Combining Multiple Pedagogies to Boost Learning and Enthusiasm *

Lori Pollock and Terry Harvey
Computer and Information Sciences
University of Delaware
Newark, DE 19360
{pollock, harvey}@cis.udel.edu

ABSTRACT

This paper describes the pedagogy we applied in a 5-week class, in which students taught themselves (and each other) a new language, new OS, GUI programming, and simple networking for collaborative games. They learned communication, negotiation, collaboration, presentation and teamwork skills; and project design and iterative development. We had four goals: increased learning, enthusiasm about CS, confidence in technical ability and communication skills. To achieve these goals, we decided to rely solely on the integration of teaching techniques that we believed would be highly effective: collaborative teams, student presentations, student critique of work, open-ended projects of student design, iterative process, journal reflection, and motivation through helping others. The students had to learn about each technique through discussion, modeling, and moderated practice. We focused on this process learning and trusted that the technical material would come from solving the (unspecified) assignments. This focus left no time for traditional teaching activities. We present quantitative and qualitative results from a student survey and the students' reflective journals. Students reported learning at a greater rate than in other CS courses while maintaining (and in some cases acquiring) a high level of enthusiasm and confidence.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computers and Education-Computer science education

General Terms

Human Factors

Keywords

collaborative learning, studio-based classroom, service learning, reflective journaling, problem-based learning

*This material is based on work supported by the National Science Foundation grant CNS-0940501.

ITiCSE'11, June 27–29, 2011, Darmstadt, Germany.

1. INTRODUCTION

In Fall 2009, the authors decided to run an independent study together. We wanted 3 to 5 students to help us explore the development of software applications for the XO laptop [13], and for the Myro [2] robot. We believed that both topics had potential as classroom projects, and wanted to explore them with a small set of students before committing to a larger class. The only common ground was that both the XO laptop and Myro robot could be programmed using Python.

We were overwhelmed with 26 applications. Instead of selecting only 3-4 students, we viewed the challenge as an opportunity to experiment with integrating pedagogies that supported students managing and teaching each other. We had enough XOs and Myro robots for three teams of four students to have one of each per team.

The key challenge was how to design and execute a learning experience given these challenges: a large set of new concepts including programming language, hardware and operating system platform, and an event-based programming paradigm that neither instructors nor students knew at all; a 5-week session; 12 students, with a range of student backgrounds from sophomore (with two prior semesters of CS) through graduating seniors; and maintaining an independent study-like, exploratory learning environment.

Our primary goals were to increase overall learning, enthusiasm, confidence in technical abilities, and communication skills. Our secondary goal was that students develop an *awareness* of their learning and their ability to teach themselves and each other. To achieve these goals, we did not directly teach computer science, but instead taught students to work in teams, critique each others' work, design their own programming projects of increasing complexity, use iterative development, and to reflect on their learning in a personal journal.

The main contributions of this paper are:

- a description of how we integrated a set of pedagogies known individually to engage computer science students in learning, to create an independent study-like experience for a 12-student class, and
- evidence both quantitatively and qualitatively of the potential impacts of such a course design on student learning, attitudes, and perceptions.

2. INSTRUCTOR AS GUIDE

It was difficult for us to give up the teaching methods we had developed for so long: lectures, textbooks, and pre-

scribed assignments. We were both excited by the possibilities of new methods, and wanted to try them in a short winter session where we would have time to collaborate, innovate, and support each other through the experiment.

We charged student teams with deciding how to learn the assigned concepts and how they taught it to each other. In this environment, the role of the instructor may not be immediately obvious. Establishing a safe and exciting learning environment was paramount. We led exercises to elicit and consolidate student knowledge on: what makes good teams work; tools and techniques to facilitate teamwork; what makes a good presentation; and how to offer kudos and critique on another team's presentation. We interleaved these exercises with other course work.

During such exercises, we tried to model what we wanted - enthusiasm, creativity, courtesy, and focus. In early classes, we would moderate and critique the student-driven process. For example, we might lighten a student critique by restating it and adding a compliment, or draw attention to some aspect of a presentation (good or bad) that we thought had been overlooked. In later classes, we frequently felt (wonderfully) superfluous as the students drove the classroom, with occasional reminders from us to move on to another topic.

2.1 Reflecting and Iterating

This course was our (and our department's) first use of reflective journaling with feedback. Reflection on learning is a pedagogical technique used to promote the higher order cognitive skills of analysis, synthesis, and evaluation, i.e., critical thinking [14]. We led discussion on what constitutes a good journal entry, and these reflective entries helped us monitor (and eventually evaluate) student experience, learning, and the quality of the environment. Giving brief feedback to entries provided another tool to guide students in their team relations and technical learning. Many students felt overwhelmed early on by the burden of discovering knowledge for themselves; while teams mitigated this burden, it was clearly a novel experience for many. Journal feedback gave us a way to acknowledge and support stressed individuals without having them feel exposed in the classroom setting.

This was the first course that these students had taken where they were given the opportunity to receive critiques of their ideas and projects as they were being developed and then revise their project based on the feedback. In this way, they watched their product take on new forms based on feedback of others and were then given the opportunity to show off their latest and greatest creation. They often commented on this iterative process and the feedback they received in their reflective journals.

2.2 Replacing Lectures and Textbooks

Our goal was for students to gain the desired knowledge and skills through discovery and collaboration. Class time was utilized for a number of active learning activities to foster discovery.

In a typical class, we began with progress reports from the teams on assignments. Teams would share successes and challenges, with other teams (and sometimes the instructors) offering kudos, suggestions and alternatives. Then a team would make a formal presentation of their learning from a previous assignment, and the class would 1) discuss the presentation, and 2) discuss the concepts. We found it was crucial to give discussion time to the presentation to establish high standards. Without such standards, other students could not be expected to learn from the presentation. (That said, since peer-teaching is highly effective [8], presentations below the standards of a professional teacher may produce learning when delivered by a peer.) Towards the end of class, a new assignment would be created, either through students deciding what knowledge they needed next (in the process of creating an activity for the XO) or the instructors providing a path.

While we had a general list of Python, the XO, and robot concepts that we wanted the students to learn, we did not plan in advance the ordering of topics or the specific assignments. The instructors met after every class to decide on scope and overall direction of the next stage. Students developed the actual assignment details. For example, in the first week, we moderated brainstorming of every feature of Python they would need to learn to program a networked, GUI game; then teams chose which features they would learn and present to the class and arranged the schedule of presentations. Sometimes all teams had the same assignment, while other times each had a separate topic.

This classroom environment integrated studio-based learning, problem-based learning, and reflective learning. It also developed communication skills, and team skills, and independent critical thinking. Preparing and giving presentations of concepts and ideas also reinforced the presenter's understanding of the concepts.

2.3 Student Choice and Helping Others

Research shows that students are engaged through choosing their own projects of interest to them [9]. We took this to the extreme, providing specifications in terms of each new concept they needed to explore. Students were then charged with demonstrating their knowledge and skills with respect to that concept. We left the means for their exploration open-ended. Each team decided how they wanted to gain that skill or knowledge, what program to write to demonstrate it, and how they would present it to the class.

Students were encouraged to find information on the Internet, in books, and especially from other teams. One assignment that turned out to be quite difficult was writing a game that used the XO's built-in mesh networking hardware. Online instructions worked only for simple examples. Two teams working at the same time realized the problem was far beyond what they had solved before, decided to join forces, and reconvened in one room until they started to make progress. All students agreed later that without team support, the frustration would have been overwhelming.

Since the available resources for students did not directly address many of their questions, a key task we set for students was to create a shared, living wiki document that would help future students learning how to program the XO laptops and robots. Many conversations surrounded thinking about how they could help future students by documenting some nugget of knowledge that they had learned through time-consuming exploration and experimentation.

Similarly, we directed students to think about how their knowledge could be used to help teach youth how to program a robot or how they could create a learning game for young children. They were concerned about how their final project, a learning game of their design, would be used

Table 1: Student evaluation of each course component in achieving course goals. Students used a four-point rating scale. The first number shown is the percentage of students who chose the highest rating; the second number is the percentage of students who chose either of the two positive ratings. For example, 89% of students said Collaborative Teamwork was *much more* effective at causing learning than previous course experiences; 100% of students said Collaborative Teamwork was either *much more* or *somewhat more* effective at causing learning.

Course Component	Course Goals							
	Learning		Enthusiasm		Technical Abilities		Communication Skills	
	highest	positive	highest	positive	highest	positive	highest	positive
Collaborative teamwork	89	100	100	100	44	88	67	100
Reflective journaling	22	78	11	55	11	67	44	100
Documentation	56	100	11	89	33	89	44	88
Continuous revision	89	100	100	100	78	100	44	77
Writing learning games	67	100	78	100	67	100	33	89
Open-ended project	89	100	100	100	78	100	44	88
Weekly presentations	78	100	33	100	56	100	67	100
Working with robots	56	100	56	100	33	100	44	77
Using Internet resources	56	100	56	100	44	100	33	66

by the young students and their teachers. This motivated many interesting discussions on how learning happens, giving these students their first taste of service learning[1].

Thus, students actively assisted others in four distinct ways: helping with tasks across teams; helping other teams through discussion and critique; helping peers taking the course in the future; and helping the intended client of their software (Service Learning). Students really enjoy making a contribution [3]. Each of these kinds of help was cited by students as a motivating factor and, we believe, contributed greatly to the success of the course.

3. EVALUATION

We evaluated the effectiveness of the course components through student post surveys and analysis of individual student journals.

3.1 Methodology

An assessment professional from University of Delaware Office of Education Assessment developed and implemented a student post survey with the goal of collecting the students' opinions of the impact of each novel course component on four course goals: increasing student learning, enthusiasm, confidence in technical abilities, and confidence in communication skills. Students rated the effectiveness of each component relative to their previous course experiences, using a four point scale (no middle) for each goal.

Three additional open-ended questions were directed towards gathering student opinion of the importance of a final, formal presentation to an audience of faculty, the effect of frustration on learning, and what the students would or would not change about the course.

The education assessment professional also developed a rubric for evaluating journal entries. The rubric measured the degree of reflective learning content and emotional content. For each student, the rubric was applied to 2-3 early entries (called pre-data) and 2-3 late entries (called post-data), based on the volume of material.

Threats to Validity: The evaluation results are based on 9 respondents to the survey. The class had twelve students whom we hand-picked from 26 based on their application letters and their transcripts. Thus, the results here may not translate to a larger course, or to a course which admits all students[2].

3.2 Post-Survey Results

Table 1 shows data in response to the question "Compared to your previous experiences, how effective were the following components of the class in making you more X where X is indicated by the column title — Learning: helping you learn, Enthusiasm: enthusiasm about your program/field, Technical Abilities: confidence in your own technical abilities, and Communication Skills: confidence in your communication skills. The entries are the percent of student respondents who labeled each as *much more* and *somewhat more* effective than previous CS courses, respectively, for each course component.

Most importantly, all student respondents rated all 12 of the 13 pedagogical components as *much more* to *somewhat more* effective in helping them learn compared to previous experiences. The only exception was reflective journaling which received *much more* to *somewhat more* effective in 88% responses. Of particular note is that collaborative teamwork, opportunity for continuous revision, and open-ended project were ranked *much more* effective at helping students learn by 89% of respondents. These same three components had 100% respondents stating they were *much more* effective than prior experiences in making them more enthused about the program/field. All respondents ranked all components *much more* to *somewhat more* effective than

[1]The course has since morphed into a full service learning course.

[2]While these results are representative of only this selected class, we have since successfully used similar methods to teach an open version of the class.

previous experiences at increasing enthusiasm, except reflective journaling and documenting process during the project.

All students perceived all components except collaborative teamwork, reflective journaling, and documenting the process as increasing their confidence in their own technical abilities *very much* to *somewhat*. These three components were rated as increasing confidence *very little*. The components ranked as increasing confidence *very much* by the highest percent of students were opportunity for continuous revision and open-ended project. There were more mixed results for the components' effects on increasing confidence in communication skills. However, collaborative teamwork and weekly presentations had the most positive responses of *very much* impact by 67% of respondents. It is also noteworthy that 78% respondents ranked weekly presentations as *much more* effective at helping them learn compared to previous experiences.

Across course goals, the students placed the highest value on being able to continuously revise a project, having a project with an open-ended design, and working in teams. We had expected that students would love to design and extend their own projects, but were surprised at the value they placed on being able to iteratively improve the design and code as they learned new skills. Working in teams was slightly less highly-rated for generating enthusiasm, but more highly-rated for increasing communication skills.

Student ambivalence about weekly presentations is apparent, as most students rated it highly for learning, but few rated it highly for producing enthusiasm. It is interesting that students were able to separate their lack of enthusiasm from their judgement about the learning value.

Perhaps not surprisingly, reflective journaling and writing documentation were not highly valued by the students. However, we feel these were invaluable to the teachers as we monitored student and team progress. Furthermore, we suspect that writing about experiences helped students clarify their thinking; more than once we heard a student express an idea, opinion, or emotion in class that had first appeared in a journal. Also, while we responded to the journals, we are not trained in doing so, and better responses might increase student opinions of the journal process.

The open-ended questions in the post survey confirmed that students thought presentation was important. Overall, students enjoyed the final formal presentation to CS faculty, providing them an opportunity to summarize their experience and see how far they had come:

> It forced us to reflect on everything we learned and really analyze what worked and what didn't.

> It not only is a final goal to work towards but it actually makes you feel like you have done alot by filling up this presentation with your work.

> It ties up everything we did, as well as reflecting on what we spent our time on.

The second question was: "Working with unfamiliar tools like the Myro and OLPC can be frustrating. How do you think the frustration you met in this course affected your learning?" Many answers indicated that teams helped alleviate frustration:

> I think that when we got frustrated, it was nice to have other team members (and other teams) who were there willing to help. And even if you were beyond your limit, and felt you had nothing left to give, knowing that your team needed you meant to just give it one more shot.

and that frustration could actually be valuable:

> The frustration is actually what helped us learn more. We had to go research and play around, trying different things to fix the problem instead of just breezing right through it. I feel I'm more likely to remember something I struggled on than something I just copied or read somewhere.

The final pair of open-ended questions asked what part of the course students would change/not change. No consistent answers appeared in what students would like changed, but two course components were consistently cited as "keepers": teams and the open-ended project design.

3.3 Results from Journal Analysis

The first rubric counted occurrences of student reflection about the learning process and/or connection between activities and learning versus merely listing activities and challenges without reflecting about the learning process. The post data shows 82% of the student journals exhibited evidence of reflection about the learning process beyond listing activities and challenges. While 36% of the early journal entries contained only listings of activities; this reduced to only 18% at the end of the semester.

The second rubric focused on how excited and curious about the project and/or confident the student sounds in his/her knowledge and abilities, based on their journal entries. The post data indicates that 91% of the student journals revealed the student was excited or somewhat excited and curious about the project and/or confident in his/her knowledge and abilities. The remaining 9% of students did not share challenges and successes in their journals. 27% of student journals at the start of the semester showed fear of the challenges in the project and/or appeared not confident in his/her knowledge and abilities. No student journals showed these fears in the later journal entries.

Some notable excerpts from the journal entries that show the student perspective about the pedagogical approach and their learning:

> Over the course of 5 weeks I was able to learn far more than I have before in a semester long course. I really enjoyed learning a language and information on my own. It was cool to try and look at online documentation and then internalize that knowledge to make something of my own.

> This course is unlike any other course I've taken, it is very different to have the professors treat the students as equals and expect to learn from the students. At the same time it is a very mature learning environment, I feel like a job in Computer Science would be a lot like this course in the sense that both superiors and employees would work together to tackle the problem presented.

> I feel that this course, as is, is a great aspect of my learning experience in regards to computer

> science - especially software engineering. A professor can lecture all night and day over how you should code and why, but until you find yourself going back and trying to reapply your code to a new system, or trying to read someone else's code, you don't fully grasp the importance of software manageability.

4. RELATED WORK

Our course used techniques that have been demonstrated separately to provide successful learning outcomes in college courses. These include aspects of Problem-Based Learning [4], critiquing in Studio-Based Learning [7], and Active Learning [12]. The iterative approach to software engineering courses has been used by a variety of universities. Anewalt and Polack-Wahl [1] combined it with dynamic groups, showing that iterative development allows students to learn from previous iterations and incorporate feedback into the next iteration, while rotating the groups increased documentation quality and increased the project completion rate from 30% to 80% in a semester. Some of our course components, particularly iterative revision and team work, are inherent in extreme programming. In a study of MS students in a course project using the extreme programming model, Loftus and Ratcliffe [11] concluded that students' accuracy in estimating and planning were improved and students believed they learned new technologies faster.

A study of reflective journals in a computer science course by George [6] found that students who liked the reflective journal reported benefits to their learning, including understanding of concepts, identifying what they don't understand, problem solving in practical programming exercises and awareness of approach and attitude to learning. Perschbach [14] evaluation of student reflective journal blogging by community college computer science students showed that blogging effectively recorded critical thinking and provided the learner with a personal voice that created a sense of ownership of ideas, active participation and empowerment in personal learning, and a contribution to the collaborative learning effort.

Our course provides students opportunities to help others in several ways (section 2.3). The EPICS project at Purdue University is a well established example demonstrating the benefits of service learning in a computer science and software engineering education [10]. Others have more recently developed different models of service learning in CS courses [15, 5].

5. CONCLUSIONS

In the course of a 5-week class, our students taught themselves (and each other) a new language, new OS, GUI programming, and simple networking for collaborative games. They also learned communication, negotiation, collaboration, presentation and teamwork skills; and project design and iterative development. Students reported learning at a greater rate than in other CS courses while maintaining (and in some cases acquiring) a high level of enthusiasm and confidence in their abilities. They were motivated to achieve through their own desire to help each other, future students, and the children who would use their games. The instructors practiced a variety of techniques which they have since used in their regular semester courses.

6. ACKNOWLEDGMENTS

We thank Gordana Copic at University of Delaware Office of Educational Assessment for her efforts on evaluation.

7. REFERENCES

[1] K. Anewalt and J. A. Polack-Wahl. Teaching an Iterative Approach with Rotating Groups in an Undergraduate Software Engineering Course. *J. Comput. Small Coll.*, 25:144–151, June 2010.

[2] D. Blank. Robots Make Computer Science Personal. In *Communications of the ACM*, volume 49, pages 25–27, December 2006.

[3] D. Cone and S. Harris. Service-Learning Practice: Developing a Theoretical Framework. *Michigan Journal of Community Service Learning*, 3, 1996.

[4] B. J. Duch, S. E. Groh, and D. E. Allen, editors. *The Power of Problem-based Learning: A Practical 'How To' for teaching Undergraduate Courses in Any Discipline*. Stylus Publications, 2001.

[5] M. A. L. Egan and M. Johnson. Service Learning in Introductory Computer Science. In *Fifteenth Annual Conf on Innovation and Technology in Computer Science Dducation*, ITiCSE '10, pages 8–12, 2010.

[6] S. E. George. Learning and the Reflective Journal in Computer Science. In *Twenty-fifth Australasian Conference on Computer Science - Volume 4*, 2002.

[7] D. Hendrix, L. Myneni, H. Narayanan, and M. Ross. Implementing Studio-based Learning in CS2. In *SIGCSE '10: 41st ACM Technical Symposium on Computer Science Education*, pages 505–509, 2010.

[8] G. D. Kuh, J. Kinzie, J. A. Buckley, B. K. Bridges, and J. C. Hayek. What matters to student success: A review of the literature. Technical report, National Postsecondary Education Cooperative, July 2006.

[9] G. Lewandowski, E. Johnson, and M. Goldweber. Fostering a creative interest in computer science. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, SIGCSE '05, 2005.

[10] P. K. Linos, S. Herman, and J. Lally. A Service-learning Program for Computer Science and Software Engineering. In *8th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '03, 2003.

[11] C. Loftus and M. Ratcliffe. Extreme Programming Promotes Extreme Learning? In *SIGCSE Conf on Innovation and Technology in Computer Science Education*, ITiCSE '05, 2005.

[12] S. Ludi, S. Natarajan, and T. Reichlmayr. An Introductory Software Engineering Course that Facilitates Active Learning. In *36th SIGCSE Technical Symp on Computer Science Education*, 2005.

[13] olpc Foundation. One laptop per child. http://one.laptop.org/.

[14] J. W. Perschbach. *Blogging: An Inquiry into the Efficacy of a Web-based Technology for Student Reflection in Community College Computer Science Programs*. PhD thesis, Nova Southeastern Univ, 2006.

[15] J. Tan and J. Phillips. Incorporating Service Learning into Computer Science Courses. *J. Comput. Small Coll.*, 20:57–62, April 2005.