Integrating Hard and Soft Skills: Software Engineers Serving Middle School Teachers*

Richard Burns, Lori Pollock, and Terry Harvey
Computer and Information Sciences
University of Delaware
Newark, DE 19716
{burns,pollock,harvey}@cis.udel.edu

ABSTRACT

We have developed and implemented, over four semesters, a model for engaging computer science majors in service learning for teachers of grades 6-8 at a K-8 school in an underserved community. This paper describes the design of a course focused on interweaving software engineering practice, service learning, and development of "soft" professional skills. CS student teams partner with middle school teacher teams to create learning games for classrooms, and then conduct classroom instruction and observation. We report on our results from evaluating the experience of CS students and middle school teachers through pre-post surveys, evaluator observation of student demo presentations and classroom instruction, focus groups, and student reflective journals.

Categories and Subject Descriptors

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

General Terms

Human Factors

Keywords

service learning, software engineering, soft skills, studio-based classroom, reflective journaling, problem-based learning

1. INTRODUCTION

We are embracing the challenges in broadening participation of computing in middle schools of underserved communities as opportunities for university computer science students to develop valuable technical and communication skills. Our central theme is that middle school teachers can be significantly supported in their quest to integrate computing into their teaching with the help and role models of carefully mentored undergraduate software development teams, who in turn gain from collaborative learning [1, 8, 13], software engineering practice with a real client [12, 11], peer leader experiences [6], and an opportunity to experience first-hand the potential impact of their computing expertise in helping others [5, 3]. The service experience may in turn improve underrepresented participation in computing at the college level [9].

The authors have partnered with a local, 2400-student, K-8 school in a city with one of the highest crime and poverty rates in Pennsylvania. The partner school received a generous donation of XO laptops designed for the One-Laptop-per-Child program which targets the world's poorest children [10]. We have developed a pilot program that leverages the low-cost, networked XO laptops to help their teachers integrate computational thinking into middle school math, science, language arts, and social studies in a sustainable way.

The program is highlighted by: (1) development of *computational thinking activities* and *increased collaborative and differentiated learning opportunities* to enable computing-based learning in middle school, (2) a *service learning* course for CS majors, with *collaborative development teams* matched with the teachers and led by *peer leaders* to develop XO learning activities and apply good *software engineering practice* with *a real client*, and (3) continuous *software and professional development customized to teachers' student learning goals*, with biannual teacher-focused workshops for training and collaborating. This paper describes the first two of these efforts.

We have developed a service learning course in computer science, with the goal of engaging computer science majors (sophomore or later) by partnering with middle school teachers to integrate XO-based computing into their teaching. Our goals for using service learning with undergraduates are: to effectively learn fundamentals of software engineering via a service context; increasing overall learning, enthusiasm, confidence in technical abilities and communication skills; and delivering a final educational software product that incorporates learning components and is useful to the partner school. We also strive to create awareness about the challenges and strategies of technology in education, the role and impact of assessment in education, and the range and difference of learning styles and how to target each through soft-

*This material is based on work supported by the National Science Foundation Grant CNS-0940501. Thanks to our evaluators Kathy Pusecker and Manuel Torres, learning specialists Cecily Selling and Jon Manon, and service learning coordinator Sue Serra.

SIGCSE'12, February 29–March 3, 2012, Raleigh, North Carolina, USA.

ware. Service learning participation has been shown to have a particularly positive impact on minorities and women [9].

We pair student teams with middle school teachers (the clients) to develop educational software. These teams are guided by "peer leaders" who have taken the course previously. The partnership targets middle school students (grades 6-8) and math, science, language arts, and social studies teachers. While our students work closely with the teachers, middle school students will be the ultimate users of the learning software, providing feedback directly to our students and through teachers on the developed software, noting robustness, usability, user-friendliness, and fun. Specifically, the goal is to design a piece of educational software which both facilitates learning and also is "replayable". The authors, as instructors in the course and supervisors of the partnership, are committed to steering both the CS students and middle school teachers away from projects that do not incorporate educational learning components in their final product. For example, projects such as building a website, creating an automated quiz grader, or facilitating web-based testing were discouraged. As a result, we believe that our course is primarily service learning for education.

We have taught four semesters of the course (Fall 2009, Spring 2010, Fall 2010, Spring 2011) so far. This paper describes the course components, logistics and lessons learned, evaluation, and guidance to other instructors interested in developing a similar course at their institution.

2. COURSE COMPONENTS

Table 1 shows the general timetable for organizing the three major components of software engineering, service learning and soft skills lessons into a 14-week semester course. Both service learning and teaching of soft skills are interleaved with the software engineering process. Attention is paid to each major component during every week of the semester. We use pedagogies that we found to be successful in our earlier independent study course: collaborative teamwork, open-ended project, weekly presentations, continuous revision, and reflective journaling. Each subsection details some uses of these pedagogies for providing a practical software engineering experience, service learning with an educational partner, and development of soft skills along with technical skills.

2.1 Software Engineering

Our main software engineering goal is to provide students with a practical experience in the software engineering process including client introduction and requirements gathering and analysis, software design creation and critique, iterative design, implementation, testing, client use and feedback, and revision [7, 12, 11]. Our first activity is to form the CS student teams to be matched with middle school teachers. Each semester, we have had enough students to create 3-5 teams of 4 students each. Students complete a questionnaire on the first day of class which we use to construct teams based on schedule availability and balanced by programming experience, CS background, confidence, and self-rating of communication skills.

Students are presented with the challenge of designing, implementing, and deploying a learning game to meet their teachers' objectives for the middle school students in mind. The first challenge is learning the XO platform, and how to program in Python for the XO, including GUI programming and working with a database on a remote server. Students complete coding assignments and help each other master the intricacies of programming interactive software with Python on the XO.

Student teams take responsibility for coding and presenting various aspects of Python to each other. Discussions of advanced Python language features which are critical to master in order to provide youth-friendly user interfaces and exploit the collaborative capabilities of the XO laptops (such as GUIs, animation/motion/display buffering, events, etc.) are typically augmented by the instructors to ensure that they are discussed in enough detail. We also discovered that a review of OO design principles is helpful at this stage. Python learning is capped by reviewing the previous semesters' products and fixing any bugs found after release.

We discuss requirements gathering and analysis, storyboarding, and how to formulate and utilize use cases, as an initial step in software design and engineering. The design presentations and critiques integrate studio-based learning into the classroom environment. Teams present their hand-drawn storyboards, example screen shots, and use cases portraying the proposed flow of their game to the class. The presentations also focus on how learning at higher levels of Bloom's taxonomy will be incorporated into the game. The student audience then critiques (and compliments) the presentation and brainstorm ways to improve the game. After initial rounds, we invite an outside panel to offer additional perspectives from their collective technical experience at the K-12 levels in differentiated instruction and incorporating learning into pedagogies. This iterative development and critique continues throughout the semester.

Following several teacher (client) meetings, prototypes, and iterative development and critique, the implementations evolve to alphas and then to betas. The betas are first tested by the CS instructors and the other teams. We provide feedback, and minor revisions (hopefully only minor!) are determined and implemented given the time constraints. This effectively becomes the final version of the learning game due to the semester constraints. A final presentation and demo to invited guests, and partner school classroom distribution and in-situ observation conclude the semester.

2.2 Service Learning

An unusual aspect of our course is that it integrates a software engineering learning experience into a service learning experience to benefit an underrepresented group. This provides our undergraduates with multiple clients (middle school teachers and students) and the opportunity to see their final product deployed in a classroom, all in a learning context outside of the university.

At the start of the semester, we discuss the XO Laptop program, our partnership with the middle school, and assign readings from popular media about the current successes and failures of the XO Laptop program worldwide. Our Service Learning office facilitates discussions on service learning features and pitfalls, and reviews preconceptions about middle school learning environments and underprivileged communities.

The student teams take their first trip off-campus to observe the middle school classroom using the XOs and meet privately with their teacher team. The CS team objective is to gain an understanding of the learning challenges and context in the partner school. The CS students must con-

Table 1: Week-by-week timetable for Software Engineering (SE), Service Learning (SL), and Soft Skill development (SS) over 14-week course with partner school (PS)

Week	**Software Engineering (SE)**	**Service Learning (SL)**	**Soft Skills (SS)**
1	familiarize with XO laptop and existing software; form teams	obtain competence in PS' computing technology; discuss and begin journaling	introduce students, peer leaders, and team members; discuss teamwork, peer review, critiquing
2,3	research programming concepts that are believed to be important; teach each other main Python programming language features, and how to bundle and distribute "activities" on XO laptop	introduce context of XO: One Laptop Per Child (OLPC); discuss potential cultural differences in community partner (PS); discuss middle school learning environments	prepare for first teacher meeting; facilitate discussion about communicating w/non-tech; goals/strategies for meeting; team presentation of Python features
4,5	demo advanced Python programming (gui, graphics, animation, events); server-client protocols and databases; IDE integration; versioning; review OO design patterns	observe PS students using XO; meet PS teachers and gain understanding of PS students' learning challenges; report out on teacher meetings; follow-up with email to teachers	thank and politely request follow-up meeting with community partner; collaborate and communicate as a team
6,7	gather and analyze requirements; present software design as storyboards and use cases; iterative redesign	brainstorm initial game ideas; understand that individuals have different learning styles; map designs into Bloom's taxonomy; identify state requirements which guide PS lessons	give and receive peer critique; write reflective journal entries on PS and team experiences; discuss how to demo software, communicate and sell ideas to teachers
8,9	develop prototypes; test and iterate; demo alpha version (on emulator); consider change from client feedback	schedule and conduct design critique meeting with teachers; review feedback from PS teachers	prepare and present a report in class that emphasizes teams' proposed handling of teacher feedback
10,11	test; present progress reports; discuss remaining challenges; critique peers; demo beta version	schedule final demo and classroom observations with teachers; develop observation goals	prepare and present a demo with a progress report on the status of their learning game
12,13	test peers' games and submit feedback; debug minor bugs; port; overcome hardware issues; fine-tune final version (on XO laptop)	demo final version to PS teachers; deploy final version in classroom and observe its use by PS students	collaborate with entire class to prepare an overall class-wide project presentation; rehearse to clock
14	share with CIS faculty and peers; package final version for future distribution by others	report out on classroom observation experiences in class; bundle final product with documentation to PS teachers	present in a formal presentation forum to a strongly technical audience

sider the objectives and challenges of both the middle school teachers and their students during the design phases; both audiences are critical to a successful product.

During the brainstorming and software design phase, we invite a learning specialist from a local K-12 school to explain that individuals can have different learning styles (e.g. visual, auditory, kinesthetic) and that an educational game needs to consider the strengths and limitations that a user may have. We also discuss Bloom's taxonomy [2] of learning objectives: remember, understand, apply, analyze, evaluate, create. We are interested in creating learning games that reach the analytical and evaluation objectives, and we discuss this with our students. To date our products are mostly skills practice (apply). This is partially due to the difficulty of designing and making fun analysis/evaluation games in ten weeks.[1]

A major objective of our partner school is for their students to demonstrate competence in learning areas via a strong performance on the state assessment exam. Thus undergraduates spend some time researching the state assessment exams and understanding the concepts they cover, so that our brainstorming can lead to a learning game that caters to multiple learning styles, reaches as many learning levels as possible, and is conceptually relevant to the state assessment for the targeted grade level.

[1]We are improving in this regard, and will report in a future paper.

2.3 Soft Skills

A significant component throughout the course is "soft skills" development. Students learn how to handle team management issues, effectively present their design ideas, carefully and constructively critique others' ideas, conduct useful and persuasive demos of their software, good reflective journal entries, and communicate with a non-technical teacher client and middle school students living in a high crime, high poverty community.

We use various techniques to increase the self-awareness of our students' soft skills and challenge them to improve. For example, each semester students typically have the false impression that they can prepare minimally and still successfully initiate and lead a meeting with their teacher team to brainstorm a semester project. However, before their first teacher meeting, each student team is put into a role play situation where the authors play the non-technical middle school teacher who is quite knowledgeable about their teaching and middle school students. During the role-play, most students typically stumble on: facilitating an introduction or ice-breaker, communicating the goals of the partnership and the current meeting, avoiding computer science lingo and acronyms, and/or establishing a back-and-forth dialogue to brainstorm educational learning game possibilities. This role play takes many of our students outside of their comfort zone. We then assist students in identifying a list of goals for the initial meeting, and a corresponding set of strate-

gies to achieve the goals. We then have a series of smaller role-plays to practice specific techniques.

We also dedicate several class sessions preparing for the client meeting, culminating in a one-page summary of the objectives and lists for requirements gathering during the meeting, and a CS team profile document to leave with the teacher. Students confidently leave their initial teacher meetings with an understanding of the teacher's goals, a sense of where the middle school students struggle in learning, a broad idea of the lesson schedules for the year, and ideas on how technology could be incorporated in the classroom. Back at the university we evaluate the meeting in terms of goals met.

Students also hone their communication skills as they work in teams, present their ideas and projects to the whole class, give and receive critiques, write in their reflective journals and demo their software to assorted audiences. They practice how to communicate technically as well as non-technically, and it is fun for us to watch the dramatic improvement over the semester. For many of our students, this is the first time they have discussed what makes communication successful, and the course emphasis reinforces the importance of building good communication skills.

3. COURSE LOGISTICS

3.1 Class Time, Assignments and Assessment

We have varied the weight of each grade component over time, but the over-riding importance of the project has been constant. Weekly project presentations are an excellent opportunity for feedback from peers and faculty. In our experience, peer feedback is much more powerful – students work much harder to please their team and peer audience than they would work for a grade from faculty. These widely varied components all contribute to the success of each team and their project; if any piece is done poorly, their entire project suffers.

60% Learning Game Project.

Weekly Presentations: Each presentation begins with the goal slide from the previous week, along with a list of successes, challenges, and a DONE/NOT DONE rating. This style makes it fairly simple to assign intermediate project grades to software engineering deliverables, e.g. storyboards, code, tests. After discussion of successes, challenges, and decisions, new goals are set for the following week. The quality of the presentation and participation are also noted.

Customer Relations: We use subjective evaluation by the faculty on a team's work with the middle school teachers.

Peer Evaluation Factor: Individuals use an online questionnaire to evaluate themselves and other team members along six continua: contributing in meetings, team communication, timely task completion, design and correctness, doing their share, and overall. Individual ratings are averaged and a zero-sum project grade factor is calculated for each team member relative to the team average rating. Thus a strong team member could be assigned a factor of 1.10, while a weaker team member could receive a .90 factor. Faculty reserve the right to intervene if these factors seem out of line, but we have not yet had to do so.

15% Class Participation. Participation is crucial to the design of the course, so students are held accountable for contributing to e.g. discussions, role plays, and critiquing sessions. Intermediate feedback about this grade can be delivered in journal comments or directly. This is a subjective grade assigned by faculty in consultation with peer leaders.

15% Reflective Journal. An important component of service learning is directed-reflection, which connects what students learn in the classroom with their experience of real-world conditions [3]. We have found that students need time to reflect on what they are experiencing, as most are being put into situations very different from their own experiences. Reflection on learning is also a pedagogical technique used to promote the higher order cognitive skills of analysis, synthesis, and evaluation, i.e., critical thinking.

Students maintain a directed reflective journal as a private wiki. We also lead discussions on effective journaling. Students are given a set of default questions to structure their entries ("How well is your team working together?"), and additional journal questions are communicated during class based on discussions or presentations ("Explain three aspects of this CS course in terms of Bloom's Taxonomy."). For full credit, students write two entries per week. Peer leaders, and sometimes faculty, read and comment in private journals as a means of delivering feedback and encouraging meta-analysis of individual, team, and class performance. Journal feedback gives us a way to acknowledge and support individuals outside of the classroom. This is a subjective grade assigned by peer leaders in consultation with faculty.

10% Quizzes and Labs. We give a small set of quizzes on programming topics, software engineering, and other discussions from class, e.g. learning styles and Bloom's taxonomy, and small labs on Python. These help students gauge their progress, and can serve as an indicator for instructors trying to understand team dynamics.

3.2 Partnering with the School

Working with middle school teachers has been enlightening. Busy teachers with established curricula and very full schedules are still willing to spend extra time to have our students visit. Our initial contacts with the partner school were all through the administration. Administration was protective of their teachers and did not want, for example, to have us peppering them with email. Communication turnarounds were sometimes two weeks. Once we established relations with individual teachers, communication sped up significantly (but only rarely is within 24 hours).

The middle school teachers do not have much experience with educational software outside of quizzes, so all of our early projects were skills tests that assigned a numeric grade. To change this, we taught our students that they would have to *sell* their more interesting project ideas to the teachers. This resulted in more interesting meetings and software.

Each CS student visits the partner school 3-5 times. Scheduling visits is tricky given students' and teachers' conflicting schedules. Our class does not have the benefit of any "open" service learning day. Driving time to the school is an hour round trip, and the trip would probably not be worthwhile if meetings and observations were less than two hours on-site. This means that each trip is at minimum a three hour commitment. Students car-pool each other[2]. Early visits are typically by the whole class, while later visits are scheduled by teams with their teachers. Having teams negotiate meetings with their teachers adds a level of professionalism to

[2]In one semester, visits were so difficult to negotiate that project teams were formed around carpools.

the process. Sometimes multiple teachers are willing to provide more critique than an individual teacher at the client meetings, especially when the ratio of teachers-to-students is closer to 1:1.

4. EVALUATION

The program evaluation was designed to measure how well the program goals were met each semester, as well as to provide qualitative feedback on the course for continuing improvement.

Program Goals and Expected Outcomes. Our goals for the CS students each semester were to (1) increase knowledge/experience in software engineering, (2) build team management skills, (3) increase technical knowledge of the discipline, (4) improve problem solving skills, (5) improve communication skills with non-technical clients, (6) design and implement a project addressing a community problem, (7) increase enthusiasm for CS, (8) increase awareness of potential impact of applying technical expertise, and (9) increase confidence in ability to succeed in CS.

Evaluation Instruments. The University's Office of Educational Assessment (OEA) developed evaluation instruments and collected and analyzed data for the surveys. This included pre and post surveys of the computer science students enrolled in the course each semester. OEA also attended the final demo presentations of the learning software created by the students each semester and wrote reports on their observations of the students products and informal conversations with them. OEA also joined the students a few times in their middle school classroom instruction sessions and provided a report on their observations of the middle school students, CS students, and teachers during those sessions. OEA also conducted a focus group study in Spring 2011 of the students who participated in the course in the third semester, Fall 2010. The instructors monitored and analyzed the reflective journals of the service learning students. In addition, CS students completed peer evaluations of their team members periodically throughout the semester.

Results and Discussion. Table 2 shows results from pre-post surveys for the last two semesters.

Software Engineering Skills - Questions 4 through 7 are targeted at gaining student perceptions on their software engineering skills before and after this course experience. For question 4, "teach myself a new programming language", there was an in increase from 89% to 100% in "strongly agree" and "agree" responses. For question 7, "be able to develop software to meet a client's needs", there was an in increase from 68% to 95% in "strongly agree" and "agree" responses. Interestingly, the percent of students who "strongly agreed" and "agreed" that they would work well with their peers decreased from 95% to 85%. This may reflect students' first experience with a team and a large project[3]. For question 5, "be able to talk to and question non-computer science clients to meet their needs", there was an increase from 95% to 100% in "strongly agree" and "agree" responses, and the percentage who responded "strongly agree" increased 39%.

Two students wrote in their reflective journals:

> "It was also interesting to work with the teachers because it added a whole new dimension to programming that I'm not used to. Some of our work was scrapped ... I guess that just shows what it would be like in the real world."

> "Storyboarding the game is pretty interesting. I've never done anything like that before, and its sort of fun. Plus it really makes you think about what feature you are going to have, and which ones you will have to cut. Its also a challenge to figure out how you are going to transition smoothly from one aspect to the next, and arrange everything on one screen."

Service Learning and its Impact - Questions 8 are 9 are targeted at gaining student perceptions on working for or seeking paid work for a non-profit. The results are mixed. While the percentage of "strongly agree" to "agree" responses decreased between the pre and post survey, the results for "strongly agree" increased.

Overall, the undergraduate students stated they had a positive experience working with their assigned teachers. Our evaluator reported these responses from CS students at the final demo:

> "These CS students reported that it was easy to develop a rapport with the teachers since they were only a few years older than the CS undergrads. They reported that the teachers were excited to work with this team. The CS students said the teachers had many ideas and informed this team specifically of the area of study (math, such as fractions and decimals) they wanted their game to be based upon, and the skills the game would be used to enhance."

> "The CS students said interacting with teachers was a lot easier than they thought it would be. After the initial contact, they felt comfortable in communicating with the teachers."

> "This team said it was easy to talk with and interact with their specific teacher. This CS student said that although his teacher was not "tech savvy" she was still actively engaging with CS undergrads on this project."

In a May report from the evaluator's interviews of CS students, she reports: "All but one [student] reports feeling like they were helping a community that needs their skills and that made them feel good about themselves. All but one reported feeling like they would continue to think about volunteering their skills."

Soft Skills - Questions 1, 2, and 3 provide insight into both service learning impact and soft skills development. Overall, the experience with working with teachers actually decreased their confidence in teaching teachers how to use laptops in the classroom. This may reflect the fact that students weren't expecting to have to instruct teachers in using XO laptops, which use a variant of the GNU/Linux OS. However, their confidence in teaching middle school students to operate laptops with unique software increased from 74% to 90% in "strongly agree" and "agree" responses.

[3]Our introductory courses now incorporate more teamwork.

Table 2: Pre/Post Survey Results from 19 responses and the latest 2 semesters of data. Students responded using a 5-point scale: AA=Strongly Agree, A=Agree, N=Neither Agree nor Disagree, D=Disagree, DD=Strongly Disagree. Out of 100 percent.

Survey Question: I am confident that I can and will ...		AA	A	N	D	DD
1. Teach Middle School Teachers how to use laptops in their classrooms	pre	47	53	0	0	0
	post	35	50	10	0	5
2. Teach Middle School Students to operate laptops with unique software	pre	26	48	26	0	0
	post	35	55	10	0	0
3. Interact with underrepresented middle school students	pre	26	53	16	5	0
	post	30	50	15	5	0
4. Teach myself a new programming language	pre	68	21	11	0	0
	post	70	30	0	0	0
5. Be able to talk to and question non Computer Science clients to meet their needs	pre	26	69	5	0	0
	post	65	35	0	0	0
6. Work very well with my peers to serve a client	pre	16	79	5	0	0
	post	35	50	15	0	0
7. Be able to develop software to meet a client's needs	pre	26	42	32	0	0
	post	45	50	5	0	0
8. Volunteer my computer knowledge to help a non-profit after I graduate	pre	21	42	32	1	5
	post	40	15	40	5	0
9. Seek out paid work with a non-profit business	pre	5	26	53	11	5
	post	20	10	60	10	0
10. Seek out employment with a for-profit business after graduating	pre	42	32	26	0	0
	post	65	25	10	0	0

5. GUIDANCE TO OTHER EDUCATORS

We conclude with what we believe are the key lessons learned from these four semesters:

- The most powerful learning moment in the course is when teams watch real middle school students use their software. Both triumphs and failures are greatly magnified. We found that having this moment early helps teams address issues before semester end.
- In general, soft skills that are explicitly discussed or role-played in class ("How do you introduce your ideas to a teacher?") are vastly more successful than those communicated via lecture.
- Role playing takes time, but is efficient. It forces students to confront their skills, or lack thereof.
- Brainstorming the characteristics of good and bad critique, and then pointing out examples of good critique, helps students learn how to critique. It is important to verbally praise good critiques when they happen.
- Coordinating with teachers is easiest using direct communication between teachers and their matched teams, with oversight from instructors. Students should provide a written list of goals before each interaction.
- Debriefing after community interactions should be done as soon as possible, with full class discussion followed by individual reflective journaling.
- Journaling improves individualized student communication with instructors.
- For the end-of-semester presentation, students should do a timed dress rehearsal and get class feedback.

6. REFERENCES

[1] *Greater Expectations: A New Vision for Learning as a Nation goes to College*. Association of American Colleges and Universities, 2002.

[2] B. Bloom, M. Englehart, E. Furst, W. Hill, and D. Krathwohl. *Taxonomy of educational objectives: The classification of educational goals*. Longman Group, United Kingdom, 1956.

[3] R. G. Bringle and J. A. Hatcher. Reflection in service-learning: Making meaning of experience. *Educational Horizons*, pages 179–185, 1999.

[4] D. Cone and S. Harris. Service-learning practice: Developing a theoretical framework. *Michigan Journal of Community Service Learning*, 3:31–43, 1996.

[5] 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.

[6] J. Liu, J. Marsaglia, and D. Olson. Teaching software engineering to make students ready for the real world. *Jouranl of Computing Sciences in Colleges*, 18:43–50, 2002.

[7] B. J. Millis and P. G. Cottell. *Cooperative Learning For Higher Education Faculty*. American Council on Education, 1998.

[8] W. Oakes. *Service-learning in Engineering:A Resource Guidebook*. Campus Compact, 2004.

[9] OLPC. One laptop per child. http://laptop.org/en/.

[10] J. A. Polack-Wahl. Incorporating the client's role in a software engineering course. *30th SIGCSE Technical Symposium on Computer Science Education*, pages 73–77, 1999.

[11] C. P. Rosience and J. A. Rosiene. Experiences with a real software engineering client. *Frontiers in Education Conference*, 2006.

[12] L. Springer, M. E. Stanne, and S. S. Donovan. Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of Educational Research*, 69:21–51, 1999.