course for non-CS students in concert with an introductory software engineering course for CS/SE students when students from the former course become the customers to the students from the latter course.

III. MOTIVATION AND RESEARCH QUESTIONS

In deciding to teach a Software Engineering without Programming course targeting non-computing students, we are motivated by two important observations.
Learning Objective
Express requirements for a computer software product
Describe the role of computer software both in everyday life, and in their future professional career
Evaluate software prototypes created by software engineers
Identify the steps of the software development process
Identify different professional roles in the software development process
Express requirements for a computer software product
Evaluate software prototypes created by software engineers

TABLE I
LEARNING OBJECTIVES DEVELOPED FOR THE PILOT SOFTWARE ENGINEERING WITHOUT PROGRAMMING (SE4ALL) COURSE.

<table>
<thead>
<tr>
<th>No.</th>
<th>Learning Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Describe the role of computer software both in everyday life, and in their future professional career</td>
</tr>
<tr>
<td>2</td>
<td>Identify the steps of the software development process</td>
</tr>
<tr>
<td>3</td>
<td>Identify different professional roles in the software development process and describe the responsibilities of non-software-engineering roles</td>
</tr>
<tr>
<td>4</td>
<td>Express requirements for a computer software product</td>
</tr>
<tr>
<td>5</td>
<td>Effectively communicate with software engineers during the software development process</td>
</tr>
<tr>
<td>6</td>
<td>Evaluate software prototypes created by software engineers</td>
</tr>
</tbody>
</table>

During the registration period (in November 2019), once the two sections of the SE4All course were filled, the course instructor sent to all students the description of the course, complete with a draft syllabus that explained to students that they will study the software development process from the perspective of a software product stakeholder. The email attempted to resolve the confusion with the course title; the draft syllabus contained a proposed schedule of topics that included “product vision”, “requirements elicitation”, “user stories”, “software architecture and design”, “UI prototyping”, “validation and verification”, and several others, and discussed the course assignments. Of the 25 students in each section, about five students dropped the course prior to the beginning of the quarter, and a few additional students joined. At the end, each section ended with 24 students.

The two sections of the course had students from 17 different majors. Among them there were 17 Business/Economics majors, five Graphics Communications students, four Animal Science and Psychology/Child Development students each, three Physics and English majors each, two Math and Journalism majors each. Additional majors represented by one student were Agribusiness, Agricultural Engineering, Philosophy, Kinesiology, Recreation, Parks and Tourism, Sociology, and Liberal Arts Engineering.

The course had four contact hours a week, broken into two two-hour lecture periods. During most of the quarter, the second hour of each lecture was used for project-related activities rather than for lectures.

B. Software Engineering I (SEIntro)

CSC 308, Software Engineering I, is the first “true” SE course for the students pursuing the Software Engineering major at Cal Poly. The course is taken by the students during their junior year, and is usually immediately followed by the Software Engineering II course. Students take these two courses in a cohort manner. Typically, this sequence features a team software development project that starts in the SEIntro course, and continues in the Software Engineering II course.

Different instructors approach teaching the SEIntro course in different ways. The version of the course, as taught by the second author exposes students to the principles and the practice of agile software development [6] early in the quar-

---

1General Education (GE) courses are part of University curricula in public universities in the State of California. They are courses that are open to all students on campus, and that do not have prerequisites other than other GE courses.

---

a) Observation 1: Informed participants of ANY process, improve the quality of the process and the interactions within it: Our personal experiences of working on cross-disciplinary projects show that stakeholders who understand the software development process and respect it are easier to work with. Additionally, the software is easier to build, and the path to success is shorter.

b) Observation 2: mutuality of benefits: we believe that integration of non-computing students as customers/product owners/stakeholders for software projects in SE courses benefits both them and the SE students they work with. Students in different software engineering classes tend to work with three types of customers: (a) a pretend customer (e.g., an instructor or a TA playing a role of a stakeholder), (b) themselves as customers (students developing their own project ideas), and (c) a customer who far exceeds them in terms of seniority (outside industry customers on course projects are often senior personnel). The one category of customers they sorely lack is their peers — someone the students can work with instead of for. Our approach is an attempt to remedy that.

From these two observations arise our research questions:

Research Question 1 (RQ1) Can we design an Introduction to Software Engineering course for non computing majors that will prepare them for work on software projects in the industry (and if yes, how should such a course be taught)?

Research Question 2 (RQ2) Is working with non-computing peers on software projects a beneficial experience for students taking a CS/SE Introduction to Software Engineering course?

IV. THE COURSES

This section provides an overview of two courses, Software Engineering Without Programming (SE4All for short), and Introduction to Software Engineering (SEIntro for short) that the co-authors taught in concert in the Winter 2020 quarter.

A. Software Engineering Without Programming (Se4All)

Cal Poly’s CSSE Department offers a General Education1 course CSC 310 with an unfortunate title “Computers For Poets”. The course, which is almost never taught, has a catalog description that says “How computers and computer devices work. Introduction to software systems and applications. How computers connect with various media including images, speech and data. How information is encoded and transmitted across networks. Relationship between the computer and human information processing.”

In Winter 2020, we “hijacked” CSC 310 to teach a course titled “Software Engineering Without Programming” because the its catalog description is sufficiently broad, and because no other course offered by our department was found to be suitable for our pilot. We offered two sections of the course. In preparation for the course, we established the learning objectives shown in Table I.

---

1General Education (GE) courses are part of University curricula in public universities in the State of California. They are courses that are open to all students on campus, and that do not have prerequisites other than other GE courses.

---

During the registration period (in November 2019), once the two sections of the SE4All course were filled, the course instructor sent to all students the description of the course, complete with a draft syllabus that explained to students that they will study the software development process from the perspective of a software product stakeholder. The email attempted to resolve the confusion with the course title; the draft syllabus contained a proposed schedule of topics that included "product vision", "requirements elicitation", "user stories", "software architecture and design", "UI prototyping", "validation and verification", and several others, and discussed the course assignments. Of the 25 students in each section, about five students dropped the course prior to the beginning of the quarter, and a few additional students joined. At the end, each section ended with 24 students.

The two sections of the course had students from 17 different majors. Among them there were 17 Business/Economics majors, five Graphics Communications students, four Animal Science and Psychology/Child Development students each, three Physics and English majors each, two Math and Journalism majors each. Additional majors represented by one student were Agribusiness, Agricultural Engineering, Philosophy, Kinesiology, Recreation, Parks and Tourism, Sociology, and Liberal Arts Engineering.

The course had four contact hours a week, broken into two two-hour lecture periods. During most of the quarter, the second hour of each lecture was used for project-related activities rather than for lectures.

B. Software Engineering I (SEIntro)

CSC 308, Software Engineering I, is the first “true” SE course for the students pursuing the Software Engineering major at Cal Poly. The course is taken by the students during their junior year, and is usually immediately followed by the Software Engineering II course. Students take these two courses in a cohort manner. Typically, this sequence features a team software development project that starts in the SEIntro course, and continues in the Software Engineering II course.

Different instructors approach teaching the SEIntro course in different ways. The version of the course, as taught by the second author exposes students to the principles and the practice of agile software development [6] early in the quar-
C. Running the two courses in concert

We ran one section of SEIntro and two sections of SE4All: the morning section (Section 01) and the afternoon section (Section 02). SE4All Section 01 was taught fully in concert with SEIntro course: the two sections met in close physical proximity to each other on the same day and time twice a week. The students from the two courses formed joint cross-disciplinary project teams, and met during most of the second hours of Monday and Wednesday classes to work together on their joint project. In SE4All Section 02 we formed project teams as well, and matched each Section 02 team with a Section 01 team and the SEIntro team. Each time the morning section had scheduled face time with the SEIntro students, the afternoon section had a team activity to produce the same artifacts as the joint morning lab, but with little-to-no in-person input from the SEIntro students. All artifacts were made available to the SEIntro teams via file sharing.

a) Projects: the key component of in-concert teaching is joint work of students from different courses on shared projects, the work that has students assume different, appropriate to their skills and knowledge, roles on the project. We decided that SE4All students will become product owners - i.e., the initial ideas about the projects would come from them. The main issue for us was to make sure that both SE4All sections (morning and afternoon) can work productively and receive software developer feedback for their work.

During the first week of classes we ran a survey asking students to score their level of interest in working on software in a variety of different categories. Each student gave a soft non-exclusive interest rating to each provided potential software project topic. We selected six topics that had strong support from multiple students, and held significant interest in both sections. We formed four-person teams2.

Following the announcement of team assignment, the SE4All teams sat down for a brainstorming session, in which they needed to come up with an idea for a software in their particular chosen area. Students in the morning and afternoon sections worked independently of the other section on their informal project proposal documents.

During Week 2, the teams met to discuss their project ideas and to work on product vision statements for their products. The morning SE4All students met their SEIntro partners, while the afternoon SE4All students worked primarily by themselves, and in consultation with the instructor, with the exception of two teams that received a visit from one student from the morning team to discuss integration of ideas. During the ensuing back-and-forth, the morning and the afternoon teams negotiated the scope of each of the projects. At the end of Week 2 morning and afternoon teams submitted their product idea descriptions and their product vision statements.

The CS4All and SEIntro instructors then forced a consensus scope of the project onto each team, making sure to account for ideas from both Section 01 and Section 02 teams on the same project (see Table II).

V. Course Flow and Activities

Table III documents the schedule of topics covered in the SE4All course, and in parallel – in the SEIntro course. The SEIntro course used agile development for the course project. The SE4All students were integrated into early sprints in the course. To properly prepare the SE4All students to their meetings with their SEIntro counterparts we adopted a just-in-time approach to the order and the topics of the SE4All lectures. This meant, on occasion, jumping from discussing one topic to discussing another and then returning back to the former topic in order to accommodate the main theme of a specific sprint. Table III contains the nine-week schedule for the courses. Week 10 was used for exams and reports and did not contain any new material3.

The SE4All course started with a brief overview of the software process, concentrating around the idea that software development ≠ coding. We introduced and discussed different non-developer roles on the software projects.

a) Product Vision: The project brainstorming sessions in Weeks 1 and 2 were followed by introduction of the notion of a formal product vision/product vision statement, and the in-class activities for development of a formal product vision statement for their projects. For product vision format we adopted the vision-in-a-box template pioneered by Moore [7] that includes identification of the target audience, their main pain point, the name of the software product and the description of its core functionality, followed by naming of the

---

2With one exception: in Section 01, the Psychology team had three students, while the Music team had five.

3Cal Poly has 10-week quarters, with an additional week for final examinations. In March 2020, Cal Poly campus closed due to COVID-19 pandemic at the end of Week 10. No course activities took place after.
As a depressed 23 year old, I want a safe and anonymous environment to be able to chat with people so that I can feel connected, supported and not alone.

As a student studying abroad, I want to see what others have done in the city I’m studying in so that I can plan my time there accordingly.

As a socially-focused user, I want to be able to find other users and have them as friends, so that I am able to connect with them frequently.

As a person with a mental disorder who wants to improve their mental health, I want my mood to be consistent, so that I can feel good.

As a environmentally aware, contain a function that will inform you how to dispose of waste properly, as well as provide an outlet for you to donate to environmentally beneficial causes.

As a student who wants to make better consumption decisions, I want to be aware of environmentally conscious targets for my purchase decisions in addition to their waste decisions and allows the user to buy products with a more informed viewpoint.

Table I. A sample product vision statement developed by one SE4All student team for their project during Week 2 of the course.

**Fig. 1.** A sample product vision statement developed by one SE4All student team for their project during Week 2 of the course.

For environmentally minded individuals looking to improve or maintain their waste footprint who want a gamified way to make more environmentally conscious decisions, The Environment App is a mobile application that will send notifications to remind you on how to be more environmentally aware, contain a function that will inform you how to dispose of waste properly, as well as provide an outlet for you to donate to environmentally beneficial causes.

Unlike Olio or Oroeco, our product targets the user’s purchase decisions in addition to their waste decisions and allows the user to buy products with a more informed viewpoint.

Main competition, and the description of a key differentiating aspect. A sample of the vision statement developed by the students in the SE4All course can be seen in Figure 1.

b) **Stakeholders.** During Week 3 lecture time, the instructor identified several categories of stakeholders who are part of the software development process, concentrating on the categories that do not require one to be a professional software developer. We discussed the roles the students were assuming in their projects: product owners, subject matter experts, target users, but also quality assurance specialists and product managers.

c) **Personas and User Stories.** During Weeks 3 and 4 we discussed user personas and the core principles of the their design. Students worked on developing user personas for their projects. In Week 4 we covered user stories, and engaged students in a story-writing workshop during which students engaged in a series of divergence-synthesis design thinking exercises. We chose to use the template to express the user stories. Table IV contains examples of several user stories for different teams in the course.

d) **UI prototyping:** At the end of Week 4, to accommodate the SEIntro sprint schedule, we discussed UI design and prototyping as the means of capturing requirements. Students developed prototype UIs for their project.

e) **Use cases:** In the second half of the course we concentrated on capturing requirements in as formal form as is possible. The starting point was the study of use cases. We discussed the nature of use cases, and the process of building use cases.

f) **Data for applications:** During Weeks 7 and 8 we discussed how to describe the data for a software application. We introduced the notions of an entity set and a relationship set and discussed capturing the data needs of an application as a simplified Entity-Relationship model[8]. Students produced data description documents that included their E-R models of the project data needs.

g) **Functional requirements:** During Week 8 of the course, we discussed formal requirements. While the conversation, and the followup project assignment concentrated on non-functional requirements (see below), some attention was devoted to functional requirements and their capture. Students were assigned writing homework to work on in pairs in which they translated one of the use cases for their application into a set of formal functional requirements. In the conversation about formal requirements, the SE4All course instructor stressed that requirements are finalized by requirements engineers, not product owners or subject matter experts, but at the same time, stressed that understanding and reading existing functional requirements is a useful skill.

h) **Non-Functional requirements:** Whereas functional requirements were discussed in the context of recognition, non-functional requirements were discussed in more detail. We introduced the notion of non-functional (quality) requirements, and discussed a wide range of different categories of NFRs, as well as the means of expressing. Students worked on identifying categories of non-functional requirements for their projects, as well as on finding appropriate ways of specifying individual NFRs.

VI. Preliminary Outcomes

Before the course started we conducted a survey of the students enrolled (at that time) in the course and received 49 responses. We asked students why they enrolled in the class. Around 40% said that they needed a GE course in...
mathematical sciences, and chose our class because it looked interesting. Another 35% responded that they are either interested in how software is built or in the societal impact of computing. About 8% said that they had their own ideas for software they would like to see built. Another 8% were fascinated by the "Computers for Poets" concept, but chose to stay. Finally, 6% were looking for any GE course to fit their schedule. Additionally, 60% of students who responded to our survey never had a CS course, 23.5% had a course in college, and 17.6% had a CS class in high school.

We conducted a mid-term self-assessment among the students in both SE4All and SEIntro courses. This was conducted as an on-line survey completely anonymous, optional, and not tied to any graded component in the course. From both sections of the SE4All course, 36 out of 48 students (18 students from each section) participated in the self-assessment. 18 out of 35 students in the SEIntro course answered the survey.

SE4All students were asked five self-assessment questions. We summarize the results in Figure 2. For four out of five questions the SE4All students assess their ability as at least satisfactory, with the large majority of students answering "good" or "excellent". Students are less certain of their ability to properly describe data needs for a software product, with 10 out of 36 students opting for "poor" or "fair", and only three students selecting "excellent". These responses are in full concordance with the grades student teams received on project-related assignments targeting each type of the artifact. Table V lists average team scores for each project assignment for each section. All assignments but the data description one resulted in fairly high overall grades, with teams generally fulfilling the expectations of the assignment. The data description assignment was the only assignment in the course with an explicit "redo" follow-up (average team redo scores were 84% and 86% for Section 01 and 02 respectively, putting them in line with other assignments) in which students had to fix their data description document (mostly the E-R model part of it) in accordance with instructor’s feedback.

In Table V we show the breakdown of answers to these questions by course section. We see that students from SE4All Section 01, in general, had more confidence in rating their abilities in all tasks, when compared to Section 02 students. At the same time, with the exception of the personas assignment, Section 01 and Section 02 scores were largely in line with each other, and in the personas assignment, Section 02 students actually did better than Section 02 students.

We believe that these results provide some anecdotal support that direct communication with the SEIntro course students is beneficial to SE4All students. The two sections showed relatively little differentiation in project assignment scores (except for the abovementioned exception in favor of Section 02), as well as in quiz and exam scores. Additional evidence that interactions with SEIntro students were productive can be found in the presented results and documenting the SE4All Section 01 and SEIntro students’ opinions on their in-person interactions (Figure 3), and the SE4All students perception of their ability to work with software developers (Figure 4 ). We see that SE4All and SEIntro student perceptions on their interactions are largely in alignment, and that Section 01 SE4All students are somewhat more confident in their ability to work with software developers.
The COVID-19 outbreak at the end of the Winter 2020 quarter caused a change in our research and assessment plans. We chose not to conduct an end-of-course surveys. However, we do have data we collected from the SE4All course by the end of the course, which is summarized in Figure 5. The responses come from the self evaluation questionnaire students filled out as part of the final course activity. Generally, student responses range from Strongly Agree to Agree, with a very few number of responses on the other side of the spectrum. We do not claim this is due to the in-concert mode, but we can at least see some preliminary evidence that our approach did not affect student performance and attitude negatively in the course. On the contrary, there might be a potential association between our method and a more positive student learning that needs to be further investigated.

We observed excellent student engagement - both within their own teams, as well as (for Section 01) with their peers in SEIntro course. The self-assessment surveys show that students in both courses remained positive throughout the term in their interactions with each other. SE4All students contributed in meaningful ways to software development projects.

Our second line of investigation relates to a more formal assessment of the outcomes of these two courses. Owing to our research questions, our key hypotheses are (a) that an SE4All course helps educate better software project stakeholders, and (b) that in-concert teaching of SEIntro and SE4All courses provides a key benefit to the SE students in the former course are yet to be properly evaluated. Our plans to conduct an in-depth exist assessment of the Winter 2020 experience were cut short by the COVID-19-related disruption. At the same time, in two upcoming academic years we plan to study both the SEIntro courses taught in isolation, as well as at least one additional SE4All/SEIntro in-concert combination.

ACKNOWLEDGMENT

The work of the first author is in part supported by an NSF CUE Award 1935108 and an NSF HDR Award 1924008. This project would not have been possible without great support and encouragement of Anonymous Colleague, who provided multiple important suggestions on how to organize the SE4All course.

REFERENCES