A Distance Learning Approach to Teaching eXtreme Programming

Christian Murphy, Dan Phung, Gail Kaiser
Dept. of Computer Science
Columbia University
New York NY 10027
{cmurphy, phung, kaiser}@cs.columbia.edu

ABSTRACT
As university-level distance learning programs become more and more popular, and software engineering courses incorporate eXtreme Programming (XP) into their curricula, certain challenges arise when teaching XP to students who are not physically co-located. In this paper, we present the results of a three-year study of such an online software engineering course targeted to graduate students, and describe some of the specific challenges faced, such as students' aversion to aspects of XP and difficulties in scheduling. We discuss our findings in terms of the course's educational objectives, and present suggestions to other educators who may face similar situations.

Categories and Subject Descriptors
K.3.1 and K.3.2 [**Computers and Education**]: Computer Uses in Education - *Distance Learning*; Computer and Information Science Education – *Computer Science Education*.

General Terms: Management, Human Factors.

Keywords: Distance and distributed learning, Software engineering education, Test-driven development.

1. INTRODUCTION
Many universities offer distance learning programs for graduate students who are full-time professionals. At the same time, CS departments are incorporating agile processes like eXtreme Programming (XP) [1] into software engineering courses. As these two trends merge together, numerous challenges arise in teaching XP to students who are not physically co-located.

In this paper, we present our findings from a three-year study of such an online software engineering course, and describe some of the specific challenges we faced. These include students' aversion to some of the aspects of XP, difficulties in pair programming, and problems related to scheduling. We discuss these findings in terms of the course's educational objectives, using our observations and the students' assessment. We then present some suggestions to other educators who may face similar situations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Conference'04, Month 1–2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004…$5.00.

2. BACKGROUND
The COMS W4156 Advanced Software Engineering course at Columbia University focuses on topics such as process life cycle, project planning, team programming, and unit and integration testing. It also covers component-based software engineering models like EJB, CORBA, and COM. Most importantly, though, the course uses eXtreme Programming as its methodology, adjusted to the classroom environment (we note, however, that much of our findings are also relevant to other agile processes). Students are required to do all programming work in pairs, and then are combined into teams of four for their semester-long project. There are three XP iterations during the semester, each lasting approximately three weeks.

The Advanced Software Engineering course is taught on campus but also offered via the Columbia Video Network (CVN) [3], which is the graduate distance learning program of Columbia University's School of Engineering & Applied Science. Classes available through CVN are taught on campus in New York City by Columbia University faculty members. Faculty and students meet on campus in specially equipped classrooms, and the classes are recorded and made available electronically to registered CVN students via online streaming media. An important difference between CVN courses and other distance learning programs is that CVN students see the same lectures, have the same homework assignments, and take the same exams as their on-campus counterparts; the CVN courses are *not* specifically tailored to off-campus students. The motivation is to have CVN students receive the same learning experience as on-campus students, so that they may receive the same academic credit.

The CVN videos are not re-recorded each time the course is taught; the same set of videos may be re-used for many following semesters. This means that even when the course is not offered on campus, it can still be offered on CVN, as long as there is a CVN course manager to oversee it. A course manager is responsible for overseeing all aspects of the course for the CVN students, such as distributing and grading homework assignments, answering students' emails, and calculating final grades.

2.1 Course Objectives
The Advanced Software Engineering course is targeted to graduate and upper-level undergraduate students who have demonstrated the ability to work on large-scale software projects on teams of four or five students. The two main educational objectives of the course are as follows: (1) allow the students to participate in a project using eXtreme Programming, and learn the value of its core practices [27], like pair programming, test-driven development, small releases, and refactoring; and (2) teach skills

in the domain of quality assurance, such as unit and integration testing, and code inspections. Additionally, the students have the opportunity to get hands-on experience working with a component-based software model like EJB, CORBA, or COM.

2.2 Details of the Study

The first two authors of this paper were the course managers for the CVN offering of Advanced Software Engineering from 2004-2007, and also acted as TAs for the on-campus version of the course, which was offered once a year; the third author is the faculty member who taught both the on-campus course and the pre-recorded version. During that time, approximately 90 CVN students completed the course. All students in the Advanced Software Engineering course (whether CVN or on-campus) were required to write a subjective assessment of their experiences both at the midpoint of the course and upon its completion. In Section 3, we present comments made by the students in their written assessments, as well as our own observations.

It is important to note that because of the nature of the course we do not have any objective "outcomes" measurements, e.g. we cannot compare how well one team learned one component model against how well another team learned another (since each team worked with only one). We have chosen not to use grades as a measurement because of the various influential factors that go into a student's success in the course. We therefore are admittedly limited to subjective experiences and our own observations.

We also note that we cannot fairly compare the off-campus students to the on-campus students in all cases, because they may be taking a different version of the course (depending on when the CVN videos were recorded), and because of the relatively small numbers of off-campus students. Having said that, however, we do make some comparisons to demonstrate that some aspects of agile processes are indeed more challenging to teach to distance learning students than to on-campus students.

3. OBSERVATIONS & FINDINGS

CVN students who took this particular course tended to be full-time professionals in the software industry who were completing graduate degrees part-time; they may or may not have also been taking other CVN classes concurrently. Given the course's educational objectives, challenges arose because of the students' physical distance and diverse backgrounds and schedules.

3.1 Aversion to eXtreme Programming

One of the difficulties in teaching XP to CVN students stemmed from the fact that, whereas the on-campus students tended to be undergraduate or graduate students with little or no professional software development experience, CVN students in this course were almost always already employed as software developers and may have been using different methodologies in their professional work. We observed that students who had not been exposed to other methodologies had an easier time adapting to XP, whereas those who were using RUP, waterfall, rapid prototyping, *etc.* during their professional work found it difficult to change their mindset and approach while working on course assignments. Furthermore, some CVN students had already had bad experiences with XP, and found that it was unmanageable or did not apply to the particular project on which they were working. Other students expressed the typical criticisms of the main XP tenets, for example that programming in pairs is less productive or that it is too time consuming to write tests before writing code.

Concerning test-driven development, one student claimed that he and his partner had *"an incongruent idea of what unit testing is supposed to accomplish. In my [opinion], functionality should be driven by unit tests, which should be developed first. However, given the … constantly changing user interface, it was hard to settle on a consistent methodology. I would create tests that I felt expressed the intention we were trying to accomplish with the program, while [my partner] was creating and modifying the UI to fit the various … requirements. In the end, I needed to modify the unit tests to fit his program, which I believe was a backwards way of building this system."*

Another core practice of XP that can be difficult to address in such a setting is that of collective ownership. Many students who were used to dividing a project into separate distinct parts typically did not interact with anyone in the team but their pair programming partner, so they were not familiar with the rest of the team's code. Wrote one student, *"The other [pair in our team] had no idea how our code worked or even was organized, because [we] never bothered to discuss it. This created a severe challenge when we later needed to do integration and testing."*

Although this aversion to core XP practices is not necessarily an effect of distance learning in and of itself, it is still related because the distance learners who are CVN students tend to be software professionals, and those were the ones who typically had difficulty accepting XP.

3.2 Difficulties of Virtual Pair Programming

The inability for pair programmers to be physically co-located was perhaps the most obvious challenge in teaching XP in a distance learning course. Although the CVN students were almost never in the same physical location, they were still required to engage in pair programming activities, and needed to figure out a way to share a desktop environment and communicate.

One student indicated that *"pair programming is a very difficult thing to accomplish in the type of environment set forth by a remote class made up of professional students. From my experience, pair programming is hard to do even in a work environment with set hours and close proximity. Even though [my partner] and I are the closest in proximity of all the groups, … we have had the hardest time using the pair programming idiom."*

In our courses, most of the students used RealVNC or a similar tool as their shared desktop, and a regular phone line for voice communication. In the cases where students were unwilling or unable to make extended long-distance phone calls, they used VoIP technologies like Skype or voice-enabled chat tools like Windows Messenger. Very rarely did students have to rely on typing messages to each other via instant messaging, though in some cases this was necessary (though obviously much slower).

Aside from the technical challenge of actually conducting virtual pair programming, we had the difficulty of breaking the group of distance learning students into pairs. As described in [13, 18], it is typically desirable to match students based on skill level (actual and perceived), and students tend to gravitate to each other based on gender and ethnicity. Unfortunately, not only were the students unable to meet each other because they were not co-located, even the course manager (who determined the pairs) could not easily

get a sense of which students would work well together. In the past, we tried to pair students based on their level of programming experience, language expertise, and physical location. Unfortunately, though, in some cases pairs have not worked out well, typically because one of the students failed to perform well and neither the instructor nor the programming partner was physically present to encourage the student to participate more.

3.3 Scheduling Problems

Related to the virtual pair programming issues are the problems that arose from the students being in different time zones. Because Columbia University is on the east coast of the United States, most of the students (even the distance learning ones) tend to live in the metro-New York area, or at least in the Eastern Time Zone. And while all students (distance learning or not) have different time commitments and difficulty in scheduling time to work together, this problem was exacerbated by the fact that sometimes a student in one time zone would be paired with a student in another, which was particularly a problem when the two students were on different coasts of the United States (three time zones apart) and worked full-time jobs during the day. This usually only left weekends as potential times to work together, which was frustrating to many students.

One student complained, *"After the [midterm], the divide in my group grew even more. [My partner] and I started working much less closely because of scheduling issues and finally, I believe, out of frustration. Pair programming was far from a reality."*

Another student wrote, *"Pair programming is a challenge to me since I'm [used] to the traditional way of individual programming. The challenge is amplified by the fact that the other half of my pair [is] not only in a different physical location [from] me, but also in a different time zone. It is very difficult at times for us to find common blocks of time to work … together. I think it would be easier just to … work independently."*

Additionally, having the students in different locations made it impossible to have *ad hoc* "stand-up meetings", which are critical to any XP project. In fact, we observed that the only real-time communications students had were during scheduled meetings and pair programming sessions, and any other communication was almost always done by email.

3.4 Issues with Code Inspections

As part of the quality assurance aspect of the course, students participated in a formal code inspection with their entire team (four students total), led by the course manager. Though many of the students had already had similar experiences in their professional work, others stated that they missed out on the opportunity for a variety of reasons, including failure to adequately plan for the meeting, insufficient time allocated for the inspection, or technical and scheduling issues. One student wrote that the problem of the code inspection *"was a result of the technical communication problems we encountered doing a large conference call. In the future, I would suggest … finding a communication technology that can robustly support 4-6 way voice calling. It is also important to choose a time when everyone involved is available by phone (or some other voice medium). Use of [text-based] chat, as was necessary to communicate with one of the team members who was out of the country, slows down proceedings dramatically."*

4. ANALYSIS & ASSESSMENT

Based on our observations the students' own assessment, we next consider the course objectives to determine what impact the distance learning program had on the educational experience.

4.1 Learning the Value of XP Core Practices

Of the approximately 90 CVN students who took the course during the time of the study, more than half pointed out that pair programming in such an environment was too difficult to be effective in practice. The biggest complaint was that it was inefficient and caused the students to spend more time on the course than they had expected to or thought was necessary.

Given our observations and the students' statements, the distance learning approach to XP has, in our experience, failed to consistently meet the educational objective of teaching some of the core XP practices, in particular the value of pair programming. We estimate that at least half of the students abandoned the pair programming approach, though we do not know the exact number because students were told that they *must* use pair programming and only a few admitted not using it. On the other hand, only a small number (fewer than 20%) claimed to have enjoyed pair programming or learned valuable lessons from it. In general, CVN students often leave the course frustrated by their pair programming experience and without having realized its benefits, whereas on-campus students tended to work in the same labs, lounge areas, dormitories, *etc.* and thus had an easier time sticking with this practice.

Another XP practice that was not adequately learned by the students was test-driven development. Students were instructed to write unit tests first, but many (over 65%) admitted that, at some point during the course, they did not write unit tests first because of time constraints or because they did not see the benefits of doing so. This situation may arise because there is no personal influence (either by the "customer" or other team members) on an individual to adhere to the test-first approach; this may not be the case for on-campus students, of course.

Fortunately, other course objectives related to XP principles (such as executing a project with frequent, short release cycles, or using a simple, metaphor-based design) were met, despite any of the limitations of distance learning. However, it typically required extra work by the course manager (more than the on-campus TA would need to do) to overcome these obstacles, and there is certainly room for improvement and more efficient practices.

4.2 Developing QA Skills

The fact that the students were not co-located did not prevent them from achieving the objective of learning about the importance of unit, integration, and system testing. Although some teams failed to conduct thorough testing as end-of-term deadlines drew near, the percent of CVN teams that did so was not much different from the percent of on-campus teams.

Working in a distributed team did bring about challenges to performing the final code inspection, though. The review sessions invariably took much more time than the students expected, and even though the students were supposed to agree upon programming conventions at the beginning of the semester, ultimately they would have some differences of opinion which stemmed from their own personal experiences as professional

software developers. However, students were able to complete this task successfully and gain some benefits from it.

4.3 Other Issues

As mentioned, the CVN courses are not targeted specifically to off-campus students. Instead, the on-campus lectures are recorded and then distributed online. Additionally, the videos are not re-recorded each semester, due to costs and the fact that the course is not offered on campus every semester. Thus, the same set of videos for a course may be re-used for many following semesters.

The issue in such a software engineering course (or any course in which the technology is frequently changing, for that matter) is that the material in the recorded lectures may be out of date. In our case, the recordings of the lectures for the Advanced Software Engineering course were made in early 2004 and were used up until Summer 2007 (CVN recorded new lectures in Fall 2007). Because this course teaches the use of component models, the versions and capabilities of the different frameworks varied greatly between what was discussed in lecture (in Spring 2004) and what the students were actually able to download and use (as late as Summer 2007). For instance, the EJB 3.0 spec was not released until after the Advanced Software Engineering course was recorded for use on CVN, so those lectures described EJB 2.1. However, within a year or two many EJB container vendors had completely moved away from EJB 2.1, and students viewing those videos were unable to match what they were taught with what they were able to download and use.

The educational objective of working with component models was still met, but led to considerable difficulties for the course manager, who had to help the students overcome the difference between what they were taught and what was the reality of the state-of-the-art. Although this issue is not related to XP per se, it can come about when teaching in a distance learning program.

5. SUGGESTIONS

In teaching a distance learning course that focuses on XP, the limitations presented by requiring virtual pair programming are clearly the most difficult to overcome. We suggest using a combination of communication and shared desktop tools (as discussed in more detail in [24, 11, 26]), though we note, however, that in our case the students' difficulties with distance pair programming were not just of a technical nature, but often an issue with scheduling and personal preference.

Fixing the scheduling problems can quite difficult, since there are so many variables and so many unforeseen factors involved (business trips, work deadlines, family issues, *etc*.), as well as the particular problem of working across time zones. The best advice is to keep a fixed weekly schedule for pair programming sessions and team meetings, so that further time need not be spent on negotiating available times and rescheduling. In addition, although *ad hoc* face-to-face meetings are practically impossible, it is still important to communicate frequently, even if the meetings are not *ad hoc* and are not face-to-face.

When meeting in person is not possible, the course manager or instructor should also encourage telephone communication as a first choice, with instant messaging a second choice, and trading emails as a last option. Emails are too easily ignored and may not result in significant progress on a matter. Although frequent in-person meetings (either with the "customer" or other members of the team) are impossible because of geographic location, we suggest at least one face-to-face meeting at the beginning of the semester, if possible, even if not all members can attend.

It is important to ensure that students participating in the class are capable of working in distributed teams on a project with such short time scales. To help all students in the course benefit from their mutual experience, we screened students by their level of past project experience. Those who had not participated in a project longer than 5K lines of code were directed to take a more intermediate course, since the typical project consisted of a client-server system that consisted of 5-10K LOC and supported by third-party software plugins and frameworks. Proper screening of the students is critical in maximizing each student's contribution and benefit within their pairs and teams.

Sometimes the screening process is not enough to assess the impact of a student in the course, however. In our case, confidential peer assessments were conducted to gauge the personal and professional fit of the pairs (this was done for on-campus students, as well). It was not uncommon to find conflicts with regards to personality and working styles (time habits, controlling natures, idiosyncrasies, *etc*.) that sometimes required a redistribution of students. The result of such redistributions usually benefited the team and project completion.

6. RELATED WORK

There has been quite a bit of work in investigating software engineering education [8, 9, 10, 15, 20], but most of these do not address the challenges that come up from teaching in a distance learning setting. Edwards directly asks the question "can quality graduate software engineering courses really be delivered asynchronously on-line?" in his 2000 paper [7] and describes the structure of the course, the assignments, and the tools, but he does not discuss any of the challenges he encountered, nor did his course use eXtreme Programming (or even team programming), which brings about separate issues aside from those in a traditional software engineering course. Similarly, the CURE tool [2] facilitates collaboration and communication in distance learning software engineering courses, and Pankratius and Stucky [19] report on the technical, economic, and pedagogical challenges of teaching software engineering at a "virtual university"; however, neither of these addresses the difficulties that arise specifically from teaching eXtreme Programming and the accompanying non-technical challenges.

There have been experience papers published about teaching software engineering by using eXtreme Programming [16, 23], but none of these has addressed the problem of teaching it in a distance learning program. Work that does discuss the problems encountered in teaching computer science in distance learning programs tends to be for introductory courses [5, 21] or for advanced courses that were specifically designed for distance learning [17]; note that, in our case, the course was taught to on-campus students and recorded for distribution on the Internet, and was not specifically targeted to off-campus students. Tomayko [25] has looked at teaching XP in a distance learning course, but admits "this course was not a true test of distance education and XP," and only describes the problems with pair programming, and no other core XP practices.

Investigation of the teaching of "distributed software engineering" is very important as the software development community

becomes more globalized. However, the work in this field, which focuses on course design [4, 12], projects [22], and tools [6], does not incorporate the challenges of eXtreme Programming or (necessarily) of distance learning. Other work on Distributed eXtreme Programming [14] addresses many of the issues raised in this paper, but not in an academic setting.

7. CONCLUSION

We have discussed some of the challenges that arise from teaching software engineering using eXtreme Programming in a distance learning course. We hope that our experiences help other educators who face similar situations.

8. ACKNOWLEDGMENTS

Murphy and Kaiser are members of the Programming Systems Lab, funded in part by NSF CNS-0717544, CNS-0627473, CNS-0426623 and EIA-0202063, NIH 1 U54 CA121852-01A1. Phung is funded by NSF ITR grant CNS-0426623.

9. REFERENCES

[1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.

[2] P. Bouillon and J. Krinke, "Using Eclipse in distant teaching of software engineering", In *Proc. of the 2004 OOPSLA workshop on eclipse technology eXchange*, Vancouver, 2004, 22-26.

[3] Columbia Video Network, `http://www.cvn.columbia.edu`.

[4] D. Damian, A. Hadwin, B. Al-Ani, "Instructional design and assessment strategies for teaching global software development: a framework", In *Proc. of the 28th ICSE*, Shanghai, 2006, 685-690.

[5] G. Davies and J. Preece, "Computer science, home computing and distance learning—the largest computer science course in the world?", In *Proc. of the 21st SIGCSE*, Washington DC, 1990, 143-146.

[6] U. Dekel, "Supporting distributed software design meetings: what can we learn from co-located meetings?", In *Proc. of the 2005 Workshop on Human and Social Factors of Software Engineering*, St. Louis MO, 2005, 1-7.

[7] S. Edwards, "Can quality graduate software engineering courses really be delivered asynchronously on-line?", In *Proc. of the 22nd ICSE*, Limerick, Ireland, 2000, 676-679.

[8] P. Freeman, A. I. Wasserman, R. E. Fairley, "Essential elements of software engineering education", In *Proc. of the 2nd ICSE*, San Francisco, 1976, 116-122.

[9] P. Freeman, A. I. Wasserman, "A proposed curriculum for software engineering education", In *Proc. of the 3rd ICSE*, Atlanta, 1978, 56-62.

[10] C. Ghezzi and D. Mandrioli, "The challenges of software engineering education", In *Proc. of the 27th ICSE*, St. Louis MO, 2005, 637-638.

[11] B. Hanks, "Student performance in CS1 with distributed pair programming", In *Proc. of the 10th annual ITiCSE*, Lisbon, Portugal, 2005, 316-320.

[12] M. Hawthorne and D. E. Perry, "Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities", In *Proc. of the 27th ICSE*, St. Louis MO, 2005, 643-644.

[13] N. Katira, L. Williams, J. Osborne, "Towards increasing the compatibility of student pair programmers", In *Proc. of the 27th ICSE*, St. Louis MO, 2005, 625-626.

[14] M. Kircher, P. Jain, A. Corsaro, D. Levine, "Distributed eXtreme Programming", In *Proc. of XP2001*, May 2001.

[15] C.W. Liew, "Teaching software development skills early in the curriculum through software engineering", In *Proc. of the 10th ITiCSE*, Lisbon, Portugal, 2005, 133-137.

[16] C. Loftus and M. Ratcliffe, "Extreme programming promotes extreme learning?", In *Proc. of the 10th ITiCSE*, Lisbon, Portugal, 2005, 311-315.

[17] M. McDonald, B. Dorn, G. McDonald, "A statistical analysis of student performance in online computer science courses", In *Proc. of the 35th SIGCSE*, Norfolk VA, 2004, 71-74.

[18] C. McDowell, L. Werner, H. E. Bullock, J. Fernald, "The impact of pair programming on student performance, perception and persistence", In *Proc of the 25th ICSE*, Portland OR, 2003, 602-607.

[19] V. Pankratius and W. Stucky, "Information systems development at the virtual global university: an experience report", In *Proc. of the 27th ICSE*, St. Louis MO, 2005, 639-640.

[20] D. Petkovic, G. Thompson, R. Todtenhoefer, "Teaching practical software engineering and global software engineering: evaluation and comparison", In *Proc. of the 11th ITiCSE*, Bologna, Italy, 2006, 294-298.

[21] J. A. Preston, L. Wilson, "Offering CS1 on-line reducing campus resource demand while improving the learning environment", In *Proc. of the 32nd SIGCSE*, Charlotte NC, 2001, 342-346.

[22] I. Richardson, A. E. Milewski, N. Mullick, P. Keil, "Distributed development: an education perspective on the global studio project", In *Proc. of the 28th ICSE*, Shanghai, 2006, 679-684.

[23] J. Schneider and L. Johnston, "eXtreme Programming at universities: an educational perspective", In *Proc of the 25th ICSE*, Portland OR, 2003, 594-599.

[24] D. Stotts *et al*., "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming", *Extreme Programming and Agile Methods - XP/Agile Universe 2003*, Springer, Berlin/Heidelberg, 2003.

[25] J.E. Tomayko, "Teaching eXtreme Programming Remotely", In *Proc. of the 18th CSEET*, Ottawa, Canada, 2005, 17-24.

[26] A. M. Zin, S. Idris, N. K. Subramaniam, "Implementing Virtual Pair Programming in E-Learning Environment", *Journal of Information Systems Education*, Summer 2006.

[27] `http://www.xprogramming.com/what_is_xp.htm`