Teaching Bioinformarics in Concert: an Interdisciplinary Collaborative Project-based Experience

Alex Dekhtyar, Anya L. Goodman, Aldrin Montana
Department of Computer Science, Cal Poly San Luis Obispo, dekhtyar@calpoly.edu/ Department of Chemistry and Biochemistry. Cal Poly San Luis Obispo, agoodman@calpoly.edu/ Department of Computer Science, Cal Poly San Luis Obispo, amontana@calpoly.edu

Abstract

In the Spring of 2012 we piloted a novel approach to interdisciplinary instruction in the area of bioinformatics that enables undergraduate students in life sciences to work *"in concert"* with computer science students to solve biological problems. Our approach relies on well-defined **interdependent** roles for biology (BIO) and computer science (CS) students in a project-based laboratory.

We recognize distinct learning objectives for each major and implement them in two separate courses taught side-by-side: Bioinformatics Applications for BIO majors and Bioinformatics Algorithms for CS majors. We rely on separate lectures for each group of students, but in laboratory we form joint interdisciplinary teams to work on building software for solving specific biological problems. The teams rely on the biological expertise of BIO students and the software development skills of CS students to produce the software and to use it to obtain requested results. For each assignment, BIO students developed a set of software requirements for a computational biology question, provided it to the CS students on their team, and participated in design and testing of the software as it was being built.

In this paper we present the results of our pilot offering of the two courses to 24 BIO and 35 CS students. We collected and evaluated a variety of student artifacts and conducted extensive surveys in both courses. We discovered that both BIO and CS students indicate improvement in the quality of work of their partners over the course of the quarter. The majority of students reported increased confidence in their ability to collaborate with colleagues outside of their discipline. We discuss these and other findings and present our plans for improvement of our approach for the Spring 2013 offering.

1. Introduction

Beyond higher education, computer science has always been an interdisciplinary field. The vast majority of software serves purposes outside of the pure field of computing, and thus, building software has always involved software developers collaborating with customers who came from a wide range of fields.

At the same time, this interdisciplinary nature of the field, best expressed as "we build software for everyone's needs", finds scant reflection in computer science education. Traditional

undergraduate computer science (CS) curriculum emphasizes technical proficiency, problem-solving skills, and breadth within the field of computer science, leaving learning about the interdisciplinary nature of the field mostly off the required experience. Senior projects, capstone experiences and occasional project-driven software engineering courses provide the only opportunities to experience cross-disciplinary collaboration within CS curricula.

Among the variety of possible reasons why systematic treatment of the interdisciplinary nature of computer science is hard to find in college, *while its very importance is well-recognized by the academic community*[3], one stands very clear. Courses that provide meaningful interdisciplinary experiences for students are *hard to develop* and are *hard to teach.* While specific perception of what constitutes a "meaningful interdisciplinary experience" may vary from person to person and from department to department, in this paper, we view as "meaningful" experience, an experience students get in a regularly taught course that

1. features exposure to computational (and/or software engineering) problems in a field of study outside of computer science/software engineering/computer engineering;
2. involves students and experts (faculty, professionals from outside) from both computer science and the other discipline;
3. provides some form of a hands-on software development experience that includes participants from more than one field.

The two key challenges with developing and teaching courses that satisfy these conditions are (i) the need to attract course participants from outside of computer science and (ii) the need for the course instructor to possess some level of expertise in the second field.

These challenges can be specifically observed when bioinformatics, one of the most popular emerging cross-disciplinary fields involving computer science, is considered. As a field of professional activity, bioinformatics requires in-depth knowledge of both computer science and biology – something that is impossible to achieve in the confines of a single course. Yet, many computer science programs simply do not possess the requisite personnel (i.e., enough computer science faculty with expertise in life sciences in general and/or bioinformatics specifically) to develop and offer a full-scale study in bioinformatics, limiting the possible curricular options to a single bioinformatics course offered as a technical elective for interested students. Similar situation is observed in the Biology/Biochemistry programs.

The key difficulty in offering bioinformatics courses as technical electives in the computer science program lies in the choice the instructor has to make. An in-depth course on bioinformatics algorithms requires significant computational background, and is inappropriate for students outside of computing majors (and *very occasional* biology majors with a CS minor). An attempt to develop a computer science bioinformatics course that would bring biology students alongside computer science students to the same room, therefore, has, by necessity, to forego any topics that require prior computational experience. But what if ***both*** technical depth and *bona fide* interdisciplinary experiences are desired?

In 2009, Pevzner and Shamir recognized the difficulty of offering truly cross-disciplinary bioinformatics courses, and posed an educational challenge asking for the means of developing a bioinformatics course that

1. assumes few computational prerequisites;
2. assumes no knowledge of programming;
3. instills in students a meaningful understanding of computational ideas and ensures that they are able to apply them[7].

In response to this challenge, and to the additional challenge of giving students a "meaningful interdisciplinary experience" (as defined above), in Spring 2012 we successfully piloted an approach that relies on ***two instructors***, one from each discipline, teaching ***two different but tightly interconnected bioinformatics courses***, one for each major, collaborating on preparing and teaching the courses. We termed this approach ***in-concert teaching.*** This approach allows students of both majors to take a technical elective course in their discipline, while at the same time actively participating in a full-quarter interdisciplinary experience involving their peers from the other course.

The rest of this paper is organized as follows. In Section 2 we discuss in detail the concept of *in-concert teaching.* In Section 3 we show how this concept was applied to our teaching of two bioinformatics courses. In Section 4 we discuss our pilot evaluation and its results. Finally, Section 5 outlines the improvements we are planning to implement in the two courses in their Spring 2013 offering.

In this paper, our main goal is two-fold: (a) we discuss the impact of *in-concert* teaching on the experience of computer science students and (b) we discuss the specific way in which we approached the "instilling meaningful understanding of computational ideas" challenge of Pevzner and Shamir[7] for BIO majors.

2. Teaching *In-Concert*

In-concert teaching (or teaching *in concert*) is the name we gave to an approach we developed for teaching two concurrent courses presenting two different *discipline-specific* perspectives (from two different disciplines) on a specific topic. The key characteristics of teaching *in-concert* are:

- **Two discipline-specific courses** stressing technical proficiency within the chosen field of study.
- **Instructors from respective fields:** each course is taught by the instructor from the respective program/department.
- **Joint preparation.** The content of both courses is prepared in collaboration between both instructors.
- **Physical and temporal co-location:** the courses are taught on the same schedule, and, at least for a portion of each course, *occur in physically collocated spaces* (e.g., two neighboring classrooms, or two neighboring labs, or, possibly, a single lab), so that students from both classes could pursue…
- **…shared hands-on projects involving teams of students from both classes.** Interdisciplinary student teams are formed from students of both classes. A non-trivial portion of class time is devoted to interdisciplinary team activities.

- **Students as experts in their field.** While working on joint interdisciplinary assignments students from each course assume roles of experts in their field of study.
- **Exposure of students to knowledge from the other discipline** via interdisciplinary team assignments and via occasional *cross-teaching.*

Informally, the idea of *in-concert* teaching is for two instructors from two different programs to jointly prepare the content of two courses, connected at the level of *shared major coursework* that is performed by teams formed of students from both courses using shared time, space and resources. While not a direct requirement for *in-concert* teaching, the concept can be aided nicely by a number of complementary educational techniques, such as

- **Reverse course design:** course development starts with basic concepts instructors are interested in, proceeds with determination of the joint assignments, from which, in turn, the specific theoretical course material and teaching schedules are derived.
- **Inverted classrooms:** can be used in either or both courses to drive both discipline-specific and cross-disciplinary learning.
- **Undergraduate research:** joint assignments can be part of an undergraduate research experience, and can directly contribute to generation of new knowledge in classroom.

Structurally, we can identify three different modes of instruction within the *in-concert* teaching environment:

1. **Discipline-specific instruction:** instruction within each course on the discipline-specific concepts related to the overall subject of the study.
2. **Joint labs:** joint hands-on work on cross-disciplinary assignments.
3. **Cross-teaching:** introduction to the necessary aspects of the other discipline presented to each class by the instructor of the *other* course.

The first teaching mode is the "traditional" course instruction that concentrates on studying parts of the subject that are directly related to the specific discipline of the course. It can be done either as a lecture or as an inverted classroom exercise, but it involves the course instructor interacting only with students from *their* course. The hands-on part of the course brings students from both courses together to work on the joint assignments. Instructors of both courses share the responsibility for working with these teams. Finally, on some occasions, students in each course may need to receive some training/information pertaining to the other discipline. This is done via cross-teaching: instructor of one course coming to present a lecture or conduct an activity in the other course. The presented material is targeted *specifically* at the students from the other discipline and is, as a rule, separate from the discipline-specific lectures/activities/training.

One of the key aspects of *in-concert* teaching is that by splitting an interdisciplinary course into two discipline-specific courses, it allows for course instructors to have minimal expertise in the other discipline. In essence, in preparing and delivering *in-concert* coursework to students, course instructors go through a similar experience of joint collaborative cross-disciplinary work, in which the contribute the knowledge from their field.

3. Teaching Bioinformatics *In Concert*

We piloted our *in-concert* teaching on the base of two courses: CSC 448: Bioinformatics Algorithms taught by the first author, and BIO 441: Bioinformatics Applications taught by the second author. Both courses were already on the books of respective programs, which allowed us to forego new course proposals. The two courses, however, have different back stories. The Bioinformatics Algorithms course was proposed as a computer science technical elective at the turn of the century and has been taught once or twice by a different faculty member, after which due to personnel changes it has essentially became a dormant course. The Bioinformatics Applications course is a technical elective in Biology and Biochemistry programs and prior to Spring 2012 it has been taught on an annual basis.

Both classes come with six contact hours per week: three hours for lecture and three hours for lab, which made physical coordination of the courses straightforward. In what follows, we provide a brief description of the organization of the two courses, course content, joint assignments and the overall course flow.

Logistics. Both courses were offered during the same time periods on a Tuesday-Thursday schedule. Both courses used computer labs run by the department of Computer Science. BIO 441 took place in its entirety in one computer lab. CSC 448 had a lecture set in a classroom elsewhere on campus, with a lab period taking place in a computer lab next door to the one in which BIO 441 took place. The lab for the CS course was unoccupied during the lecture timeslot, so, on occasion, the entirety of CSC 448 happened in the lab, or the class was brought to the lab early.

24 students took the Bioinformatics Applications course, with the majority coming from Biology, Biochemistry and Animal Science majors, although the course was also taken by one Political Science and one Math major. 35 students took Bioinformatics Algorithms course. Most students were Computer Science or Software Engineering majors, with a few Computer Engineering majors and one Physics major.

For the hands-on part of the course we formed 12 teams consisting of two BIO 441 students and three CSC 448 students (except for one team, which only contained two CS students). The BIO pairs and the CS triples were formed by the two instructors independently based on first-day surveys. BIO students were primarily matched by attitude (how hard they expected to work in class). CS students were organized to provide balance, with each team having at least two students who completed the Intro to BIO or Intro to Chemistry sequence, some software engineering experience (having taken a software engineering course required for CS and SE majors) and with at least one student per team having taken the Algorithms course.

Throughout the course, teams wound up having 2-3 hours of face time per week. During most lab periods, each team worked together, with odd-numbered teams occupying one of the two labs and the even-numbered teams residing in the neighboring lab. Each team received its own work space (two sides of a computer lab isle) that allowed the team members to both work independently on their own tasks, as well as join together for a team meeting by simply turning the chairs around, as shown in Figure 1.

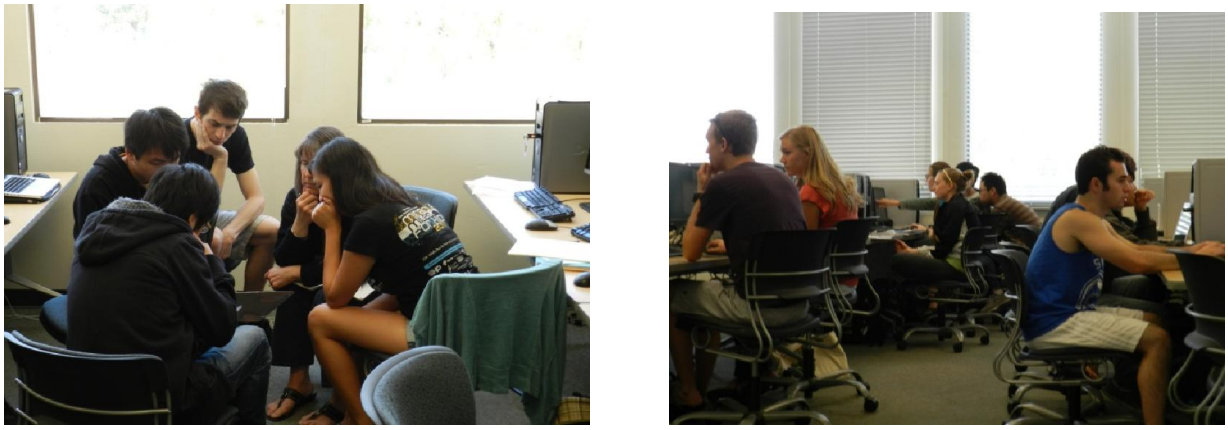


Figure 1. Students work together (left) and separately (right) on their assignments during joint lab periods.

Curricula. The curricula for the two courses have been co-designed by the course instructors in the *reverse course design* fashion. As a starting point, the instructors agreed on the set of learning objectives for each course. The learning objectives are presented in Table 1.

Table 1. Learning objectives for the two Bioinformatics courses.

CSC 448: Bioinformatics Algorithms	BIO 441: Bioinformatics Applications
1. Know main problems in bioinformatics	1. Use and explain bioinformatics concepts/terminology
2. Understand key bioinformatics algorithms	2. Use web-based tools to access gene/genome information
3. Model bioinformatics problems	3. Use scientific method to investigate questions related to gene structure and function
4. Apply algorithmic techniques to solve bioinformatics problems	4. Convert a biological question into a computational problem
5. Effectively communicate and cooperate with colleagues in biology and computer science	

To support the learning objectives of the BIO 441, we used two intertwined course-long assignments based on the work of the Genomics Education Partnership (GEP)[1,6]. GEP is a community of biology faculty who develop and incorporate in undergraduate curriculum research projects related to fruit fly genomes. One of the two course-long assignments in BIO 441 was annotation of genes in the D*rosophila mojavensis* genome, a project that BIO 441 students completed independently of CSC 448 students. The second assignment used two regions of D*rosophila erecta* genome, Chromosome 3 and Chromosome 4. It asked students to conduct a comparative study of the two chromosomes and look for differences in organization of genetic information. The instructors broke the overall assignment into a sequence of individual

tasks that had to be completed by the joint teams of BIO 441 and CSC 448 students. Each task asked BIO students a specific research question, or asked them to prepare the genome data provided to them in a specific way. Either way, the BIO students had to turn their assignment into a formal software requirements specification provided to the CS students. The CS students developed software solving the given problem, deployed it for the use of the BIO students, who then ran it on their data and reported the results.

BIO 441 lectures covered the specifics of genome analysis, required for the students to be able to successfully complete the gene annotation project, and to be able to successfully explain through documentation and conversation the requirements for the software that the CSC 448 had to build.

The content of the Bioinformatics Algorithms course was built to match the specific programming assignments. The topics covered in both courses are listed in Table 2.

Table 2. Topics covered in the two courses.

week	**CSC 448 topic**	**BIO 441 topic**
1	Biology background	Bio and software engineering background
2	DNA analysis (codon bias, %GC, entropy)	DNA sequencing technologies genome sequencing (HGP)
3	String matching	Genome annotation, UCSC genome browser, custom tools
4	Suffix Trees	Sequence comparison: dot plot, local and global alignments
5	Suffix Trees	Genome annotation (practice contig)
6	Repeats and palindromes	Chromatin and gene expression
7	Alignment (Dynamic Programming)	Comparative genomics
8	Alignment (FASTA, BLAST)	Research paper: 12 fly genomes
9	Alignment (FASTA, BLAST)	Research
10	Clustering	Research

The key topics covered in the Bioinformatics Algorithms course were DNA measures (codon bias, GC percent, DNA enthropy), exact string matching algorithms (Knuth-Morris-Pratt, Boyer-Moore) and suffix trees and their uses for string analysis, string alignment using dynamic programming techniques and using approximation techniques (FASTA, BLAST) and clustering. Each topic, with the exception of clustering was tied to a lab assignment, requiring the students to adopt, adapt and implement the algorithms covered in lecture to address a specific problem supplied to them by their team partners from BIO 441.

Labs. Both classes came together for 2-3 hours a week to work joint lab assignments. Starting the first week of classes, where students from both classes participated in ice-breaking activities involving Computer Science Unplugged games[2,4] and joint work on a task of finding and retrieving genome data from on-line databases, and going through the last week of classes, when the teams worked jointly on the oral presentation for BIO students and poster for CS students, all labs targeted *joint work* of students in cross-disciplinary teams (the stable teams were formed on the second week of the classes; the ice-breaking activities of week one were performed by constructing *ad hoc* teams for each lab). Table 3 shows the list of laboratory assignments in the two courses.

Table 3. List of joint lab assignments in CSC 448/BIO 441

Lab	**Name**
0	CS unplugged: Marching Orders
1	Applications: Genes and Traits
2-1	DNA sequence analysis: GC content
2-2	DNA sequence analysis: Codon Usage Bias
3-1	Contig assembly (Overlap, Boyer-Moore)
3-2	Repeat Search, Palindromes (Suffix Trees)
4	Alignment (conflict resolution)
5	Poster (CS)/Presentation (BIO)

The core programming assignments in the course consisted of Labs 2-1, 2-2, 3-1, 3-2 and 4. In terms of technical assignments for CS students, both parts of Lab 2 dealt with building software for measuring different properties of DNA sequences. Lab 3 assignments were reduced to different approaches to exact string matching problems and their extensions, such as palindrome detection. Lab 4 involved implementation of a dynamic programming algorithm for global alignment. Each assignment was structured in a similar way:

1. **Origination:** the BIO 441 instructor explained to BIO 441 students the specific question to be addressed.
2. **Requirements:** BIO 441 students prepared a requirements specification for CS students.
3. **Design:** CSC 448 students obtained the requirements specification from their BIO 441 teammates. The teams worked together on ensuring that CS students understood what software needed to be built. CS students worked (informally, no formal specifications were required) on the design of the solution.
4. **Implementation:** CSC 448 students worked on implementing the software.
5. **Testing:** both CS and BIO students participated in testing of the software. CS students used feedback from BIO students to both fix bugs in the code and to improve the

usability of the software, ensuring the BIO students would be able to run the software by themselves.

6. **Deployment and use.** CS students delivered the final version of the software to the BIO students, who used it on their data to either answer the research questions posed in the lab assignment, or to prepare the data for the next assignment.

Steps 2-6 of this process happened during the joint lab periods. The time frames for different labs varied. Lab 2-1 was designed to be completed in a single lab period -- by the end of the lab period, teams were expected to produce a simple but usable program computing GC percent in an input DNA fragment. Other labs usually had a two-week window, although some deadline slippage occurred: in the second half of the course teams often worked on multiple lab assignments in parallel – testing and maintaining/debugging code from an older assignment, while building requirements and design for a newer one.

For Step 4, when CS students were fully engaged in activities (coding) where their BIO partners were largely not needed, BIO students worked on their genome annotation assignments, while staying in close proximity of their CS partners, in case consultations were required (see Figure 1, right).

While CS students got detailed instructions from the course instructor concerning the specifics of work organization and deliverables for each assignment, as well as hints as to what algorithms needed to be implemented, the actual nature of assignment was conveyed to the CS students by their BIO partners. We felt that it was important to engage students from both courses in direct communication as soon as the assignments came out, and we felt that BIO students explaining the nature of the assignment, and why they needed the software to complete their tasks added an important wrinkle to the overall experience of both BIO and CS students. The former got to experience the role of experts in their field. The latter obtained an important experience of knowledge engineering from non-technical customers.

Cross-teaching. At the beginning of the quarter the CSC 448 instructor gave a guest lecture to the BIO 441 students describing the software development process, and outlining the importance of each of the steps in the process. To illustrate the process, the V model of software development[5] was used, in which the steps of the software development process are plotted in the form of the letter "V". Five stages were discussed: *Requirements, Design* (the left "arm" of the "V"), *Implementation* (the "point" of the "V"), *Testing* and *Deployment/Maintenance* (the right "arm" of the "V"). A special emphasis was given to the *Requirements* stage.

To help BIO 441 students prepare requirements for the lab assignments, the course instructors developed a simple requirements template. The template listed different types of requirements (functional requirements, non-functional requirements, design constraints, process constraints). Functional requirements were broken into three categories: input specification, output specification and processing instructions. BIO 441 were asked to fill the template with specific requirements for each joint lab assignment.

While no additional formal cross-teaching occurred, throughout the quarter the CSC 448 instructor interacted with BIO 441 students as a group on several occasions, discussing specifics

of some of the assignments. Additionally, the Teaching Assistant for CSC 448 (the third author) was present for all lab periods and worked actively with individual teams to bridge the gaps in communication whenever they arose.

Overall course flow. Both courses were successful in covering expected theoretical material in the time frame of the course. In CSC 448, the last week was originally planned for informal discussions of various bioinformatics problems not covered elsewhere in the course, but due to some lecture slippage, it was used to cover clustering as applied to multiple sequence alignment.

Fairly early in the course, we realized that strict code delivery deadlines were not enforceable, as a wide-range of post-deployment updates had to be made to the code by request of the BIO students. We adjusted our expectations, by introducing new assignments on original schedule, but allowing work on prior assignments to continue for a number of weeks after the initial deadline. For grading purposes, a deliverable from each team for each assignment was collected at some specific point, but *in an important distinction of this course from most other computer science courses, students continued working on the programs even after they were collected for grading,* as the final deliverables of their BIO partners were due only at the end of the quarter. The need to work on multiple software products at the same time added a level of stress (and certainly found its way into student comments), but it also added a touch of reality of working in a professional environment, where multiple tasks need to be juggled at the same time.

4. Evaluating the efficacy of *in-concert* teaching

We viewed the Spring 2012 offering as a pilot, designed to test the concept of *in-concert teaching* as applied to bioinformatics courses, and to illuminate our further efforts on improving these courses. To that extent, we were specifically interested in answers to the following questions:

1. *Are the students meeting the learning objectives of the courses?* Specifically, are the students acquiring the technical knowledge and skills taught in the courses (learning objectives 1-4 for CSC 448 and 1-3 for BIO 441).
2. *Are life sciences students acquiring computational skills?* The key computational skill presented in BIO 441 was the ability to convert a biological problem into a set of software requirements (learning objective 4 for BIO 441).
3. *Are students learning to work effectively with their peers within and outside of their discipline?* In particular, is there evidence that cross-disciplinary collaboration within the *in-concert teaching* framework is beneficial for the students? (learning objective 5 for both courses).
4. *How did students perceive in-concert teaching format?* Subjective satisfaction of students with their course may either help or significantly hamper their success with the course, often regardless of how well the students are actually meeting the course learning objectives. The unusual format for the course presented some challenges to the students. It was important for us to understand what the students thought about the class format, and what, in their opinions, worked and did not work.

In the rest of this section we provide a brief overview of our efforts to evaluate, both objectively and subjectively, the answers to these questions. Except for question 2 which concerns primarily students from BIO 441, we stress the evaluation results obtained from the CSC 448 students.

4.1. Evaluation Overview

Our key evaluation instruments are three-fold. To evaluate overall student performance in the course – both for individual students and for student teams we use coursework grades. To evaluate the cross-disciplinary experience we conducted an exit survey for all CSC 448 and BIO 441 students in which we asked them to evaluate their work and the work on their peers on team assignments. Additionally, we asked open-ended questions about individual experiences in the course and about the major take home lessons from the course. Below we provide an overview of the observed results as they apply to the four questions specified above.

Question 1: Technical proficiency of CSC 448 students. Our only instruments in assessing the first four CSC 448 learning objectives were grades for various coursework. CSC 448 had paper-and-pencil midterm and final exams. The exams consisted of a variety of questions asking students to (a) apply algorithms covered in the course to solving instances of bioinformatics problems, (b) modify studied algorithms and techniques to adapt them to solving a specific bioinformatics problem and (c) craft simple multi-step solutions to bioinformatics problems out of existing "pieces". Evaluation of individual CSC 448 learning objectives 1-4 on the basis of a specific problem or a set of problems from the exams is difficult, as the problems often combined elements of modeling and application. Instead, we group the objectives 1-4 into a single goal of "achieving technical proficiency with algorithms for solving bioinformatics problems." We use the exam scores to evaluate the overall technical proficiency of individual students in the course.

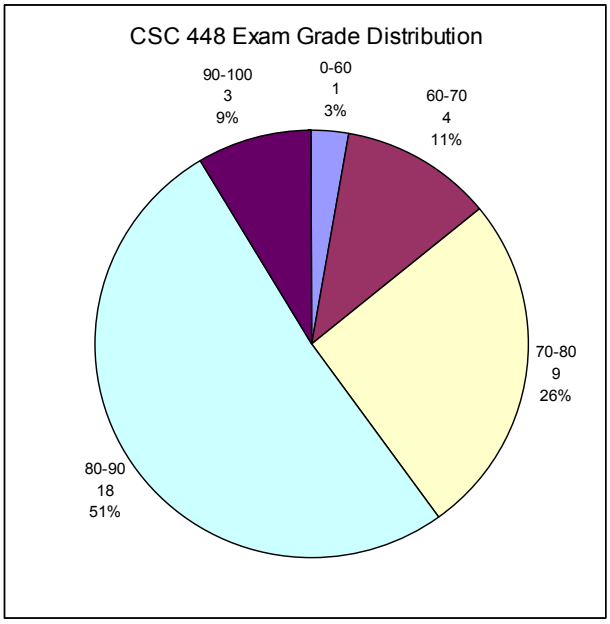


Figure 2. Distribution of normalized combined exam scores in CSC 448.

Two exams were administered: a 90-minute midterm and a three-hour final exam. The midterm was out of 75 points and the final exam was out of 100 points, with points on each exam having the same weight. Together, the two exams accounted for 50% of the course grade. The midterm exam, thus accounted for about 43% of the combined exam score or about 21.5% of the course grade, while the final exam accounted for about 57% of the combined score or about 28.5% of the course grade.

Figure 2 shows the distribution of the combined scores for the students in CSC 448 normalized to 100. An "A" range for the two exams was considered to be a normalized score of 80% or above. As seen from the chart, 60% of students (21 individuals) received a score of over 80%. An additional 26% had a score of 70-80% corresponding to a grade of "B". Four students were in the "C" range (60-70), and one student had a score lower than 60%. A total of 86% of students in the course, therefore, displayed good or excellent technical proficiency as measured by the problem-solving on the exams.

Question 2. Can BIO students write good requirements? In the context of our courses, learning objective 4 of BIO 441 was met by introducing the notion of software lifecycle into BIO 441, and by giving students hands-on experience with participation in the software development process on almost all its stages. The computational skills we wanted to teach BIO students are the skills associated with development of proper software specifications. Throughout the course, BIO 441 students were required to produce five complete and formal requirements documents, one for each programming lab assignment (see Table 3).

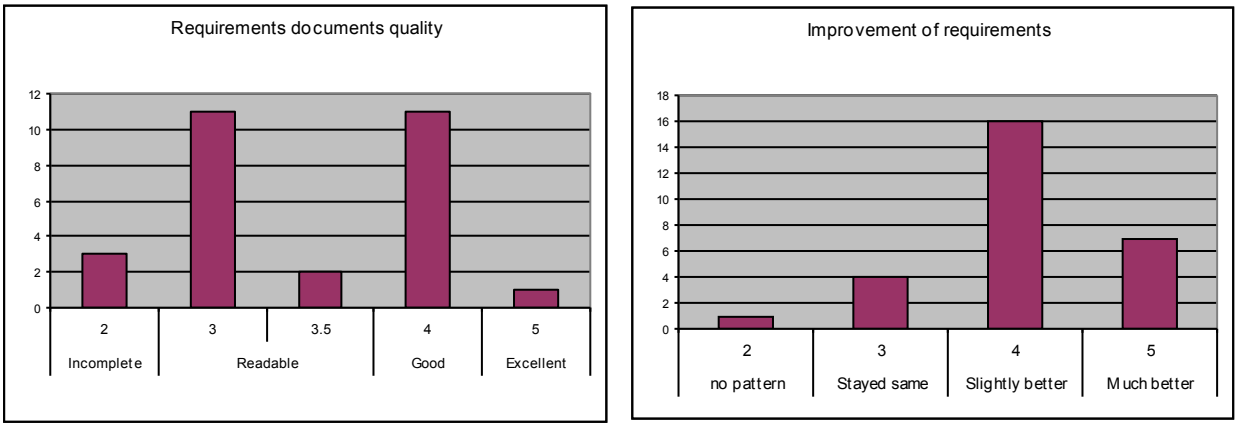
We used the exit survey to evaluate this question. The exit survey was administered to both courses. The survey asked students a number of open-ended and closed-form (multiple choice or scaled answer) questions about what they considered to be their most important achievements in the course, about their experiences on the cross-disciplinary team, and about their advice for the future course offerings. Some questions were different on the CSC 448 and CSC 441 surveys, asking students to look at the same thing from their specific perspectives. Some questions persisted across the course boundaries.

28 out of 35 CSC 448 students and 23 out of 24 BIO 441 students took the survey. At least one person from each team participated in the BIO 441 surveys. In the CSC 448 surveys, 11 out of 12 teams were represented by at least one student, with 10 of them represented by at least two. Because we wanted to link responses from students in each team together, the survey *was not* anonymous.

The survey included two questions for evaluating the quality of requirements documents written by BIO 441 students. The CSC 448 survey's questions were:

1. *How would you rate (overall) the requirements documents provided to you be your BIO 441 partners?*
   We used the 0-5 scale for the question with 0 being "requirements documents completely lacking," and 5 being "excellent."
2. *Did the requirements documents get better as the quarter proceeded?*

We used the 0-5 scale for the question, with 0 being "no basis for judgment", 1 – "document actually decreased in quality", 2 – "no clear pattern", 3 – "quality stayed the same", 4 – "slight improvement" and 5- "got much better".



(a) (b)

Figure 3. Peer assessment of quality of requirements documents (a) and the level of improvement of the requirements documents over the course of the quarter (b). CSC 448 students assessing BIO 441 students.

Figure 3.a shows the summary of the responses to the first question. Only three CSC 448 students stated that requirements documents provided to them throughout the quarter were incomplete and otherwise lacking in quality. The vast majority of the class split evenly between the requirements documents being "readable" and usable, and between the requirements documents being "good". The average score on this question was 3.4 with the standard deviation of 0.72.

Figure 3.b shows the summary of the responses to the second question. Only one student indicated that the quality of the requirements documents fluctuated over the course of the quarter (incidentally, this was also one of the students who gave a score of 2 for the previous question). Four students stated that the quality stayed the same. The remaining students indicated an improvement of the quality of the requirements documents over time: 18 students indicated slight improvement, while seven students indicated significant improvement. Overall, 82% of CSC respondents indicated that the quality of the requirements improved. This aligns well with the self-assessment of BIO 441 students: 86% of them, in response to a different survey question indicated that their ability to write program requirements improved over time.

In drawing our conclusions about the efficacy of biology majors translating bioinformatics programs into formal software requirements specifications we rely on the opinion of their computer science partners. There are two possible threats to validity that may affect our overall evaluation. First, CS students are asked to evaluate their peers with whom they worked closely for the duration of the quarter. As such, they may be uncomfortable reporting their real opinions, especially in the situation where we know whose requirements documents they are supposed to evaluate. To a large degree the scores provided by the CS students reflect their level of comfort working with their BIO partners as well as their assessment of the requirements documents. We

are comfortable with this evaluation for the pilot study, but will use more objective instruments for evaluation in the future. The second threat to validity comes from the fact that students are unreliable judges of quality of software engineering artifacts and that the grading scales of individual students may have been different – i.e. two different students assigned different numeric scores, while holding the same qualitative opinion of the requirements documents. This threat is present in any assessment that relies on peer evaluation though. In our case, we view student-assigned scores as the evidence of their personal perception of the difficulty of understanding the requirements documents provided.

Question 3: How well did the teams work together? The exit survey contained a group of questions designed to gauge the participation of BIO 441 students in the software development process and the level of their engagement. CSC 448 students were asked:

- *For each step of the software development process shown below, indicate the extent to which **your BIO 441** partners contributed.*

BIO 441 students were asked:

- *For each step of the software development process shown below, indicate the extent to which **you and your partner from BIO 441** contributed.*
-

Both classes were provided with the same list of steps: *Requirements, Design, Development, Testing, Deployment, Maintenance.* The response scale was 0 – 5, described as shown in Table 4 below:

Table 4. Response scale for the question about level of participation of BIO 441 students in the software development process.

Value	Meaning
0	Did not contribute at all
1	Contributed very little
2	Minor contributions
3	Significant contributions
4	Same level of contribution as CSC 448 partners
5	Were team leaders

Student responses are summarized in Figure 4 and Table 5. As seen from the collected data, students from both courses agreed that there was a distinct contribution from BIO 441 students in the requirements and testing stages of the lab assignments. CSC 448 students reported diminished contributions on the design stage. Deployment stage was characterized by large diversity involvement of BIO 441 students – they were making large contributions on some teams, but not contributing on others (hence the largest standard deviation of all stages). Most CSC 448 teams agreed that BIO 441 student contributions on the development and maintenance stages were not very significant.

Compared to their CSC 448 peers, BIO 441 students tended to be more optimistic about the value of their contribution to the joint work, overstating their contributions by about one point on the 0-5 point scale (the average difference between CSC 448 and BIO 441 evaluations is 0.96).

This bias was expected. The differences of opinion were smaller on the stages where BIO 441 students participated more actively, and larger on the stages where their participation was not as widely acknowledged. Only one stage, deployment, showed a significant disagreement between the two courses.

Overall, the results paint a clear picture of how the collaboration process worked, and the picture aligns well with our expectations. As expected, the core interactions in the teams happened at the beginning of each lab, while requirements were discussed. We expected the contributions of BIO 441 students to be minimized during the design and development stages, and were hoping that they would come back for testing. Our hopes found cautious justification in the student responses, as we saw the contributions of BIO 441 students increase significantly at that stage. From there on, different teams chose different paths to proceed, with some teams continuing to work tightly on the deployment and maintenance of the programs, while other teams essentially not having significant post-delivery interactions concerning the software.

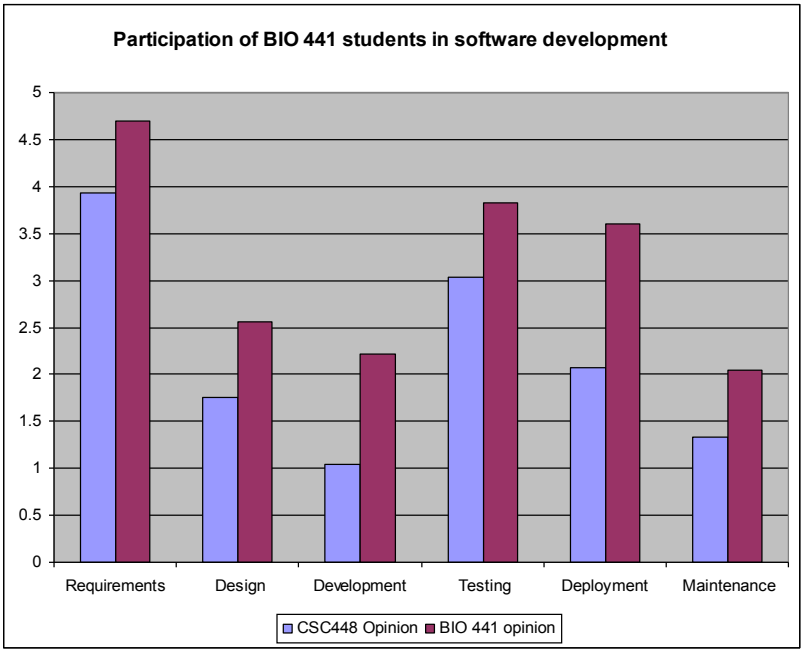


Figure 4. Participation of BIO 441 students in different stages of the software development process, as reported by CSC 448 students and BIO 441 students themselves.

Additional survey questions were used to assess student perception of the outcomes of their collaborative work. A pair of questions asked the students to specify if their confidence in their ability to work effectively with colleagues in their discipline and outside of their discipline has increased. Table 6 shows the information about the percentage of respondents in each class who indicated that their confidence has either increased somewhat or increased significantly. As seen from the table, solid majorities in both classes (but especially in BIO 441) indicated that they are more confident in their ability to work with people from outside of their discipline.

Table 5. Participation of BIO 441 students in different stages of the software development process, as reported by CSC 448 students and BIO 441 students themselves.

Stage	CSC 448		BIO 441		Difference
	Mean	St.Dev	Mean	St.Dev	
Requirements	3.92	1.05	4.7	0.63	0.77
Design	1.75	1.26	2.56	1.31	0.815
Development	1.03	1.29	2.22	1.25	1.18
Testing	3.03	1.48	3.82	0.82	0.79
Deployment	2.07	1.89	3.6	1.09	1.528
Maintenance	1.33	1.38	2.04	1.46	1.46

Table 6. Student confidence in their ability to work with colleagues.

Increased confidence in ability to work with colleagues	BIO 441 students	CSC 448 students
In own discipline	62%	75%
Outside of own discipline	86%	68%

One more survey question asked *"What are the three most important things you learned in this course?"* The most popular responses to this question for the students from CSC 448 and BIO 441 are summarized in Tables 7.a and 7.b respectively.

Table 7. Most popular responses about the most important things learned in the class for CSC 448 (a) and BIO 441 (b) students.

(a)CSC 448

Response	**Number (out of 28)**	**Percent**
Algorithms	16	57%
Work with non-CS teammates	14	50%
Teamwork	8	25%
Suffix trees	6	21%
Communication	5	18%

(b) BIO 441

Response	**Number (out of 23)**	**Percent**
Genome annotation	18	78%
Work with CS teammates	12	52%
Concepts related to genes	11	48%
Software requirements	5	23%
Research/data analysis	4	17%

Student responses in both classes paint a clear picture. In both courses, technical knowledge in own discipline – bioinformatics algorithms, and genome annotation and the science of genomics, were the most popular answers. However, work with teammates outside of their discipline came in second in both courses, with about half of students in each class naming *explicitly* not just the teamwork aspect, but also the *collaboration outside the discipline* aspect. In CSC 448, an additional 13 responses mentioned teamwork and communication, without specifically addressing cross-disciplinary collaboration. When combining student responses, 23 out of 28 CSC 448 students (82%) mentioned algorithms (generic or specific) in their response. What is interesting, is that almost the same number, 22 students (78.5%) mentioned at least one thing that alluded to Learning Objective 5 (collaborative work). We conclude from this data that students

in both classes, but especially CSC 448 students found their experience of working on a cross-disciplinary team throughout the entirety of the course to be its key component.

Question 4: What did the students think about their experience? A number of questions on the survey were designed to elicit open-ended responses. We asked students to describe the benefits of working on cross-disciplinary teams, explain what challenges they encountered in coursework and how they overcame the challenges, and let us know what they liked the most about the course and what could be improved. A representative sample of comments by the CSC 448 and BIO 441 students is included in Appendix A.

A reading of the comments of CSC 448 students revealed, that almost uniformly, regardless of their prior expectations about the course, and regardless of the actual achievement of their teams on joint assignments, and specific experiences with their teams, *students understood the importance of working with peers from other disciplines* and *appreciated the opportunity to do so throughout the quarter*. **Some of the most encouraging and positive comments came from students whose teams struggled the most throughout the quarter.** Course critique and suggestions for improvement were, for the most part, well thought-out and based on actual in-class experiences.

5. Lessons Learned and Future Improvements

As evidenced by the student comments (see Appendix A), teaching bioinformatics *in-concert* was challenging, and not everything in the course went smoothly. Below we list and discuss some of the challenges we and the students encountered.

Heterogeneity of student body. The prerequisites for both courses were historically kept relatively low: CSC 448 required CS III (Data Structures) as the only prerequisite, while BIO 441 required only an introduction to biology course. In BIO 441 it resulted in a number of students from outside of life sciences majors taking the course. In CSC 448, some students taking the course took neither algorithms nor software engineering coursework. Some Computer Engineering majors took this class after almost a year-long break from CS coursework, which affected their programming skills. This meant that the experiences and the knowledge of some of the students in each course were in conflict with the key idea behind *in-concert* lab assignments: that students from each class will play the roles of experts in their discipline. In a small number of teams, this issue existed on both sides, and these teams experienced more difficulties in organizing software development process.

CS lab deliverables. In response to concerns of some students, expressed prior to the beginning of the course, that their grade may depend fully on students from another course, CSC 448 lab assignments were designed with two deliverables in mind: a deliverable for BIO 441 students, which had to run and produce results needed for the research project, and a deliverable for CSC 448, which typically included a generic implementation of an algorithm or data structure discussed in the lectures (and was supposed to be the basis of the BIO 441 deliverable). The lab grade was based on both deliverables.

However, it turned out that the fears of grade dependence on outsiders were *premature* – no student in the class ever complained about it. At the same time, the dual deliverable policy caused a lot of confusion among the students and resulted in significant deadline slippage throughout the course. One of the strongest suggestions received from CSC 448 students was to require only the BIO 441 deliverable.

Deadline slippage. Perhaps the biggest lesson learned is that software is never complete. BIO 441 students continued using the code developed by CSC 448 students throughout the entire quarter, which meant that even the second-week assignment, a simple GC% computation program was still in active use around week 9 of the course. Additionally, as more pieces of the course-long project were developed, revisions to older programs were requested to make the inputs/outputs of deliverables from different labs match. This meant that by the second half of the quarter, each team was working on different stages of 3-4 programs at the same time. This, and the abovementioned confusion about the nature of some deliverables led to teams being unable to complete initial code delivery for labs on the original deadline dates.

Lack of cross-teaching. Two more common comments we received from students indicated that BIO 441 students had trouble writing requirements that properly reflected their assignment, and CSC 448 students struggled at times to understand the nature of the assignment itself. We identified lack of cross-teaching as the underlying reason for both of these comments. BIO 441 students received some instruction on software development process and requirements at the beginning of the course, but no further cross-teaching interactions occurred. Similarly, CSC 448 students only communicated with their team partners on the matters related to the specific nature of the assignments, and whenever their partners struggled to explain the assignments, confusion arose.

At the same time, even with the significant challenges described above, we believe that the courses largely served their purpose. We showed that *in-concert* teaching format is feasible and is accepted by students. We observed student achievement with respect to every learning objective in both classes, and we specifically observed significant achievement associated with the new learning objectives in BIO 441 (writing requirements, working with CS peers) and the matching learning objective in CSC 448. Students *bought into the concept of the courses* even when their individual experience was imperfect.

The next offering of both classes is scheduled for Spring 2013 quarter. We plan to organize the courses logistically in the same way, and admit 12 five-person teams worth of students in both classes (24 BIO + 36 CS). We are actively revising some aspects of the courses to take into account the issues discussed above. In particular, the following course changes and improvements are planned:

- **Tighter enrollment control.** We plan to limit enrollment in BIO 441 to life sciences majors with significant biology background to ensure that they can successfully assume the role of experts in their field. Similarly, we plan to raise expectations for computing sciences majors enrolling in CSC 448.

- **Streamlined lab assignments.** All lab assignments will have only one set of software deliverables – the programs required by BIO 441 students. We will establish a two-tier submission process, where an early version is submitted on a short deadline, but the final version of each program is due during last two weeks of the quarter. Grading policies will be adjusted accordingly.

- **More cross-teaching.** A series of lectures on software engineering for BIO 441 students by the CSC 448 instructor is planned. We expect to have 30 minutes to 1 hour of cross-teaching time per week for the first 6-7 weeks of the quarter. The time will be used to discuss software development process in more detail, provide hands-on experiences with requirements specification and test case development, as well as address any questions BIO 441 student might have about the labs and collaboration with CSC 448 students. We plan to adjust the lab assignments so that each subsequent assignment emphasizes the participation of BIO 441 students in one of the steps of the software development process: requirements, design (data modeling), testing, deployment and maintenance. During the same time, BIO 441 instructor will conduct more in-depth discussions of biological topics with CSC 448 students.

With the commitment of both Biology and Biochemistry and Computer Science programs at Cal Poly, the *in-concert teaching* of bioinformatics is slated to become an annual staple.

Acknowledgements
This work is supported in part by a grant from the W.M. Keck foundation.

Bibliography

1. Genomics Education Partnership, http://gep.wustl.edu/.
2. Computer Science Unplugged, http://csunplugged.org/.
3. G.A. Amoussou, M. Boylan, Joan Peckham, Interdisciplinary Computing Education for the Challenges of the Future, in *Proceedings, ACM SIGCSE' 2010,* pp. 556-557, 2010.
4. M. Fellows, M., T. Bell (2010) *Computer Science Unplugged.*
5. K. Forsberg and H. Mooz, The Relationship of System Engineering to the Project Cycle, in Proceedings *of the First Annual Symposium of National Council on System Engineering*, October 1991: 57–65.
6. Lopatto D, Alvarez C, Barnard D, Chandrasekaran C, Chung H-M, Du C, Eckdahl T, Goodman AL, Hauser C, Jones CJ, Kopp OR, Kuleck GA, McNeil G, Morris R, Myka JL, Nagengast A, Overvoorde PJ, Poet JL, Reed K, Regisford G, Revie D, Rosenwald A, Saville K, Shaw M, Skuse GR, Smith C, Smith M, Spratt M, Stamm J, Thompson JS, Wilson BA, Witkowski C, Youngblom J, Leung W, Shaffer CD, Buhler J, Mardis E, Elgin SCR., Education Forum: Genomics Education Partnership, 2008, *Science 322, 684-5*
7. Pevzner P, Shamir R., Computing has changed biology--biology education must catch up, *Science. 2009 Jul 31;325(5940):541-2*.

Appendix A.

We include a sample of comments made by students from CSC 448 and BIO 441 in response to open-ended questions in the exit survey.

CSC 448.
Question: *What did you like best about the course?*

"Working on meaningful code that another discipline can use for research. It was rewarding and a valuable experience."
"The course is fast paced. I felt like I learned a good amount."
"The way the lab period was structured…"
"I liked that we got to work in teams with non-CSC majors. It was a fun experience."
"Having a real research project."
"Writing programs that actually had practical use."
"Multi-discipline [sic] groups… Real-world experience."
"It was fun. I really enjoy group work and that is all this was."
"I liked that we were doing something with significance to a developing field."

Question(s): *What can be improved in the course and how could it be improved? What can the instructors of both courses do to help students overcome the challenges better?*

"It was hard to complete some projects on time …"
"Spend a little more time so the BIO students know what they need/want better. Program requirements were often incomplete or wrong from what they actually want [sic]. Obviously, this can never be completely avoided, but it can be reduced."
"Work towards a single 'tool'."
"Reduce complexity of some labs."
"Help CHEM students understand how to explain problems more clearly…"
"Proofread requirements. Allow time … for BIO students to test the program."
"Testing… Sometimes it's hard for CS students to see a problem with data."
"Make it perfectly clear what needs to be turned in [for BIO 441] and what for [CSC 448]."
"…the flow of communication was the biggest issue."
"More preparation of test cases for CS student to use during development."
"Require more communication before building software."
"Meh, suffering is how we all learn best."

Question: *"What were the benefits of working with partners from a different discipline?"*

"You do that in the real industry so you get a head start. You get a break from all-CS partner teams."
"They could help fill in the blanks when we did not understand what they needed."
"You learn to really work with subpar or unclear prog[ram] requirements."
"Understand how to handle knowledge gaps/barriers."
"Learned how to get requirements from people who don't understand them."
"Got to work on programs that go towards real use. Meet people outside the major."
"We get their domain knowledge and they make decisions about what they want the software to do."

"If you have to explain/teach others how to do something, you end up with a better understanding."

Question: *"What were the challenges/drawbacks of working with partners from a different discipline?"*
"They didn't know exactly what they needed/wanted often."
"We were both learning so it was hard to explain if we didn't understand how things went."
"They didn't fully understand what were capable of doing and we didn't always understand the [bio]chemistry concepts."
"Having to teach basic knowledge of own discipline."
"Time management. Things they thought were quick problems weren't."
"They didn't have any experience writing spec documents before."
"You have to swap back and forth between different mindsets quickly."
"…getting the program to work the way they wanted."

Question: *"If you overcame challenges, how?"*

"By communicating … and programming late at night."
"We eventually learned how to speak the same language…"
"We used to finish programs early so bio partners can help us to verify results, give us feedbacks[sic]."
"Constantly calling, texting or meeting up."
"We did it by staying in close communication with CHEM students on our team."
"By redeveloping software at first. Later by making sure their requirements were correct."
"Frequent conversations with our CHEM partners. Calling them late at night."
*"*We overcame them by failing to give them what they wanted at first, so they figured out how to explain to us what they needed."
"We showed them how to give us useful specifications."

BIO 441.

Question: *What were the benefits of working with partners from a different discipline?*
"Learning how different we think from each other and how to communicate more effectively"

Question: "*What were the challenges of working with partners from a different discipline?"*

"Really the same as the benefits…differences in background knowledge and communication."
"They had little/no prior knowledge of the concepts, so everything had to be very clear and precise while also very detailed."
"They didn't know what we wanted, and we didn't know the extent of what they could do."

Question: *If you overcame the challenges, how?*

"Dogged determination, sacrifice, and communication efforts/skills got us through. When these failed, we failed."

“Learning to speak up when something is not correct on either side of the team and trying to teach each other bits of background info …”

“By trying to fully understand the information before communicating it to CS partners.”

“We had to discuss frequently and rewind our explanations until they made sense.”