Experiences in Teaching eXtreme Programming in a Distance Learning Program

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 our experiences and observations from managing such an online software engineering course, and describe some of the specific challenges we faced, such as students' aversion to using XP and difficulties in scheduling. We also present some 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.

1. INTRODUCTION
Many universities offer distance learning programs for graduate students who are full-time professionals. At the same time, computer science departments are incorporating core software engineering principles into their courses, and introducing agile processes like eXtreme Programming (XP) [1] as a main focus of software engineering methodology. 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 experiences and observations from managing such an online software engineering course, and describe some of the specific challenges we faced. These include students' aversion to using XP, difficulties in pair programming, problems related to scheduling, the lack of a personal relationship with the "customer", and issues stemming from out-of-date course material. We also present some suggestions to other educators who may face similar situations.

Conference'04, Month 1–2, 2004, City, State, Country.

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 experiences are also relevant to other agile processes). Students are expected 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), which is the graduate distance learning program of Columbia University's School of Engineering & Applied Science [4]. 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 advantage is that 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 (including during the summer), 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. The first two authors of this paper were the course managers for Advanced Software Engineering from 2004-2007, during which time each course offering generally had 6-8 CVN students per semester; the third author is the faculty member who taught the pre-recorded course.

3. CHALLENGES
CVN students who take this particular course tend to be full-time professionals in the software industry who are completing Masters degrees part-time; they may or may not also be taking other CVN

classes concurrently. Since CVN students taking Advanced Software Engineering must also use eXtreme Programming, challenges arise because of their physical distance and diverse backgrounds and schedules.

3.1 Aversion to eXtreme Programming

One of the difficulties in teaching XP to CVN students stems from the fact that, whereas the on-campus students tend to be undergraduates or Masters students with little or no professional software development experience, CVN students in this course are almost always already employed as software developers and may be using different methodologies in their professional work. Students who have not been exposed to other methodologies have an easier time accepting XP, whereas those who are using RUP, waterfall, rapid prototyping, *etc*. during their professional work find it difficult to change their mindset and approach while working on course assignments.

In some cases, 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. Or they expressed the typical criticisms of main XP tenets, for example that programming in pairs is less productive or that it is too timely to write tests before writing code. Although this aversion to XP 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 are the ones who have difficulty accepting XP.

3.2 Virtual Pair Programming

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

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 MSN Messenger. Very rarely did students have to rely on typing messages to each other, though in some cases this was necessary; obviously, that practice was discouraged.

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 [16], 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 are the students unable to meet each other because they are not co-located, even the course manager (who determines the pairs) cannot get a sense of which students may work well together.

In the past, we have tried to pair students based on their level of programming experience, language expertise, and physical location (so that they are in the same time zone). Unfortunately, though, in some cases pairs have not worked out well, typically because one of the students fails to perform well and neither the instructor nor the programming partner is physically present to encourage the student to participate more.

3.3 Scheduling Issues

Related to the virtual pair programming issues are the problems that arise 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. Unfortunately, this situation became inevitable when there would be one student on the west coast and all others on the east coast. Students in the Eastern Time Zone were generally unwilling to pair with someone who worked until 7pm or 8pm Pacific Time (which is 10pm or 11pm ET). This usually only left weekends as potential times to work together, which was frustrating to many students.

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 strictly done by email.

Lastly, the challenges presented in scheduling conspired with the challenges presented by the students' various programming backgrounds when it came to doing code reviews. In our course, a code review was one of the final tasks of the semester, and it was difficult for the students to schedule enough time for the review sessions. Moreover, the review sessions invariably took much more time than the students expected. 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.

3.4 Course Manager as Customer

In any software engineering course that teaches XP, the issue arises "who is the customer?" Ordinarily it is the instructor or a teaching assistant, but in our CVN course it was the course manager who took on that role. This raised a number of challenges. Scheduling meetings with the "customer" was a problem because the CVN students were grouped into teams of four (two pairs), so the issue arose of finding a time when five people were free, which in and of itself is a challenge, compounded by the fact that (as described above) they sometimes may have been in different time zones.

Aside from any technical issues, such as trying to arrange five-way phone conversations or setting up a shared whiteboard, the main problem is that the CVN students had a very impersonal relationship with their "customer". In an on-campus XP course, when the customer is a member of the teaching staff, the students can engage in face-to-face meetings with the customer. In our case, though, the students had no way of getting to know the customer personally, and could not get a "feel" for how the customer was reacting. One of the main reasons for the XP practice of having the customer always be available is so that

professional relationships can blend into personal relationships, but this was generally impossible in our case.

3.5 Out-of-synch Course Material

As mentioned, the CVN courses are not targeted specifically to off-campus students. Instead, the on-campus lectures are recorded and then distributed. 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 is recording 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 released right 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 using. This 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.

4. SUGGESTIONS

It is first 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 was 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, frequent 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.

Fixing the issues around scheduling is probably the toughest challenge, since there are so many variables and so many unforeseen factors involved (business trips, work deadlines, family issues, *etc*.). 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. Such a meeting can generally be held if students live near each other or in the vicinity of the university campus, and provides a suitable background for all subsequent conversations.

Obviously the best way to address the issue of having out-of-date material is to update the material more frequently, *i.e.* to record new lectures every semester or at least every year. However, this is not always feasible and, in our case, not a decision that we were able to make. One suggestion for addressing this challenge is to find (or create) documentation that describes the important differences between what is described in the lecture and what is available currently. Another suggestion would be to borrow material from the teaching assistants of the most recently taught on-campus course, since they would have material related to what is more up-to-date.

5. RELATED WORK

Of course there has been quite a bit of work in investigating software engineering education [9, 10, 24, 11, 26, 2, 15], but 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 [8] 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, which brings about separate issues aside from those in a traditional software engineering course. Similarly, the CURE tool [3] facilitates collaboration and communication in distance learning software engineering courses, and Pankratius and Stucky [20] report on their experiences of teaching software engineering at a "virtual university"; however, none of these address the difficulties that arise from teaching eXtreme Programming and the accompanying non-technical challenges.

Others discuss some of the problems they have encountered in teaching computer science in distance learning programs, but these tend to be for introductory courses [6, 21] or for advanced courses that were specifically designed for distance learning [18]; note that, in our case, the course was taught to on-campus students and recorded for distribution on the Internet (by CVN policy), but was not specifically targeted to off-campus students.

There have been a number of experience papers published about teaching software engineering by using eXtreme Programming [14, 23], but none of these has addressed the problem of teaching in a distance learning program. Others [16, 19] have looked at the dynamics of pair programming in the classroom setting, but not for "virtual pairs" who are not physically co-located. Many of the problems related to virtual pair programming have been addressed [12, 25, 27], and [7] has described how distributed software

design meetings can be conducted, but these focus mainly on the technical issues and not necessarily the pedagogy.

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 [5, 13], projects [22], and tools [7], does not incorporate the challenges of eXtreme Programming. Other work on Distributed eXtreme Programming [17] addresses many of the issues raised in this paper, but not in an academic setting.

6. CONCLUSION

We have discussed some of the challenges that arise from teaching software engineering using eXtreme Programming in a distance learning course. Despite these challenges, there are many benefits to such a program. We hope that our experiences help other educators who face similar situations.

7. 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.

8. REFERENCES

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

[2] M. Bernhart *et al*., "Dimensions of software engineering course design", In *Proc. of the 28th ICSE*, Shanghai, 2006, 667-672.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] B. Hanks, "Virtual Pair Programming", *Doctoral Symposium at the 25th ICSE*, Portland OR, 2003.

[13] 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.

[14] G. Hedin, L. Bendix, B. Magnusson, "Introducing software engineering by means of Extreme Programming", In *Proc. of the 25th ICSE*, Portland OR, 2003, 586-593.

[15] S. Honiden *et al*., "Top SE: Educating Superarchitects Who Can Apply Software Engineering Tools to Practical Development in Japan", *Proc. of the 29th ICSE*, Minneapolis MN, 2007, 708-718.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[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] M. Shooman, "The teaching of software education", In *Proc. of the 14th SIGCSE*, Orlando FL, 1983, 66-71.

[25] 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.

[26] H. van Vliet, "Some myths of software engineering education", In *Proc. of the 27th ICSE*, St. Louis MO, 2005, 621-622.

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