FIRST STEPS TOWARD INTEGRATING COMMUNICATION INSTRUCTION THROUGHOUT COMPUTER SCIENCE AND SOFTWARE ENGINEERING CURRICULA

Janet E. Burge[1], Paul V. Anderson[2], Michael Carter[3],
Gerald C. Gannod[1], Mladen A. Vouk[4]
[1]Department of Computer Science and Software Engineering, [2]Howe Center for Writing Excellence, [3] Department of English, [4] Department of Computer Science
[1,2]Miami University, [3,4]North Carolina State University

Introduction

One of the more recognized challenges facing engineering education has been providing graduates with the communication abilities necessary to ensure their success in the workforce.[1,2] Employers typically place effective communication at the top of the qualities they seek in new engineers.[3, 4, 5, 6] To prepare their students to communicate effectively in their careers, engineering programs may require a technical writing course taught by another department and, in some cases, one or two communication-intensive courses in their programs. Nevertheless, new college graduates encounter significant difficulty adjusting to workplace communication practices,[7,8,9] and employers invest substantial sums in mentoring, providing in-house training, or subscribing to external programs to teach new employees the communication skills that are basic in their workplace.[10] While technical writing courses provided by non-engineering faculty are helpful, they are too general to prepare students adequately for the domain-specific communication tasks demanded by their careers.[11] Attention to communication in a few engineering courses is also beneficial but does not provide enough breadth or guided practice to move students from novice to highly competent communicators in engineering contexts. Studies of the communication abilities needed by new engineering graduates produce a longer array of topics than a single communication course can provide, even when supplemented by a few writing-intensive courses in the major.[12,13] Isolation of communication instruction in these ways reinforces the assumption by many students that writing, speaking, and other communication assignments are "busy work" rather than key aspects of their professional education.

Supported by a three-year grant from the National Science Foundation, we are developing and piloting model curricula that teach communication skills as an integral part of computer science (CS) and software engineering (SE) courses.[14] Among the compelling reasons for exploring this pedagogical approach is the way it positions communication instruction within the disciplinary context in which students will pursue their careers, an especially effective way of teaching domain-specific communication abilities.[15, 16] Recent research has demonstrated that well-designed writing assignments that are based on the intellectual content of courses not only develop students' writing abilities but also increase their mastery of course content. Thus, communication instruction in engineering courses can support technical instruction instead of detracting from it.[17, 18, 19, 20] Also, when communication assignments based on real-world practice are integral with their technical assignments, students can see how communication is the means by which they make their technical knowledge valuable to their employer, clients, and other stakeholders. Repeated attention to communication in engineering courses over the four years of their undergraduate study can enable students to see that acquisition of communication expertise is an essential element in their development as engineering professionals.

Given these benefits, a few universities have explored ways to integrate technical and communication instruction in one or more of their engineering programs.[21, 22, 23] The scope of our project is distinctive: Our goal is to develop resources that can be used, in theory, by all programs; our project team includes specialists from 14 colleges and universities; and we are focusing on four modes of communication: reading, writing, speaking, and teaming.

We are identifying learning outcomes and developing course materials that provide students with the skills in these areas that are needed to communicate effectively in engineering and

production environments. Our interdisciplinary team includes computer science, software engineering, and technical communication specialists from the lead institutions, Miami University and North Carolina State University, and from twelve other colleges and universities. We are also benefitting from advice provided by industry professionals. In the project we are producing model curricula developed and evaluated at Miami University and North Carolina State University, sample assignments developed at a variety of types of institutions, and teaching materials for instruction and evaluation of students' communication skills. We will create a resource site where these materials can be easily searched and then adapted in whole or in part by any CS/SE program. While we are working specifically with CS and SE education, the potential benefits and the overall design of our project could apply to any engineering field.

In this paper, we present results from the first year of the project, including communication skills, general and domain-specific, that our industry partners and faculty participants have categorized as essential for CS/SE graduates; challenges we have identified in implementing a communication-infused curriculum; and general strategies, including examples, we developed so far for integrating communication and technical work in CS/SE curricula.

Project Description

Our project, titled "Integrating Communication Learning Outcomes Across the CS and SE Curriculum" is a collaboration among CS/SE faculty, technical communication specialists who also have expertise in communication across the curriculum (CAC), and industry professionals. The five Principal Investigators (PIs) include at least one CS/SE faculty member and one communication-across-the curriculum specialist from our two collaborating institutions: Miami University and North Carolina State University. During its initial phase, October 2009-May 2010, the five PIs recruited six CS/SE faculty from each of our institutions plus six others, each from a different college or university. With the latter group, we selected faculty from a variety of institution types so that we could incorporate the perspectives of institutions much different from our own into the project. We also recruited six communication-across-the-curriculum specialists from six institutions to expand the range of perspectives still farther.

We selected the CS/SE participants so that the project team included an instructor teaching each of six courses from each of the PIs' programs and from one of the other institutions. These courses start with the introductory programming course, CS1, taken as first or second term Freshmen, and end with the Senior Capstone/Senior Design course that typically concludes most programs. The courses in between are the second programming course (CS2), Data Structures, Databases, and Software Engineering. These courses are common to both the CS and SE curricula and, depending on the institution, often span all four years of the curriculum. These courses were chosen as a common set that we could use to demonstrate how communication skills instruction and practice could be integrated at different points in the curriculum to allow students' expertise to grow as they progressed through the program. We then formed course-based teams that included CS/SE faculty from three institutions (Miami University, North Carolina State University, and a partner institution) as well as a CAC specialist to create course-specific communication assignments and instructional support materials for their courses.

We recruited industry partners representing various sizes and kinds of employers to assure that the guidance we received from CS/SE professionals represented the broad range of careers our graduates might pursue. In an earlier NSF-sponsored project seeking advice from industry to guide the communication abilities desired by the CS/SE industry, the PIs realized the importance of such breadth.

We launched the project with a three-day workshop in June 2010 at which university participants and industry representatives discussed the communication skills needed by CS/SE graduates; attended training sessions on developing program and course outcomes; and introduced a framework for assignment construction and assessment rubrics. The course-based teams then worked during the summer to develop an initial set of assignments to pilot during the next course year. In August 2010, we reviewed pilot assignments and participated in training sessions on teaching and evaluating communication skills in CS/SE courses. A workshop was held for each communication skill: reading, writing, speaking, and teaming. The CAC

participants in the project conducted these workshops with assistance from CS/SE participants.

Since then, we have continued to develop and assess teaching materials and pilot assignments, focusing on ways to integrate work done in individual courses into clearly articulated pathways in which students develop communication abilities progressively in the same way they build technical expertise as they advance through their four years of undergraduate study. We are developing ways of making the resources we create available to engineering educators nationwide.

Required Communication Skills

Because our project focuses on providing students with the communication abilities that are critical to success in their specific careers, we decided to start our project by asking our industry partners and our faculty to identify specific skills they felt were especially important for recent CS/SE graduates. Our first workshop was attended by executives and managers from 11 large and small corporations, including Microsoft Research, NetApp, Northrop Grumman Electronic Systems, EMC, SAS, Fidelity Investments, IBM Corp., and Integrated Industrial Information, Inc (I3). We broke into small groups and started the discussion with a series of questions:

- What types of writing and oral presentations do you expect CS/SE professionals to be able to do in their first year on the job?
- Which of these expected communication abilities do they generally not possess?
- What kinds of reading are important for first-year CS/SE employees to do well in their work (including requirements and code)? What do you expect these employees to be able to do as a result of those kinds of reading?
- What are important skills, attitudes, other attributes expected for effectiveness in working on teams?
- What communication abilities do you expect recent hires to learn on the job rather than bring to the job?
- What communication abilities must CS/SE employees have to advance in your organization?

The answers to these questions and the discussions around them in small groups and then a larger meeting covered a wide range of concerns. Some of the communication abilities identified were CS/SE specific, such as reading someone else's code, while others were more generic, such as asking questions. Many of the communication tasks involve more than one mode of communication, such as speaking in team meetings or writing notes or scripts for presentations. Table 1 lists some of the skills identified.

Much of the discussion focused on some of the differences between the ways communication is practiced and taught in a classroom setting and the ways it takes place in a professional or "real world" setting. One example concerns audience—in the workplace, the writer is not always aware of who the audience is or could be for their documents. Even when writing for an audience other than their instructor, students are usually told who the audience is that they should be writing for. Another involves understanding expectations. Students are usually told what is expected from them during a class but in the workplace this is something they often need to figure out for themselves, usually by talking with their new co-workers. Another challenge is understanding cultural differences. This is especially critical if parts of a project are multi-national projects overseas, a fairly common occurrence nowadays.

There also were some areas where our industry participants thought students were having a particularly difficult time. One is the need to be brave about sharing failure. Students may try to hide problems they are having from their instructor in fear of receiving a lower grade, but following that approach on the job can cause some serious issues. Another is in looking at problems from multiple angles. Students tend to be more black and white in thinking and do not always consider different perspectives. Another skill that often does not get taught is the ability to see the "big picture"—what are the goals behind the project and how does the project fit into the goals of different audiences? Other skills include deciding which communication medium is most appropriate (e-mail vs. phone vs. in person), negotiating a position, and asking questions. The skills identified by our industry participants are now being used to motivate the development of assignments that address these skills.

Table 1: Communication Skills Identified during June 2010 Workshop.

Reading		**Teaming**
Read specifications		Participate in team meetings– including
Find bugs in specifications		making a technical argument in them
Read someone else's code		Participate in a scrum
Read training manuals		Structure a plan for a small group
Read with a purpose (different purposes		Collaborate with team members (rather than
appropriate for different situations)		competing as can happen in academe)
Foraging for information		Make the team's goals your goals
Read e-mails		Communicate across cultures— work
		with team members from cultures that do
Writing		not fight for their opinions
Write technical summaries		Adopt to a new team
Prepare bug reports		Determine when it is important to update
Create design specifications		your manager on your project's status
Create implementation specifications		Manage conflict
Write/abstract a white paper		Assess yourself
Write e-mail		Ask questions
Write status reports		Prepare meeting agendas/minutes
Whiteboard		
Prepare meeting agendas/minutes		
Speaking		
Make a technical argument		
Walk through code or design		
Make PowerPoint presentations		
Make an elevator speech		
Determine when to make a phone call		
(rather than send e-mail, memo, etc.)		
Ask questions		
Communicate status of projects and tasks		
Whiteboard		

Challenges Encountered

During the process of developing the outcomes and course materials for this project, we have encountered a number of challenges that pose potential risks for institutions that intend to adopt our work. In this section we identify those risks, while in the next section we provide a number of strategies that serve to mitigate the risks. The challenges we have encountered in implementing a more communication-intensive curriculum can be grouped into four non-exclusive categories:

a) Curricular issues,
b) Instructional issues,
c) Logistical issues, and
d) Motivational issues.

Curricular issues are primarily concerned with identifying how to best incorporate communication skills into a larger degree program. The biggest issue is the add/subtract problem—is it possible to include communication without having it be at the expense of technical instruction? Although research demonstrates that incorporating well-designed writing assignments and instruction writing into a course increases mastery of technical material,[17, 18, 19, 20] the fact remains that many courses, particularly in the initial programming sequence, are already quite crowded. This means that it is critical that the communication-based assignments be integrated into the instruction in a way that does not remove the focus from technical skills. Removing technical writing and English courses from a curriculum to provide room for domain-specific communication courses or materials in a technical curriculum is a possibility, but not a wise action because communication instruction by writing specialists and by technical faculty complement each other in ways that round out students'

communication expertise. Another challenge is determining which communication skills should be addressed in which courses. For example, many faculty are concerned that working in teams during lower level courses may mean that weaker students may be able to get through without demonstrating that they have mastered the knowledge and skills they need to succeed in subsequent courses. The level, amount, and emphasis on communication skills need to be appropriate for the goals of each course.

Instructional issues are primarily concerned with the degree to which technical faculty are trained in teaching and assessment of communication skills. Specifically, instructional issues arise because technical faculty may not be formally trained in how to teach or assess communication. Over the past several decades, the writing-across-the-curriculum and communication-across-the-curriculum-movements have developed many strategies that faculty who have not had training in communication pedagogy can use to incorporate writing and speaking into any college course.[15, 24, 25] These practices have been adopted by some engineering faculty and programs.[26, 27, 28, 29] Often the result is that a few faculty in an engineering program include and even emphasize writing, but the majority do not. In contrast, by emphasizing the infusion of writing in six courses spread over all four years of students' undergraduate studies, our project would engage a much larger portion of a program's faculty, perhaps even all of them, in leading courses where attention to communication is a significant feature. Further, in contrast to the usual approach of WAC and CAC initiatives, which usually involve assigning writing activities and projects, our project includes *instruction* by the professor. For many engineering faculty, it may not be clear how much instruction needs to be given, when it is best provided, and what they would say if they were to teach communication skills to their students. In addition, faculty may not have a clear picture of which communication skills their students possess upon entry into a course and which they will need to teach. For example, many CS/SE students are required to take a technical communication course in order to complete their degree, but this course is usually not a prerequisite for any of their technical courses, so they often approach writing a technical report in the same way that they would approach writing an essay. Instructors need to know how to communicate their expectations to the students. Assessment (including grading) of communication is also new to many instructors. They need assistance in learning what to comment on, how to comment productively, and how to assign grades to this kind of student work. Also, they fear that grading will take a great deal of time, so they need advice about how to do it efficiently.

Logistical issues are concerned with the degree to which communication skills can be operationally achieved given constraints within a department or institution. In particular, there is not a one-size-fits-all approach to incorporating communication skills into technical courses. Different types of institutions and programs will have different challenges related to size, resources, and infrastructure. An institution that teaches small courses where the instructors are responsible for grading will be able to offer different types of assignments than one that teaches large courses where grading is handled by teaching assistants (who often will not be native English speakers or have not received appropriate instructional training, adding another level of challenge). The same course may be taught at different levels at different institutions or a single course could have students taking it at different levels. For example, the Database course at Miami University could have sophomores, juniors, and seniors all in the same section.

Motivational issues are concerned with creating instructional buy-in from both student and faculty perspectives. From the perspective of the technical faculty member, the technical content is seen as paramount while the technical communication skills are seen as important but secondary. Many students choose CS/SE for their major because they enjoy the technical aspects of the work. There is a common perception that "soft skills" such as writing, speaking, and teaming are not needed and less critical and are acquired automatically (in a mysterious and unspecified way). If the students and faculty do not perceive something as being valuable they are less likely to invest time and effort into doing it well.

Strategies

In the first year, our project focused on developing and piloting assignments in at least six different courses and at eight different institutions. In this section we describe some of the strategies that we have developed to address the challenges identified above. Specifically, we have identified four key strategies:

a) Identify communication learning outcomes at both the program and the course level;
b) Design rubrics for communication-based assignments to both assist in communicating expectations to students as well as support instructor grading;
c) Provide instructional supports to instructors to assist in teaching domain-specific communication skills; and
d) Develop a framework for communication-based assignment development that emphasizes outcomes, rubrics, and a real-world context for each assignment (to provide motivation for the student and instructor).

Communication Skills Outcomes

Many faculty in engineering and computer science are familiar with program and course outcomes through ABET. For faculty in programs that have not sought ABET accreditation, the concept of student learning outcomes (SLOs) may need to be introduced and explained. In any program, whether or not affiliated with ABET, communication outcomes are only one of many kinds of student learning outcomes, leaving them only a small place in the program's graduation-level outcomes. They may only appear in the list of outcomes for one or two courses. We recommend giving prominence to communication outcomes by making them more explicit among both program and course outcomes. This strategy serves to address both the curricular and motivational issues identified earlier. When included in the graduation-level outcomes, they answer such questions as "Why should we do this?" by indicating the prominence of communication skills among the qualifications graduates will bring to their careers. Included among the outcomes for specific courses, they answer questions like "Where should we do this?"

In the example program-level outcomes shown in Table 2, communication outcomes are integrated with technical outcomes. The message is that all faculty are responsible for enabling students to achieve them all in whatever courses they may teach. Faculty can use these outcomes to identify those they will incorporate in their courses so that students have a broad experience of using the major forms of communication in the field in a variety of courses.

Although our project focuses on six courses, every course in a program could have its own set of student learning outcomes (SLOs) that contribute towards the eventual achievement of the program-level outcomes. These outcomes describe what students should be able to do upon completing the course. They can be created in two ways: by writing new (additional) outcomes that describe the communication skills incorporated into the course or by modifying existing outcomes to incorporate or highlight communication. The advantage of the latter is that it makes it clear that communication is integrated into the instruction and evaluation of technical skills rather than something separate. In the CS2 course at Miami University, both strategies were employed. In considering the communication within a course, one needs to remember that communication in that context means being able to speak, read, write and in general communicate in the language of the field. In the case of software engineering, the communication would use SE ontologies, dictionaries, specific SE languages, and so on. In the following examples, the words in boldface highlight the text that was added to three technical outcomes in order to incorporate communication outcomes:

- Write basic UML[1] class diagrams **based on a problem statement**
- Break a programming problem down into an appropriate set of classes, identify appropriate methods for each class, **and explain the design choices made.**
- Design **and document** a complete set of test cases and use this to identify logic errors.

The first outcome incorporates "reading" and "writing" as a communication skill: Students need to be able to demonstrate that they can read and understand the problem statement as an input to their design. They also need to be able to write that down in a domain-specific way. The second outcome indicates that students need to "read" a programming problem specification and explain their solution (design) and its semantics to someone. In this case, it is left open whether they will explain orally (speaking) or in writing. The third outcome requires that they document their test cases, which requires writing in an SE-specific sub-genre called test cases.

[1] UML (Unified Modeling Language) is a collection of notations used by software engineers to specify requirements, design, and development artifacts.

Table 2. Program-Level Learning Outcomes Developed by CS/SE Faculty at North Carolina State University.

Program-Level Learning Outcomes

To demonstrate that graduates can reason effectively about computing and develop software, they should be able to:

1. Identify and define abstract computing models that could provide a basis for solving a given problem and analyze them for their potential and limitations for a solution.

2. Prove mathematically the characteristics and limitations of an abstract model of computation with respect to the ability to solve specific abstract problems and/or to do so efficiently; inherent in this ability is the mastery of techniques such as (a) decomposing and synthesizing instances, (b) providing the equivalence of different models, (c) searching for patterns in the various instances, (d) proving that certain patterns fit the model and others do not fit the model, and (e) determining the extent to which a model can solve the problem and solve it with acceptable use of resources as defined mathematically.

3. Develop efficient algorithms and data structures for solving a problem and identify other problems or algorithms to which these apply.

4. Recognize and define a problem related to a specific scenario that can be solved with a software application. Describe how the end-users or internal actors within a system intend to use the application to be developed. Gather and analyze information that allows for requirements that will solve the problem to be created; validated; verified; and, if necessary, revised.

5. Create and express a design for an underlying abstract model of computation that accommodates defined system requirements—including considerations of privacy, security, and efficiency—so that a developer can implement the application. Review the design to ensure it can accomplish the requirements and, where it does not, redesign until it meets the requirements.

6. Implement software conforming to a specified design so that it is usable, testable and modifiable by others. Review the implementation to ensure it meets the system requirements and conforms to design and, where it does not, correct the implementation until it meets the requirements and design.

7. Plan and execute appropriate tests in order to identify ways in which the software does not meet the requirements and, where it does not, to redesign, implement and retest until it meets the requirements.

The following **communication outcomes** are derived from the general program outcomes above. By achieving these communication outcomes, students both learn to do what is described in the general outcomes and demonstrate that they have attained those outcomes.

To demonstrate that graduates have achieved the general program learning outcomes, they should be able to:

1. Present in writing or orally an abstract model that could be used to solve a real-world application problem so that the presentation could be understood by stakeholders.

2. Write a mathematical proof related to an abstract model of computation so that it can be understood by an audience with sufficient mathematical maturity (ability to understand proofs by induction, contradiction, etc.)

3. Present in writing or orally the reasoning they have applied in creating a mathematical proof related to an abstract model of computation so that it can be understood by someone acquainted with an application of the model.

4. Present in writing or orally a description of how an abstract model of computation can be productively applied to solving a problem related to software engineering in another area of computer science or in another field

5. Present in writing or orally a critical assessment of a problem situation defined by a need for software to be developed for solving the problem: (a) collect information from sponsors, end-users, and on-site observations; (b) analyze that information; (c) use the analysis to define the problem in terms of the stakeholders' needs and goals for addressing those needs

6. Write requirements representing the stakeholders' needs and goals in such a way that the requirements can be applied in a design by others

7. Read requirements for various purposes, such as to inspect and correct them, to validate them as meeting the user's needs, to revise them so that they better meet user's needs, to implement them in a design, and to identify what students don't know and what they need to know to create code.

8. Write a design that accommodates the defined system requirements—including considerations of privacy, security, and efficiency—so that a developer can implement the application.

9. Read a design for various purposes, such as to ensure it can accomplish the requirements and, where it does not, redesign until it meets the requirements and to translate it into code.

10. Write a program to conform to a specified design so that it is usable, testable, and modifiable by others.

11. Write a narrative description of code, including a list of file names or directories included.

12. Read code and comments for various purposes, to find and correct errors in syntax and semantics, to determine what a program is supposed to do, to revise a program so that it accomplishes what it is supposed to do, to modify a program for different purposes, to ensure that a program conforms to system requirements and conforms to design, to provide productive feedback to those who created it, to continue a program begun by someone else, and to apply it to new uses.

13. Write a developer guide that is appropriate to the audience.

14. Write a user guide that is appropriate to the audience.

15. Present in writing or orally a test plan and results of testing that identifies ways in which the software does not meet the requirements.

16. Present in writing or orally progress reports that describe advancements and difficulties in a software development project.

17. Present in writing and orally a full technical report describing a software development project.

18. Read technical literature in the field for various purposes, such as to summarize, to analyze it, to answer a technical question, and to solve a technical problem.

19. Present in writing or orally a research report that solves a technical problem based on an analysis of literature in the field.

20. Work effectively in teams: (a) develop ground rules to guide the team's approach to work; (b) define roles so that expectations of team members are clear and followed; (c) create agendas and minutes for team meetings; (d) interact with other team members in ways that assure the productive contributions of all team members; (e) create specific action items for each member and then hold him or her accountable; (f) identify, create, and manage the tools that enable teams to work effectively; (g) resolve conflicts among team members.

For the CS2 course, faculty also developed new communication-centric outcomes:

- Interpret a UML diagram and explain its relationship to a problem statement.
- Read and understand code written by people other than themselves.
- Use a problem statement to define a set of software requirements.
- Explain how a final software implementation deviated from their original design.
- Follow good programming style and documentation conventions to write code that is easily understandable and extensible.
- Explain issues encountered and progress made during a software development project.

In these examples, not all categories of communication skills were explicitly required. For instance, none of the outcomes address teaming or explicitly involve speaking. One of the advantages of distributing communication skills across the curriculum is that it is not required that every class address every skill. For example, teaming might not be desired in lower level programming classes where students must program on their own to master critical skills, and classes taught in large sections will not be able to manage the logistics of students presenting in class. In the former, it is still possible to have students team in a lab setting and in the latter students could still practice speaking in small groups.

Each institution distributes skills across their curriculum in different ways so it would not be practical to produce definitive lists of SLOs for each course involved in this project, but we will produce examples from Miami University and from North Carolina State University as well as instructions on how existing outcomes can be tailored to add communication skills.

Rubrics to communicate expectations and guide assessment

Rubrics serve a number of different roles.[30, 31] For the faculty member, they provide guidance about the communication principles to discuss with students, and they offer the criteria by which student work will be evaluated. The details of a rubric provide a grader (either the faculty or teaching assistant) with specific characteristics by which to differentiate between excellent and novice student work. For the student, a rubric acts as a statement of expectations for a work product. In addition, a rubric can be used as a specification of the relation between outcomes and student achievement of those outcomes. For instance, in the rubric provided in Table 3, different traits can be directly related to this outcome: "The student can give an effective oral presentation of requirements."

Table 3. Rubrics for a Requirements Presentation Assignment.

Grading:

1. Presentation dry run (2 pts): 2 pts if a complete dry run is given to the instructors prior to the dinner, 1 pt if a dry run is given where the presentation was thrown together hastily, 0 if no dry run is performed.
2. Presentation introduction (3 pts): 2 pts if a slide or two is given introducing the project and why it is valuable to the clients. This serves as the motivation for the rest of the talk. 1 pt if the introduction is not clear, 0 otherwise.
3. Requirements description (7 pts): 7 pts if requirements (functional and nonfunctional) are clearly described in nontechnical language and are organized logically, 4 pts if requirements are not clear or lacking in detail, 2 pts if requirements are incomplete, 0 otherwise.
4. Task descriptions (7 pts): 7 pts if all the major tasks (or task categories) are described clearly in nontechnical language, 4 pts if some parts of the system appear to be missing, 2 if descriptions are vague, 0 otherwise.
5. Storyboards (7 pts): 7 pts if storyboards are legible and provide enough detail for the client to visualize how someone would interact with the system, 4 pts if some storyboards are confusing or if one or two are missing, 2 pts if storyboards are incomplete,0 otherwise.
6. Presentation flow (2 pts): 2 pts if the flow of the presentation is easy to understand with clear transitions, 1 pt if there is a spot where a listener can get lost, 0 if it is difficult to follow the presentation.
7. Team Presenting (2 pts): 2 pts if team members introduce each other and all team members speak, 1 pt if not all team members speak or if some team members appear unengaged while their teammates are speaking, 0 if the presentation was not developed as a team.
8. Professionalism (2 pts): 2 pts if the team presents themselves professionally, 0 otherwise.
9. Audience aware (4 pts): 4 pts if any technical terms are explained clearly for a nontechnical audience, 2 pts if one or two spots are not clear, 0 pts if the talk is not accessible to non CS people.
10. Visuals (4 pts): 4 pts if all graphics and text are clearly readable, 2 pts if there are any "eye test" slides, 0 if the presentation is difficult to read.
11. Speaking (4 pts): 4 pts if all speakers speak clearly and enthusiastically, make eye contact with the audience, and appear to have rehearsed the talk, 3 pts if one person appears disengaged, etc. Note that nervousness will not be penalized nor will the use of notes as long as the speaker still attempts eye contact.
12. Questions (4 pts): 4 pts if the team actively solicits and accurately responds to questions and feedback, 2 pts if questions are dodged or dismissed out of hand, 0 pts if no attempt is made to actively solicit questions.
13. Peer evaluation summary (5 pts): 5 pts for a summary that lists responses to all the major points made by the peer evaluations plus an overall summary of how the presentation could be improved, 4 pts if some points are missing, 3 pts if the overall summary is missing or if some peer evaluation comments are not given a thoughtful response, 0 otherwise.

In addition, significant point reductions may occur if any of the following are detected:

1. Use of any graphics, pictures, text without appropriate citations (the source MUST be

given for any graphics used, etc.).
2. Lack of sensitivity towards the clients using the project.
3. Inappropriate responses to audience questions.

Instructor Supports

The August 2010 workshop contained four sessions on teaching each of the four communication skills—reading, writing, speaking, and teaming. These were designed to help project participants get started in teaching their students these skills. This was a good start towards training one set of instructors, but the goal of this project is to provide assistance so that other instructors can incorporate communication into their courses as institutions adopt more communication skills into their curricula. To facilitate this, we will be developing a variety of instructional supports.

Instructor supports are required to assist with three issues encountered in teaching and using communication skills. One issue is that it is not always clear how or when students need to be trained in writing, speaking, teamwork, or reading. Faculty members may not be comfortable teaching these topics (which they may not have been taught themselves). Another issue is that while some instruction is necessary if the students are to be successful, it needs to be done in a way that minimizes the impact on the time given to technical topics and avoids repeating the same (nontechnical) instruction in multiple courses. The third issue is assessment, which we hope to address through the use of rubrics. While some rubrics are assignment specific, there are some generic ones that can be defined for common types of assignments that can then be tailored as needed.

The instructor supports are being designed and developed based on the experience of piloting the first set of communication-based assignments. Some supports have already been requested, suggested or employed:

- Instructional materials, such as PowerPoint slides, to teach each communication skill.
- Rubrics for assessing presentations.
- Rubrics for assessing peer review.
- Document templates (those already defined include status reports, meeting agendas/minutes, requirements specifications).
- Podcasts of training materials so instruction will not involve class time.
- A quick reference guide on communication skills that can be provided to students.
- Examples of good student work to accompany assignments.

As additional assignments are piloted, instructors are reporting back on where they require additional assistance. The CAC experts on the project are working with the instructors to design, evaluate, and refine supports needed.

Framework for Assignment Development

The project had eighteen faculty from eight different institutions developing assignments. A framework was defined to guide assignment development by requesting that faculty define the following information along with each assignment:

- Which communication abilities the assignment would develop (writing, speaking, reading, teaming, and listening).
- Course learning outcomes addressed: both technical and communication (separate sections were given, however faculty were encouraged to combine these when possible).
- An explanation that could be given to the students on how the assignment benefits them. This explanation should relate the assignment to their future professional practice, a key factor in providing them with motivation for doing the assignment and taking it seriously. When possible, assignments are mapped to the specific communication skills that our industry partners identified (as listed in Table 1).
- Technical tasks that the assignment would be used with.
- The genre of the assignment. Genre, in this context, refers to the type of communication. For example, a Software Requirements Specification would be a genre.
- The audience for the assignment. Audience is critical in communication. A document or presentation designed for a technical audience would use terminology that would be inappropriate for a nontechnical audience.
- The purpose of the assignment. For example, a requirements specification is written to define what the finished system is required to do, while a status report is written to keep a

manager or customer apprised of the progress being made on a development project.
- Specifications of the assignment–the length, formality, and level of polish required from the students.
- Evaluation criteria for assessing student work. This section typically would refer to a rubric written for the assignment.
- The process followed in administering the assignment. Some assignments consist of multiple steps and deliverables.
- Project milestones required for longer projects that may include revisions and dry runs.
- Resources needed by the instructor to teach the students the skills needed to administer the assignment. These may also include resources that directly support the students such as document templates or examples of successful prior student work.

This framework is still a work in progress and will be modified based on feedback from assignments as they are piloted. The framework for a specific assignment also will be adapted each time an assignment is piloted to adjust to problems as they were encountered. Table 4 gives the framework for a Requirements Presentation assignment developed for Miami University's capstone course.

Table 4: Framework Example for a Requirements Presentation Assignment.

Title of Assignment	Requirements Presentation
Course	CSE448 – Senior Design Project I
Communication abilities developed by the assignment	_x__Writing _x__Speaking _x__Reading __x_Listening _x__Teaming
Typical course learning outcomes addressed by this assignment	1.1: The student can define the problem, determine requirements to solve the problem, and analyze alternative approaches to solving the problem. 2.2: The student can document and present the results of the design process by the following means: Prepare and deliver various written engineering reports as requested by the client; prepare and deliver effective professional oral presentations. 3.1: The student can successfully function in a team environment.
Learning outcomes for assignment	
Technical	The students will demonstrate their ability to understand the problem, define requirements (including use cases), and develop alternative approaches.
Communication	The students will demonstrate: • The ability to design and create an effective presentation aimed at a nontechnical audience. • The ability to deliver the presentation as a member of a team. • The ability to attentively listen to, and clearly answer, questions from the audience.
Explanation to students of the assignment's benefit to them	The requirements specification is used to document the results of the collaboration with the client to determine what the completed system must be able to do. In addition to the written document, which not all stakeholders will have the time to read, a presentation can serve as a mechanism to get feedback on the requirements prior to beginning the design process. The client is often not a computer programmer so it is critical that the language used in the presentation is that of their domain, not that of the implementation domain. This presentation can also serve as a way to "sell" the client on your ideas.
Technical task(s) with which it might be used	System Requirements

Deliverable	
Genre	Presentation for a nontechnical audience.
Audience	Therapists, High School Students, outside visitors.
Purpose	To communicate the software requirements to the client and obtain feedback on the initial plan for what the system will do.
Specifications (length, level of polish, or formality)	Thirty- minute formal presentation. Approximately 20 slides.
Evaluation criteria (attach rubric)	See Rubric (in Table 2 of this paper).
Additional, when appropriate	
Process	1. Dry run the presentation for the instructors. 2. Present the slides at a formal dinner for the instructors, clients, and HS students working on the project. 3. Conduct peer evaluations as well as grading by the instructor. 4. If questions do not come automatically, call on audience. 5. Read and summarize the peer evaluations.
Milestones	
Other	
Resources to support instructors using this assignment	[Some of these resources will be developed as the project proceeds.]
Writing related	Slides giving guidelines for what makes a good presentation; specific instructions for this presentation; peer review form and instructions on each criterion from the form.
Speaking related	See above but focus on speaking.
Reading related	
Teaming related	See above (project should be a team project).
Listening related	Instruction to students on how to answer questions.

Summary and Conclusions

The three-year project described here is tackling the ambitious problem of developing a new methodology to better prepare our students for the kinds of communication they will need to be proficient at in order to succeed in the workplace. Our approach is to target six core courses that span the CS and SE curricula as opportunities to integrate reading, writing, speaking, and teaming into their technical instruction. This allows the skills to be taught in context and also serves to reinforce the idea that communication is a necessary component in professional success.

Prior to and during the implementation of this project, currently at the half-way point, we have identified numerous challenges to its adoption that can be categorized as curricular (how to best incorporate skills into a larger program and into individual courses), instructional (how to teach and assess communication), logistical (how to incorporate communication into courses at different levels of the curriculum and at institutions with different class sizes), and motivational (how to convince students and faculty of the importance of communication). The project addresses these issues in several ways. For curricular issues, we are developing program and course-level student learning outcomes as a guide for skill distribution and integration and will provide a curriculum spanning set for institutions of two different sizes. For instructional issues, we will provide instructional supports to faculty to assist with instruction and sample rubrics to assist with assessment. For logistical issues, we are working with eight different institutions and will provide sample assignments that have been piloted at these institutions. For motivation, we have teamed with industry professionals to provide their assessment of what skills they need to see in new graduates and we will be using this insight to design assignments that

can target these skills. We expect students and faculty to be more receptive to assignments that are grounded in actual professional practice.

The results of the project will be disseminated on-line and will include all outcomes, assignments, and instructional materials generated. By providing two model curricula piloted at two very different institutions, Miami University and North Carolina State University as well as individual assignments developed by our partners, we hope to support adoption of this approach at other institutions so that CS and SE students will graduate with the communication skills necessary to succeed in their professional careers. The strategies employed and lessons learned will also be valuable to programs in other engineering fields interested in increasing the communication abilities of their graduates.

Acknowledgements

This work was funded in part by NSF CPATH-II Awards CCF-0939122 and CCF-0939081, IBM Corp., and NC State Engineering Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF). We would also like to acknowledge the invaluable contributions of our project participants.

References

1. ABET Engineering Accreditation Commission (www.abet.org). *Criteria for Accrediting Engineering Programs—Effective for Evaluations During the 2011-2012 Accreditation Cycle*.

2. ABET Computing Accreditation Commission (www.abet.org). *Criteria for Accrediting Computing Programs—Effective for Evaluations During the 2011-2012 Accreditation Cycle*.

3. United States Department of Labor. (2010-2011). "Engineers." *Occupational Outlook Handbook.* http://www.bls.gov/oco/ocos027 .htm.

4. Pfeiffer, P. "What employers want from students." Association for Computing Machinery,http://www.acm.org/membership /student/emplymntart.html.

5. Leibowitz, J. 2004). "Teaching the importance of communication in IT." *IT Professional, 6*(1), 38-42.

6. Karunasekera S., Bedse K. (2007). "Preparing software engineering graduates for an industry career." *Proceedings of the Conference on Software Engineering Education and Training*.

7. Begel A., Simon, B. (2008). "Struggles of new college graduates in their first software development job." *Proceedings Of The 39th ACM Technical Symposium On Computer Science Education*.

8. Dias, P., Pare, A. (2000). *Transitions: Writing in academic and workplace settings*. Cresskill, NJ: Hampton.

9. Dias, P., Ed. (1999). *World's apart: Acting and writing in academic and workplace settings.* Mahwah, NJ: Earlbaum.

10. National Commission on Writing. (2004). *Writing: A ticket to work . . . or out of it*. Princeton, NJ.

11. Wolfe, J. (2009) "Why technical communication textbooks fail engineering students." *Technical Communication Quarterly*. 18.4, 351-75.

12. Katz, S.M. "Part I-learning to write in organizations: what newcomers learn about writing on the job." *IEEE Transactions on Professional Communication. 53*(4), 107-115.

13. Ruff, S., Carter, M. (2009). "Communication learning outcomes from software engineering professionals: A basis for teaching communication in the engineering curriculum." *IEEE Frontiers in Education Conference*.

14. Can't include this reference in the submission because of the double-blind review process.

15. Carter, M., Ferzli, M., Wiebe, E. N. (2007). "Writing to learn by learning to write in the disciplines." *Journal of Business and Technical Communication 21*(3), 278-302.

16. Russell, D. R. (2002). *Writing in the academic disciplines: A curricular history.* 2nd Ed. Carbondale: Southern Illinois University Press.

17. Paine, C., Gonyea, B., Anson, C., Anderson, P. (2009)."The so-called 'best practices' for writing: Do they make a difference for engagement and learning?" Paper presented at Annual Meeting of the American Association of Colleges and Universities.

18. Bangert-Drowns, R. L., Hurley, M. M. Wilkinson, B. (2004)."The effects of school-based writing-to-learn interventions on academic achievement: A Meta-Analysis." *Review of Educational Research 74*(1), 29-58.

19. Carter, M. (2007). "Ways of knowing, doing, and writing in the disciplines." *CCC 58(3),* 385-418.

20. Ferzli, M., Carter, M., Wiebe, E,. (2005). "LabWrite: Transforming lab reports from busywork to meaningful learning opportunities." *Journal of College Science Teaching 35*, 31-33.

21. Sageev, P. Bernard, K., Prieto, F., Romanowski, C. (2005)."Safe passage through the engineering curriculum: guiding subject experts toward integration of communication instruction and outcomes assessment." *Proceedings of the International Professional Communication Conference,* 139-146.

22. Adamczyk, B., Blauch, A. (2005). "Work in progress—Unified technical writing guidelines for engineering courses." *ASEE/IEEE Frontiers in Education Conference*, F3E-10-F3E-10.

23. Writing-Enriched Curriculum Program, University of Minnesota. http://www.wec.umn.edu/

24. Bean, J. C. (2001). *Engaging ideas: The professor's guide to integrating writing, critical thinking, and active learning in the classroom.* San Francisco: Jossey Boss.

25. Dannels, D.P. (2002)." Communication across the curriculum and in the disciplines: Speaking in engineering." *Communication Education, 51*(3), 254-268.

26. Cunningham, S.J. (1994)."Learning to write and writing to learn: integrating communication skills into the computing curriculum." *Proceedings of the IEEE Software Education Conference*, 306-312

27. Larkin-Hein, T., Budny, D. (2001). "Learning the "write" way in science and engineering." *Proceedings of the Frontiers in Education Conference*, T1B.7-T1B.13.

28. Wang, A. I., Sorenson, C.-F. (2006). "Writing as a tool for learning software engineering." *Proceedings of the Conference of Software Engineering Education and Training,* 35-42.

29. Hendricks, R.W., Pappas, E. (1995). "Writing- and communications-across-the-curriculum in the Materials Science and Engineering Department at Virginia Tech." *Proceedings of the Frontiers in Education Conference*, *2,* 4a4.10-4a4.14.

30. Stevens, D.D., Levi, A. (2005). *Introduction to rubrics: An assessment tool to save grading time, convey effective feedback, and promote student learning.* Sterling, VA: Stylus.

31. Rice, R., Boysen, A., Stetler, L. (2004). "Assessing oral presentation and writing skills. *Proceedings of the International Professional Communication Conference*, 147-150.

Biographical Information

Janet Burge is an Associate Professor in the Miami University Computer Science and Software Engineering Department. She received her Ph.D. in Computer Science from Worcester Polytechnic Institute (2005) and performed her undergraduate work at Michigan Technological University (1984). Her research interests include design rationale, software engineering, AI in design, and knowledge elicitation. She is a co-author (with Jack Carroll, Ray McCall, and Ivan Mistrik) of the book "Rationale-Based Software Engineering". Dr. Burge is a recipient of a NSF CAREER Award for her project "Rationale Capture for High-Assurance Systems". She has been at Miami University since 2005. Prior to that point, she worked for more than 20 years in industry as a software engineer and research scientist.

Paul Anderson is the Roger and Joyce L. Howe Director of the Howe Center for Writing Excellence at Miami University, Oxford, Ohio. His publications on technical communication have won awards from the National Council of Teachers of English and the Society for Technical Communication. His textbook, Technical Communication: A Reader-Centered Approach, is in its seventh edition. His current research focuses on the ways college faculty in all disciplines can help their students develop high-level writing abilities in college. Anderson is a Fellow of the Society for Technical Communication, Association of Teachers of Technical Writing, and Miami University's Institute of Environmental Sciences.

Michael Carter is Professor of English and Associate Dean of the Graduate School at NC State University. His research interests are in writing and rhetoric with a particular interest in the connection between writing and learning in the disciplines. He is the author of Where Writing Begins and many articles in a variety of journals. He has been PI or co-PI on NSF grants, including one that created LabWrite, an online instructional guide to writing better lab reports. He is currently working on projects related to teaching science in elementary schools.

Gerald C. Gannod is a Professor in the Department of Computer Science and Software Engineering and Director of the Mobile Learning Center at Miami University in Oxford, Ohio. He received the MS('94) and PhD('98) degrees in Computer Science from Michigan State University. His research interests include service-oriented computing, software product lines, mobile learning, software reverse engineering, formal methods for software development, software architecture, and software for embedded systems. He is a recipient of a 2002 NSF CAREER Award.

Mladen A. Vouk received the Ph.D. from the King's College, University of London, U.K. He is Department Head and Professor of Computer Science, and Associate Vice Provost for Information Technology at N.C. State University, Raleigh, N.C., U.S.A. Dr. Vouk has extensive experience in both commercial software production and academic computing. He is the author/co-author of over 250 publications. His research and development interests include software engineering, scientific computing, information technology (IT) assisted education, and high-performance computing and networks. Dr. Vouk has extensive professional visibility through organization of professional meetings, membership on professional journal editorial boards, and professional consulting. Dr. Vouk is a member of the IFIP Working Group 2.5 on Numerical Software, and a recipient of the IFIP Silver Core award. He is an IEEE Fellow, and a member of several IEEE societies, ASQ , ACM, ASEE, and Sigma Xi.