Balancing Hands-on and Research Activities: A Graduate Level Agile Software Development Course

Joseph Chao
Department of Computer Science
Bowling Green State University
Bowling Green, OH 43403
jchao@bgsu.edu

Abstract

Agile software methodologies promote developing better software faster and have been gaining popularity in industry. However, agile methods are still unfamiliar in the academic world. While it is important to introduce agile methods to undergraduate students, we believe that having a graduate-level course in agile development is as crucial. We offer a graduate-level topic course that is dedicated to teaching agile software development while emphasizing both hands-on experience and research activities. In this course, students first work on a series of projects with real customers and then propose and conduct research on topics in agile development. The result of this approach has been well received by students and has stimulated student research interests in the area. This paper describes our experiences in offering the course.

1. Introduction

Developers in the United States wasted over $55 billion, against $255 billion in project spending, on failed and poorly run software projects in 2002, according to the Standish Group 2003 CHAOS Report [15]. These failures can be traced to a lack of understanding of the business issues being addressed and the failure to use appropriate implementation methodologies. The solution is not in creating better technologies, but in the way in which systems are developed.

Agile software development is a new set of software methodologies, lead by eXtreme Programming (XP), which promises to produce better software more quickly and more cheaply. By emphasizing effective communication, striving for customer satisfaction, and responding to changes, agile methods have shown strength in recent years, and more and more people in industry are convinced that agile methods are viable solutions to the software crisis that has been building over the last forty years. However, as with most technology-related approaches, many academics are behind the curve on agile development.

The good news is that academics are catching up. Many academics have reported adaptations of agile methods in classroom settings, though mostly in undergraduate studies. In many instances, key practices in XP were introduced to lower-level Computer Science (CS) students in programming proficiency courses. Pair programming is the most studied and the most applied XP practice in the classroom, as noted in [3], [5], [7], [9], [12] and [20]. The benefits of pair programming are well documented in [4] and [13]. Many other XP practices are also used in lower-level CS classes. For example, students in [16] learned unit testing and refactoring in their Java programming course, students in [1] used small releases and refactoring in addition to pair programming, and the CS1 course in [10] adopted all XP practices. For upper-level CS courses, educators often try to integrate XP or agile development methods into software engineering or senior capstone project courses. Examples can be found in [6], [8], [11], [18], [19] and [21]. While most found XP integration a success, a survey [14] showed that students in a senior software engineering class opposed the idea of using XP for the course. One reason cited was that students felt that XP is not for beginners, since it requires some prior knowledge of the software development life cycle to fully understand and appreciate the concept behind it.

2. The course description

The CS department at our university offers BS and MS degree programs, but not a doctoral program. While it is important to introduce agile methods to undergraduate students, we believe that introducing agile methods to our MS students is as crucial. Since most of our MS students work in the software development field after graduation, it is critical for them to learn these breakthrough agile methods before entering the job market. However, without a well-designed course dedicated to agile development, they are forced to learn from other sources and/or by other means. Furthermore, since they typically stay in the program no more than two years, there may not be enough time for them to really understand agile methods or research additional topics in the area.

We proposed a graduate topic course, titled "Agile Software Development," that is designed to introduce agile software development methodologies to our graduate students. The course was approved by the CS department and was first offered in fall semester, 2004.

The prerequisite to this course is previous experience in software engineering, which can be satisfied either by previous coursework in software engineering or equivalent work experience upon instructor approval. This restriction is intended to allow the instructor to omit explanations of the basics of the software development life cycle, or details of various traditional software processes such as waterfall models or other incremental or iterative models.

Both hands-on software project development and research activities are required in this course. The final student grades for the course are determined by exams (fifty percent), software projects (twenty-five percent), a research paper and a presentation (fifteen percent), and class participation (ten percent).

The course was well received by students, though most of them had never heard of XP or agile software development prior to the course. During our first iteration there were fourteen students enrolled in the class. Since there are only forty one total students in the MS program, this was considered high for a topic course. Students in the class were excited about learning new and better software methodologies.

3. Course objective

The objective of this course is to introduce various agile methods and stimulate student research interests in the area of agile software development. With that in mind, the course first provides an overview of the entire agile movement in the software industry, and then focuses on teaching XP practices. By focusing on a single well-formed and better-defined methodology such as XP, students can develop a better understanding of how to apply agile methods to a complete software development life cycle. Students are informed that the course will emphasize software development processes rather than technologies, and focus on development activities rather than the final products. That is, while students will definitely learn new technologies and produce software systems along the way, the main objective is to learn the development process – agile methods – and to be able to conduct research in the area.

The development environment for the first iteration of this course was Microsoft .NET with C#, due to its widespread use and its availability in our institution. However, neither the technology nor the tool was the focus of this course; rather students must engage in their own learning process and acquire the necessary skills through their own efforts. For instance, on the first day of the class, students were informed that at the end of three weeks they were expected to be familiar with .NET and C#.

The schedule of this course was ambitious. In a sixteen-week semester, students learned agile methods (particularly XP), worked on three projects exercising all twelve XP practices, and conducted research on advanced agile-related topics. In addition, two of the three projects were real-world development projects with actual customers. The idea was to introduce some XP practices quickly and engage students in XP projects as soon as possible. While students worked on their first project, the instructor simultaneously prepared them for the next project that demanded more agile practices. By the mid point of the semester students had obtained adequate agile skills to think creatively and propose research topics.

4. Balancing hands-on and research activities

One difference between graduate and undergraduate students is their research ability. Further, it is evident that students learn software engineering concepts better when provided with hands-on exercises. To provide a course that encompassed both hands-on exercises and research, we adopted a textbook [17] that was written to support a hands-on project-based approach, and supplement the book with papers and other handouts on the subject of advanced agile topics. We also require a research project.

The lectures in the first half of the semester cover most of the chapters in the textbook to ensure that students gain a thorough understanding of XP and agile development concepts. At the midway point of the semester, as students work on development projects, the research project begins. One of the take-home midterm exam questions requires that students write a two to five-page single-spaced research proposal on an agile development-related topic.

It was not surprising that students in the first iteration of the class were eager to work on development projects applying their newly acquired knowledge in XP and agile development. However, we were pleasantly surprised to find that many students also enjoyed doing research in agile development. Our observation was that students were curious about the new methodologies and wanted to know more. The take-home midterm question that required a research proposal served to stimulate their interest.

For hands-on activities, we adopted a phase-in approach in which a series of projects – small, medium and large – were used to phase-in students on XP practices. This approach was advocated in [6], in which it was found that such an approach avoids overburdening students with too many new practices. However, our motivation was to quickly introduce sufficient concepts and skills so that students could start on the small project sooner.

The small project required students to develop a handful of object classes to perform some simple tasks such as counting words and validating passwords. Students worked in pairs for this project. The objective of the project was to have students practice the XP concepts of pair programming and test-driven development with nUnit. By the fifth week of class students were ready to begin their second (medium) project with real customers. Both the medium and the large projects involved real customers on real-world projects.

5. Real-world projects with real customers

While more tightly controlled projects allow each student the opportunity to practice crucial development skills, we believe it is better for graduate students to learn agile methods using real-world projects with real customers. Real-world projects expose students to situations and learning opportunities that cannot be replicated in tightly controlled projects. There is no better way to practice XP or other agile methods than working with real customers on real-world projects where communication is vital and changes are highly anticipated. Both real-world projects and agile methods involve communication, frequent feedback, and customer satisfaction-skills that many undergraduates have not yet developed. Graduate students are typically more mature and more motivated to learn, and therefore are better able to handle the demands of real-world projects.

5.1. The first real customer project

The medium project was the first real-world project with real customers. For this project, the fourteen students formed three teams to work on three different projects proposed by three different customers. We were fortunate to have an even number of students to practice pair programming throughout the semester. One team of four worked for a nonprofit organization on a "Horse Race" game application to be used in a fundraising event. Another team of four developed a "Meeting Agenda" web application for an academic department at the university. The system allows authorized members to view, add, delete or modify agenda items for scheduled meetings. The remaining six students worked with a local elementary school developing a "Billing System" for their extended-day childcare service.

The project was time boxed at four weeks. All XP practices such as pair programming, test-first, the planning game, small releases, metaphor, simple design, collective ownership, and coding standard were to be strictly applied. The use of user stories and storyboards were also required. During this project, a version control tool, Microsoft SourceSafe, was introduced and put into practice. While it was not possible to have on-site customers in an academic setting, all teams started the project with an initial face-to-face customer meeting.

In that meeting, the team played the XP planning game with the customer. After explaining the software process in plain English to the customer, the whole team, both the student and the customer, worked together and created a set of story cards arranged in the order of priority. The instructor observed the first customer meeting of each team to ensure that the meeting was effective and the project was understood. This added to the instructor's workload since the meetings were held outside of the regular class time. However, we believed that this was necessary since students often didn't know how to start a project, or even how to ask good questions to elicit user requirements. Furthermore, getting the whole team (including the customer) together was not an easy task.

At the end of the allotted four weeks, the Horse Race team delivered a superb product and had an end-of-project lunch celebration sponsored by the customer. Unfortunately, the Meeting Agenda team produced a low-quality product that lead to an unhappy customer. The team tried to mend fences by "maintaining" the product after the delivery. However, the customer finally gave up on the project, and the team had their first failed project in the real world. Fortunately, there were no hard feelings since we did explain to the customers at the beginning of each project that there were risks involved due to student learning.

The Billing System team unwisely committed to more than ten weeks worth of work, thinking that they could do it in four weeks if they worked hard enough. The instructor observed the situation and decided not to interfere at that point. The team of six bright students worked extremely hard on the project but, as the instructor expected, they were way behind schedule after the first two iterations (two weeks into the project since they had chosen to use one-week iteration). They also violated an important XP practice, sustainable pace. To make a long story short, they enlisted the instructor's assistance and negotiated with the customer for a two-phase plan. Phase 1 was to be completed when the medium project was due at the end of the fourth week, and Phase 2 became their large project, providing them with an additional eight weeks for the project. This project provided them with a valuable lesson that they would never have learned without working with real customers.

5.2. Student experience on the real projects with real customers

Students were very appreciative of the opportunity to work with real customers. Many students commented that having real customers was the one thing that they did not want to see changed in this course. Other comments were also positive, such as "working on problems for real clients gives us experience on real world issues" and "working with real clients gives us an early exposure to real-life work environments."

As much as they valued real customers, students lacked the experience of working and communicating with them. One major issue was that, unlike a controlled classroom project, there was no clear scope on any of the projects. Without prior real customer experience, it was difficult for students to come up with a reasonable estimation on the project. As a result, the Meeting Agenda team misunderstood the customer requirements and ended up with a failed project, and the Billing System team underestimated the scope and exceeded their budgeted time by two hundred percent.

Nevertheless, all students agreed that this was a great experience and that they had learned a lot from real projects. Student feedback reaffirmed that real customers were critical when learning agile methods.

5.3. The large project

The two large projects started in the ninth week, when students were regrouped into two teams of six and eight. As noted earlier, one of the large projects was Phase 2 of the Billing System, and was conducted by the original six-student Billing System team that began the project. In this phase they placed greater emphasis on customer communication and satisfaction. As a result they were able to complete the system on time, on budget, and with satisfied customers. The other large project was to develop a "School Website" for the elementary and junior high school that commissioned the Billing System project. The website consists of three main areas: parent, teacher and student areas.

The large projects went much more smoothly than the medium projects. The success can be attributed to previous project experience, better planning, and the synergy among team members.

6. The research project

The research project also started in the ninth week of the semester. As mentioned previously, one of the midterm exam questions required students to write a two to five page proposal on a research topic in agile software development that had not been discussed in class. Each student was required to find a topic of interest, to perform preliminary research, and to describe the research objective along with a proposed approach. The quality of the student proposals was excellent, and many were above and beyond our expectations.

The research projects were to be conducted in pairs, following the concept of pair programming. Some of the fourteen student proposals were combined and others were modified. Ultimately seven topics were chosen through the combined efforts of the students and the instructor. The final seven topics were as follows.

- Better ways to deal with the issues of personality traits in pair programming
- Automated GUI testing
- The effectiveness of refactoring in an XP environment
- Agile modeling - its viability and adaptability
- What is Scrum?
- DSDM and its combination with XP
- Crystal software development methodology: can a mixture of XP and Crystal improve overall effectiveness?

Each of the seven students who wrote the original proposals was assigned to lead the research and was paired with one of the other seven students. Each pair of students was required to turn in a research paper and to give a thirty-minute presentation on the topic.

The research requirement was a valuable course component for a variety of reasons. It served to expose students to rudimentary research design and techniques. The presentations provided the opportunity for students to practice their communication skills and to be exposed to additional advanced concepts beyond the one that they personally researched. Overall the research presentations were well done and greatly enhanced the class.

7. Student perceptions

Since this was the first agile development course ever offered in our department, and since the course involved so many new pedagogical approaches, the instructor monitored student progress closely. The overall feedback from students, based on an end-of-semester survey of the fourteen students, was overwhelmingly positive.

All students indicated that they learned a great deal about agile development and that they would like to learn more. Most of them were convinced that agile and/or XP methods were viable solutions to today's software development. However, one student was skeptical about XP, and stated that "while I like the course, my experience with XP so far has not been positive." Students indicated that their favorite aspect of the course was the opportunity to work with real customers, and the feature that they most disliked was the amount of work required. Many of them also stated that they liked the research aspect of the course, since they not only learned how to conduct research but also benefited from the results of others' research.

Students were much more motivated once the project began since they were finally able to put their knowledge to the test. Use of the phase-in approach allows student to start working on a project early. As pointed out by [6], the approach avoided overburdening students with too many new practices at once. However, the approach also created a heavy workload for both students and the instructor.

There was not much breathing room with three hands-on projects and one research project. In a sixteen-week semester, students had two weeks to learn the tools and technologies required by the course, two weeks for the first (small) project, four weeks for the second (medium) project, and eight weeks for the third (large) project. While working on the large project, they simultaneously had to work on the research project. Three students declared that the workload in the class was "more than any other classes I have ever taken." One student commented that "it feels as though the project is my second job." Two students reported that they put in more than twenty hours a week for the real world projects.

The student suggestions for reducing the workload included the following:

- Reduce the number of projects or have smaller projects.
- Do not assign both a development project and the research project at the same time.
- Provide more time or more people on the projects.

It is unlikely that adding people to projects will reduce workload, considering the added overhead required by team communications. We observed that while this was a demanding course, the workload was manageable with good teamwork and a more equal distribution of work among team members. Students, like many professionals, often do not understand the art of teamwork. In fact, we felt the need to give a lecture near the middle of the semester titled "teamwork is an individual skill" based on [2]. Teamwork seemed to improve considerably after the lecture.

Another reason that students had problems with the workload was their lack of real customer experience. Since a major portion of the project grades were based on customer satisfaction, students felt obligated to do everything the customers requested. Based on student feedback, the next time that we offer this course we will present the "team work is an individual skill" lecture earlier in the semester, and will also consider reducing the number of projects.

In terms of the workload for the instructor, each project involved a great deal of preparation, monitoring, mentoring, and grading, all of which were time consuming. The incorporation of real-world software projects entails maintaining customer relationships, coordinating meetings, and ensuring quality products. It is inevitable that as more projects are required, the instructor workload increases. However, we felt that the course outcomes in both student learning and student satisfaction made the additional workload worth the effort.

8. Conclusion

This paper describes our experience in offering a graduate-level agile software development course that emphasizes both hands-on experience and research activities. The course was well received by students and many of them expressed interest in learning more about agile methods. Since the students all had prior background in software development, they were able to grasp the concepts behind agile methods quickly and apply those concepts to real-world projects with real customers.

Use of a phase-in approach, in which students were assigned small, medium, and large projects successively, worked well in providing students with hands-on experience much sooner than the more commonly used approach of a single large project assigned closer to the middle or end of the course. However, the shorter and more frequent deadlines stressed some students. In future iterations of the course we may consider reducing the number of projects.

Students enjoyed working on real-world projects with real customers. They learned many valuable real-world lessons from working on the projects, something that is generally not possible with tightly controlled classroom projects. Students also liked the research requirement, which allowed them to learn more advanced topics about agile development.

Our first foray into teaching agile software development at the graduate level was both satisfying and instructional. While student feedback indicates that some tweaking of the format is in order, students expressed overall satisfaction with the approach. This demonstrates that although teaching agile development almost requires a real-world project, careful planning and organization make it not only possible, but rewarding to both the student and the instructor.

9. References

[1] Astrachan, O.L., Duvall, R.C., and Wallingford, E., *Bringing Extreme Programming to the Classroom*, in *XP Universe*, Raleigh, NC, USA, July 2001.

[2] Avery, C, Walker, M. and Murphy, E. Teamwork is an Individual Skill: Getting Your Work Done When Sharing Responsibility, Berrett-Koehler Publishing, 2001.

[3] Bevan, J., Werner, L., and Dowell, C.M., Guidelines for the Use of Pair Programming in a Freshman Programming Class, in *15th Conference on Software Engineering Education and Training (CSEET'02)*, Kentucky, USA, 2002.

[4] Cockburn, A., and Williams, L., The Costs and Benefits of Pair Programming, in *eXtreme Programming and Flexible Processes in Software Engineering-XP2000*, Italy, 2000.

[5] DeClue, T.H., Pair Programming and Pair Trading: Effects on learning and Motivation in a CS2 Course, *Journal Of Computing Sciences in College (JCSC).* 18:5, 2003.

[6] Fenwick, J., *Adapting XP to an Academic Environment by Phasing–In Practices*, in *XP/Agile Universe 2003*, New Orleans, LA, USA, August 2003.

[7] Hanks, B., McDowell, C., Draper, D., and Krnjajic, M., Program Quality with Pair Programming in CS1, in *Annual Joint Conference on Integrating Technology into Computer Science Education (ITiCSE)*, 2004.

[8] Johnson, D. and Caristi, J., Extreme Programming and the Software Design Course, in *XP Universe*, Raleigh, NC, USA, July 2001.

[9] McDowell, C., Werner, L., Bullock, H., and Fernald, J., The Effects of Pair-Programming on Performance in an Introductory Programming Course, in *33rd ACM Technical Symposium in Computer Science Education*, Covington, Kentucky, 2002.

[10] McKinney, D., Froeseth, J., Robertson, J., Denton, L.F., and Ensminger, D., Agile CS1 Labs: eXtreme Programming Practices in an Introductory Programming Course, in *XP/Agile Universe 2004*, Calgary, Canada, August 2004.

[11] Melnik, G., and Maurer, F., introducing Agile Methods in Learning Environments: Lessons Learned, in *XP/Agile Universe 2003*, New Orleans, LA, USA, August 2003.

[12] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., and Balik, S., Improving the CS1 Experience with Pair Programming, in *34th Technical Symposium in Computer Science Education, SIGCSE 2003*, Reno, Nevada, 2003.

[13] Padberg, F., and Muller, M.M., Analyzing the cost and Benefit of Pair Programming, in *Ninth International Software Metrics Symposium (METRICS'03)*, Sydney, Australia, 2003.

[14] Sanders, D., *Student Perceptions of the Suitability of Extreme and Pair Programming*, in *XP Universe*, Raleigh, NC, USA, July 2001.

[15] Standish Group, 2003 CHAOS Chronicles, The Standish Group International (http://www.standishgroup.com), summarized at http://www.findarticles.com/cf_dls /m0EIN/2003_March_25/99169967/p1/article.jhtml

[16] Steinberg, D.H., *The effect of unit tests on entry points, coupling and cohesion in an introductory Java Programming Course*, in *XP Universe*, Raleigh, NC, USA, July 2001.

[17] Steinberg, D.H., and Palmer, D.W., *Extreme Software Engineering–A Hands on Approach*, Pearson-Prentice Hall, 2004.

[18] Wainer, M., Adaptations for Teaching Software Development with Extreme Programming: An Experience Report, in *XP/Agile Universe 2003*, New Orleans, LA, USA, August 2003.

[19] Williams, L., and Upchurch, R., Extreme Programming for Software Engineering Education, in *31st ASEE/IEEE Frontiers in Education 2001 Conference*, Reno, NV, USA, 2001.

[20] Williams, L., Wiebe, E., Yang, K., Ferzil, M., and Miller, C., In Support of Pair Programming in the Introductory Computer Science Course, *Computer Science Education,* 12:3, 2002.

[21] Wilson, D., Teaching XP: A Case Study, in *XP Universe*, Raleigh, NC, USA, July 2001.