Teaching Communication Skills in Software Engineering Courses

Chang Liu, Karin Sandell, and Lonnie Welch

Ohio University
Athens, Ohio 45701, U. S. A.
{liuc | sandell | welch}@ohio.edu

Abstract

Communication skills are important to software engineers. Yet, this topic is sometimes overlooked in computer science and software engineering curricula. To address this problem, we attempted to explicitly teach communication skills in a software engineering course. We experimented with a number of approaches, including lectures by the instructor, student presentations, mini-lectures mixed with in-class discussions, and other in-class activities such as student-designed scenarios. The results of these approaches were mixed. There were approaches that clearly worked better than one or more other approaches; there were also approaches to which students with different backgrounds responded differently. Overall, after taking this course, students communicated better and were more self-confident in team environments. Our experience shows that with careful planning and innovative pedagogy, we can help our students become both technically competent software engineers or computer scientists, and good team players in the same time.

1. Introduction

The vast majority of software engineers work in teams. To accomplish their tasks, they often need to communicate with technical or non-technical coworkers and clients through in-depth discussions on software requirements, design, and implementation. Clearly, communication skills are an important skill set to software engineers. Yet computer science undergraduate students, many of whom will become software engineers after they graduate, receive little training in teamwork and communication skills, especially in the context of computer science coursework and projects. As a result, many computer science students do not recognize the importance of communication and do not possess satisfactory communication skills. For example, in Spring Quarter 2004, on an anonymous comment card collected from CS456/556, a software engineering course offered at Ohio University, one student complained that: "I don't care for the vast amount of time needed outside of the classroom not working on the 'project' itself." This student was likely referring to the necessary communication with the project customer on project requirements as outside of the scope of the "project itself," and thus not part of the learning experience. Many students compartmentalize their course experiences and thus in this case they are likely to perceive communication skills as extraneous to the subject

"Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition

matter of a software engineering course. Providing an experience that replicates the whole software design process becomes the challenge. The importance of embedding communication across the curriculum in this way and building upon basic skills taught in dedicated communication courses has been endorsed by a growing number of institutions of higher learning that have formalized such programs (see for example, University of Pittsburgh, and their Oral Communication Center http://www.cxc.pitt.edu/).

To address the goals of embedding oral communication toward the end of improving student skills and understanding along with their recognition of the need to study communication topics, we attempted to explicitly teach communication skills in CS456/556. In this paper, we share our experience of using different approaches to cover these "soft" or non-technical contents in a technical course. Section two provides the background of this course. Section three details the different approaches that we experimented with and student responses that we observed. Section four offers concluding remarks.

2. CS456/556 "Software Design and Development"

CS456/556 "Software Design and Development" is a software engineering course with a significant project component and thus is a good choice for embedding communication topics. The class meets four hours each week, either in two two-hour sessions or in four one-hour sessions. CS456/556 mainly covers the Unified Software Development Process[1], the Unified Modeling Language[2], and various software tools, languages, and platforms needed in class projects, such as CVS (a source code version control tool)[3] and SourceForge.net Tracker Tool (an issue tracking tool)[4]. Students form project teams in the beginning of the course and work on quarter-long software projects, in which they must apply software development techniques discussed in class.

CS456/556 is a dual-listed course offered to both senior computer science and computer engineering undergraduate students and first year graduate students. Undergraduate students and graduate students may be mixed in project teams. In the past few offerings, CS456/556 gradually incorporated service-learning projects[5], in which community partners served as external clients for the projects. The open-source software development approach is adopted to facilitate the development and deployment of service-learning projects[6]. Table 1 lists the enrollments and the project clients of the past five offerings of CS456/556. The enrollment numbers show that most undergraduate students were domestic students and that most graduate students were foreign students. In addition, not shown in this table, over 90% domestic undergraduate students were from Ohio.

Table 1. CS456/556 enrollments and projects.

Quarter	Enrollment					Project Clients
	Undergraduates Total	Foreign Undergraduates	Graduate Students Total	Foreign Graduate Students	Total Enrollment	
Fall Quarter 2002	30	2 (7%)	6	4 (67%)	36	No external clients
Spring Quarter 2003	37	1 (3%)	10	10 (100%)	47	An EECS faculty member An EECS graduate student
Fall Quarter 2003	21	0 (0%)	13	11 (85%)	34	An EECS graduate student
Spring Quarter 2004	34	0 (0%)	9	9 (100%)	43	Nelsonville Public Library[7] City of Athens
Fall Quarter 2004	22	1 (5%)	10	10 (100%)	32	Nelsonville Public Library An EECS professor emeritus
Total	144	4 (3%)	48	44 (92%)	192	

3. Teaching Communication Skills in CS456/556

To better prepare future software engineers so that they can quickly become productive in a team environment, we included communication theories and techniques as a formal component of CS456/556 "Software Design and Development." We adopted *Communicate!* [8] as the second textbook of the course and selected topics in four important areas for software engineers to cover in class. The four areas were foundations of communication, interpersonal communication, group communication, and public speaking. Our emphasis was on interpersonal and group communication, because such skills could have a direct impact on the success of student projects. The foundations of communication provided theoretical grounding for the students, in order to broaden their understanding of the whole process of communication and increase the likelihood that they would be able to understand communication situations outside of those discussed in class or in the book. The remaining topics focused on the dynamics of professional interpersonal communication, from dyad to large group, including public speaking skills necessary when communicating in a large group setting.

We felt that among technical courses in our computer science curriculum, this software engineering course was the most appropriate course to cover communication topics because software engineering deals with complete life cycles of large-scale software development and must address issues of communications among team members of a development team and communications between developers and non-technical clients. This observation is concurred in the Computing Curricula 2004 developed by ACM, AIS, and IEEE Computer Society[9], which recommended that the topic of "interpersonal

communication" should be weighted more heavily in a software engineering program than in a computer science program.

We were not alone in integrating communication topics with technical contents. Tapper and Buchanan integrated communication and technical skills in their course *Analog Circuit Computer Simulation* and produced encouraging results[10]. In our course, it was a challenging task to teach the principles and theories of communication and ask students to apply this knowledge in their own teamwork, while covering the regular technical contents of a Software Engineering course and completing significant, real-world software projects, all in one term. This is particularly true given the quarter system on which the university operates; developing communication skills, in particular, take time as students gain increasing experience through participating in different communication settings. Over the last five offerings of this course, we tried several different approaches, as listed below, to approach this problem while using the same communication textbook, moving from approaches that were more instructor-controlled to approaches that were more student-centered.

- Lectures on communication topics by the instructor.
- Mini-lectures on communication topics by the instructor, along with various in-class activities to reinforce the concepts covered in the mini-lectures.
- Student team presentations on communication topics.
- Student-designed scenarios that illustrate the key concepts of different communication topics.
- Unrestricted in-class student activities that cover assigned topics.

Lectures on communication topics by the instructor.

We used instructor lectures to teach communication topics in one quarter. We divided the selected topics from *Communicate!* into twelve mini-topics that could be covered in fifteen to twenty minutes. The instructor then delivered the contents one piece at a time between Week 3 and Week 9. To reinforce student learning, these contents were covered in homework assignments, in-class quizzes, and exam questions.

This approach worked well in covering all important topics thoroughly because the instructor had full control of the lecture contents. However, communication topics were non-technical in nature. Even though good communication skills were hard to master, communication theories on paper were relatively easy to follow. When delivered in the same session as other highly technical contents, these short communication lectures were often considered mental breaks by students and were not effective in getting the messages across. In addition, as the work from cognitive psychology demonstrates, learning new material requires practice over time, with fewer cues provided by the instructor, in order for student mastery to take place[11]. Students' theoretical understanding of the concepts was an important first step but did not lead to actual skills in applying the theory.

Mini-lectures by the instructor.

To address the drawbacks of instructor lectures on non-technical topics, we tried to turn fifteen-to-twenty-minute lectures into three-to-five-minute mini-lectures that were intended to set the stage for more engaging in-class activities such as group discussions, questions-and-answers, and in-class debates.

This approach helped keep students engaged. These activities required preparation and when students were unprepared, these in-class activities were less efficient and often could not fully cover the planned topics. In addition, even with instructor guidance, unprepared students sometimes would drive discussion in wrong directions, which prevented student learning in the intended area from taking place.

Student team presentations on communication topics.

Later on, we experimented with student presentations in another quarter. The selected communication topics were still divided into mini-topics. Each mini-topic was assigned to a student team. Student teams were then required to prepare and deliver a fifteen-to-twenty-minute team presentation in class to present their assignment topics. Research in cooperative learning suggests that more learning takes place when students teach each other the material[12]. In their preparation and planning, students gain deeper insight into the subject matter. Two potential problems include the variability of the learning gains as a function of the amount of time and effort spent preparing, and the tendency for students to devote the most time and attention to their own presentations and be less familiar with the subject matter of other groups. These problems can be lessened with careful planning and assigning of tasks to student teams, thus ensuring equitable work outcomes through clearly specified team expectations.

In this approach, student teams performing presentations learned well on their assigned topics. The quality of their presentations, however, varied from team to team. Most of them tended to simply go over the content in the textbook one by one. Some students presented by reading each bullet on their slides and following each with a paraphrased sentence. Overall, the presentations were even less attractive to the rest of students than instructor lectures. Further, this tended to reinforce negative communication patterns, such as reading from slides, a public speaking style that does not work well in any setting. This reinforces the need for careful design of team projects, with clear specification of expected outcomes. As the need for greater specificity became apparent, the assignment of tasks became more student-centered, as described in the next section.

Student-designed scenarios.

To improve ineffective slide-by-slide, bullet-by-bullet student presentations, we encouraged students to include dialogues, scenarios, and mini-dramas in their presentations as the main tool to get the messages across. Students became creative in designing their presentations. Some teams came up with scenarios that were quite dramatic.

For example, when it was one team's turn to present, all team members approached the front of the classroom. One member carried a floppy disk. He inserted the disk into the computer in the classroom, which was connected to the projector. He then attempted to load from the disk a computer file that contained the presentation slides[i]. The disk did not appear to be readable. He then asked a teammate, who seems to have prepared the disk. That team member said he did not know what the reason for the failure. Then the rest of the team members started to blame each other right in front of the class. Worrying about maintaining order in the classroom, the instructor suggested the team use a USB keychain device to copy the file, or a network connection to download it. But then the situation worsened and the team member who took the most blame quickly walked out of the classroom without saying a word. Another team member ran after him, trying to get him back. Just when the instructor thought things had gotten out of control and was ready to reschedule the presentation, a slide showed up on the projector, displaying the word "Conflict" in large font, followed by a tag line: "When the needs or ideas of one person are at odds or in opposition to the needs or ideas of another." In the meantime, one team member started the presentation.

This team had been assigned to present on the topic of "conflicts." Apparently, they had preloaded the slides onto the computer. They decided to demonstrate different behaviors in a conflict before the presentation started. The faulty floppy disk was just a stage prop. Their performance caught both the instructor and most of the class off guard, and left a long-lasting impression on the topic of conflicts.

Unrestricted in-class student activities.

To encourage students to be even more creative, we later allowed unrestricted in-class activities designed and managed by student teams. Since this was an open-ended requirement, we added creativity as one of the grading criteria to award innovative ideas. Students were highly motivated and came up with many surprising and effective ways to cover their topics. Providing this flexibility allows students with different learning style preferences, such as those represented by the VARK learning inventory (Visual-Auditory-Read/Write-Kinesthetic), to develop learning materials that reflect their optimal learning situations [13]. The different learning orientations that are reflected in the presentations of the individual student teams provide a rich and diverse set of learning experiences for students in the course.

For example, one team was assigned to cover the topic of visual aids in public speaking. They developed a six-minute video, in which team members showed up in turn, each explaining one type of visual aid. What was highly creative and effective was that the video was only half of the show. When they played the video in class, team members went next to the screen in turn, shaking hands with themselves in the video, and passing objects to and from the video (the objects used in the video were actually hidden behind the screen). The show was both funny and informative. The presentation was highly

[i] Up until this moment, everything was normal. It was common that students load presentation slides to the computer in the classroom when it was their turn to present.

successful and the topic of visual aids was covered much more vividly and effectively than instructor lectures.

Another team was assigned to cover the topic of formal and informal leadership in teams. They came up with a short but meaningful game that illustrated the mechanisms of team leadership.

The Leadership Game

Before the presentation on the topic of leadership, the presenting team coordinated a game in the class. All students in the class other than the presenting students were split up into four groups and gathered in four corners of the classroom. The presenting team then randomly appointed two formal leaders in group one and group two. Group three and group four did not have appointed formal leaders. Each of them had two minute to discuss and decide an informal leader for their group.

The presenting team then passed out candies to the four groups. In group one and three, the candies were given only to the two leaders. The two leaders were asked to decide how to distribute the candies among members and deal with any repercussions any perceived unfairness may cause. In group three and group four, candies were given to all but one member. The two leaders were asked to work with their groups to make everyone happy. All groups had two minutes to distribute or re-distribute candies. After that, all students in four groups completed a short survey that measured their perceptions of the leader and the candy distribution method in their groups.

The presenting team then started the presentation and used the following results to illustrate related topics from the textbook.

Group one (with a formal leader; the leader got all candies in the beginning): The leader in this group gave one piece to each group member, and suggested giving the rest away to charity. This example highlighted that being fair was a good leadership skill. The people in this group were happy.

Group two (with a formal leader; candies were given to all but one): In this group, the appointed formal leader did not do much to help distribute the candies. Most group members stated in the survey that they were unhappy with their leader's actions. This demonstrated that when leaders did not participate, the group became dissatisfied.

Group three (with an informal leader; the leader got all candies in the beginning): This group decided that one member should get all the candies. They had a lively debate about playing rock-paper-scissors to decide who got all the candies, but then one person decided to take it all and just grabbed all candies. This

demonstrated that in a group where no clear leadership had emerged, rules may be broken and members may show disruptive behaviors.

Group four (with an informal leader; candies were given to all but one): In this group, one person did not want any candy, so the leader was able to re-distribute the candy fairly among the group members, who were all happy with the leader's decision. This demonstrated the importance of listening, because the leader had to listen to the group's input to make the best decision on how to distribute the candy.

Results.

The results of these approaches were mixed. There were approaches that clearly worked better than other approaches. Unrestricted activities clearly worked better than instructor lectures and simple student presentations.

There were also approaches to which students with different backgrounds responded differently. For example, even when given complete freedom, teams with more graduate students were more likely to choose simple presentation of content from the textbook. For these teams, further instructions, such as suggesting that mini-dramas are encouraged could be helpful. Further, as shown in Table 1, our graduate students were predominantly international students. Therefore, the choice of straightforward presentation style could be a function of the different cultural backgrounds of the international students rather than the difference between graduate students and undergraduate students. We did not, however, have enough international undergraduate students or domestic graduate students to note any patterns regarding differences corresponding to cultural perspectives.

Overall, after taking this course, students became more self-confident in team environments. More students become aware of team dynamics and the responsibilities of a good team player. Most students learned the importance of dividing large software projects into smaller, manageable pieces to assign to individual developers, and how to manage a software team. The communication skills they learned about and shared with their fellow students improved as they continued to practice them in further presentations and other communication situations with their fellow students and team members. Thus, as hoped, the students acquired communication skills within the context of the project work they were doing, thus strengthening the likelihood that they would be able to apply what they had learned in further project settings.

In addition to the approaches described above, we also attempted to use issue tracking tools to facilitate student learning of communication topics and to improve communications between the students and the instructor[14].

4. Summary

In summary, we believe that it is critical to integrate communication skills into a computer science or software engineering curriculum. From previous research, we know

that innovative pedagogy can improve student learning of technical content[15]. While it remains challenging to do so in a technical course where students spend most of their time focused on technical content, we have demonstrated how with careful planning, we can also use innovative pedagogy to improve student learning of non-technical content and thus help our students become both technically-competent software engineers or computer scientists and good, productive members of projects. In particular, we found that mixed approaches accommodated both students with different backgrounds and students with different learning styles, and were therefore the most effective. Some differences that we observed may have been a function of the cultural background of the student team members (international versus domestic students), but our student population wasn't large enough to fully track patterns between the student groups. We believe that we have offered a model for incorporating communication skills in a computer science/software engineering course, toward the end of building highly functioning teams of students, able to communicate well with each other and with potential clients. Our major suggestions include (a) utilizing a communication textbook such as Communicate!, (b) choosing key communication content and skill outcomes for students to learn, (c) dividing up this critical content and assigning student teams to prepare innovative materials to share with their classmates, (d) providing background in the different kinds of effective communication of content that could be used successfully in presenting the content (for example, considering the different kinds of learning styles and developing materials aimed at specific learning types), and (e) encouraging students to practice the concepts and skills that are critical to successful communication.

Acknowledgements

We thank past and current CS456/556 students for their creativity in learning and teaching communication topics. In particular, we thank Christina Lee, Walt Novosel, Chris Morway, Marc Macenko, Matt Hirsch, and Mark Kulich, whose creations were described in this paper.

[1] Ivar Jacobson, Grady Booch, and James Rumbaugh, *The Unified Software Development Process,* Addison Wesley, 1999.
[2] Object Management Group, "OMG Unified Modeling Language Specification," Version 1.5, Technical Specification, March 2003, available at http://www.omg.org/docs/formal/03-03-01.pdf .
[3] Moshe Bar and Karl Fogel, *Open Source Development with CVS*, 3rd Edition, Paraglyph Press, 2003, available at http://cvsbook.red-bean.com/OSDevWithCVS_3E.pdf .
[4] SourceForge.net Tracker Tool, accessible at http://sourceforge.net .
[5] Edmund Tsang, *Projects That Matter – Concepts and Models for Service-Learning in Engineering,* AAHE, 2000.
[6] Chang Liu, "Enriching Software Engineering Courses with Service-Learning Projects and the Open-Source Approach," the *27th International Conference on Software Engineering (ICSE'05)*, St. Louis, Missouri, May 15 - 21, 2005.
[7] Nelsonville Public Library, http://www.athenscounty.lib.oh.us/ .
[8] R. F. Verderber and K.S. Verderber, *Communicate!*, 10th Edition, Wadsworth Publishing, 2002.

[9] Joint Task Force for Computing Curricula 2004, a cooperative project of the Association for Computing (ACM), the Association for Information Systems (AIS), and the Computer Society (IEEE-CS), "Computing Curricula 2004: Overview Report," draft, November 22, 2004, available at http://www.acm.org/education/Overview_Draft_11-22-04.pdf .
[10] Jerome Tapper and Walter Buchanan, "A Novel Approach to Integrating Communication and Technical Skills Creating a Seamless Transition into Today's State of the Art Engineering Technology Industrial Environment," *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*, Session 1348, 2004.
[11] Diane Halpern, *Changing College Classrooms: New Teaching and Learning Strategies for an Increasingly Complex World*, Jossey-Bass Higher and Adult Education Series, 1994.
[12] Barbara J. Millis and Philip G. Cottell, *Cooperative Learning for Higher Education Faculty*, American Council on Education/Oryx Press Series on Higher Education, 1997.
[13] Neil Fleming, VARK: A Guide to Learning Styles, http://vark-learn.com/english/index.asp, captured January 4, 2005.
[14] Chang Liu, "Using Issue Tracking Tools to Facilitate Student Learning of Communication Skills in Software Engineering Courses," the *18th Conference on Software Engineering Education and Training (CSEE&T)*, Ottawa, Canada, April 18-20, 2005.
[15] Lonnie R. Welch, Sherrie Gradin, and Karin Sandell, "Enhancing Engineering Education with Writing-to-learn and Cooperative Learning: Experiences from a Software Engineering Course," *Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition*, Session 2558, 2002.

CHANG LIU
Chang Liu is an Assistant Professor of Electrical Engineering and Computer Science at Ohio University. He received his B. S. in Computer Science from Fudan University in 1991, his M. S. in Computer Science from Fudan University in 1994, and his Ph. D. in Information and Computer Science from University of California at Irvine in 2002.

KARIN L. SANDELL
Karen Sandell is the founding director of the Center for Teaching Excellence at Ohio University. She received her B. A. in Rhetoric from University of Minnesota in 1971, her M. A. in Communication from Illinois State University in 1972, and her Ph. D. in Communication Studies from University of Iowa in 1977.

LONNIE R. WELCH
Lonnie Welch is a Professor of Electrical Engineering and Computer Science at Ohio University. He received his B. S., M. S., and Ph. D. in Computer and Information Science from Ohio State University in 1985, 1987, and 1990 respectively. He is the director of Center for Intelligent, Distributed and Dependable Systems at Ohio University.