

**The Intersecting Futures of Technical Communication and Software Engineering:
Forging an Alliance of Interdisciplinary Work**

Ann Brady, Robert R. Johnson, Charles Wallace
Michigan Technological University
mabrad@mtu.edu, rjohnso@mtu.edu, wallace@mtu.edu

The future of technical communication has been explored and debated in several recent journal and book collections (Kynell-Hunt and Savage 2003/2004; Mirel and Spilka 2002; Albers 2005). Issues such as legitimacy, professionalism, professional status, the role of technology in the profession, and the current state of academic curricula have been addressed from a variety of points of view. Implied in many of these discussions is the issue of “interdisciplinarity.” Interestingly, the problem of interdisciplinary work lies beneath the surface, yet it appears to hold strong coin in most of the discussions of the field’s future. That is, work across disciplines is continually evident, and the word “interdisciplinary” surfaces from time to time. However, the term is most often used to point to the fact that working with other disciplines is central to technical communication: the work is so ubiquitous that the term is taken for granted. Discussions about interdisciplinarity are thus often impoverished because they impart so little new knowledge about the meaning of such work. An enriched conceptualization of interdisciplinarity, which we hope to introduce here, opens the pathways between the disciplines and makes them potentially more visible to all.

Speaking of Software is a multi-year project that has grown from an interdisciplinary experiment into an alliance between our two disciplinary spheres. As a result of our efforts, we have come to more fully appreciate the successes as well as the potential pitfalls of such work. In this article, we share with you the story of the formation of this interdisciplinary alliance and explain its potential for demonstrating how such efforts can play a small part in preparing the field of technical communication for some of the greater challenges we will face in the coming years. We begin with background on the two disciplines involved in this particular project. Next, we explain why this project came to be and how it has progressed. We conclude with a framework for understanding our interdisciplinary exchange, what its outcomes are, and how they are relevant to academic programs and the field in general.

The Departmental Alliance

Through the auspices of a National Science Foundation grant, two of us in humanities and technical communication (Ann Brady and Robert R. Johnson) have joined with a colleague in computer science and software engineering (Charles Wallace) to form just such an interdisciplinary alliance. The Humanities Department has an undergraduate Scientific and Technical Communication (STC) Program and a graduate Rhetoric and Technical Communication (RTC) Program. The STC and RTC programs have long sought to improve the practice of technical communication and to examine and teach it as a productive art and science. The Computer Science Department at Michigan Tech has a long history in training undergraduates in software development, and it now has an

undergraduate degree program in Software Engineering (SE). This program serves as the initial test bed for the project materials.

Titled *Speaking of Software: Integrating Communication and Documentation Techniques into an Undergraduate Software Engineering Curriculum*, the three-year project intends to increase opportunities for all those participating to view communication as a rhetorical act and to integrate the theories and pedagogies of technical communication and software development. More specifically, software engineering curricular materials that we produce draw from research in technical communication to develop case scenarios that sensitize students to the needs and knowledge of end users and other traditionally marginalized stakeholders in the development process. Further, we use software engineering tools and methodologies for documenting requirements and design. And finally, our evaluation methods, drawn from qualitative studies and usability techniques in technical communication, seek to determine the degree to which engineering students instructed in rhetorical analysis have increased their abilities to communicate with stakeholders.

Moving beyond the typical service approach that presents communication as a side-car for engineers and scientists, the project embedded technical communication practices into software engineering courses for the purpose of preparing software engineers for the communication realities of their professional lives. Grounded in interdisciplinary exchange, *Speaking of Software* thus exemplifies the kind of pedagogy needed to move technical communication forward to position the discipline for future growth.

The Disciplinary Alliance: Software Engineering Meets Technical Communication and Rhetoric

Software Engineering

Software engineering is concerned with establishing an engineering discipline for software development: a body of guidelines and repeatable best practices for producing quality software in a cost-effective way. Though it has historical and institutional ties to computer science, its scope extends beyond the mathematical and technical principles of computing. Software engineers draw not only from traditional computer science skills of modeling, coding, and analyzing computer programs, but also from principles of planning, team management, and cost estimation. A common factor that ties together all aspects of the software development process is precise oral and written communication.

In virtually every aspect of human endeavor, there is software that supports or entirely automates activities that were once managed or overseen exclusively by human workers. The contexts in which such software systems operate are highly complex. Consequently, among experts a rich body of know-how exists: artifacts and techniques for working in such domains (Johnson 1998). Developers ignore this knowledge base at their peril, since software will most likely fail its users if it is not informed by user experience. Developing software of such complexity requires communication with a wide variety of stakeholders (Poole 2003). These stakeholders include customers and end users, as well as other members of the development team: designers, quality assurance experts, technical

writers, and managers. Communication with other stakeholders is both vital and problematic. Several studies point to deficiencies in requirements as the primary cause of large-scale project failures (Curtis, Krasner, and Iscoe 1988; Davis 1990; Glass 1998). Because such failures can be traced to a lack of commitment and trust between customer and developer (Keil et al. 1998), the evidence indicates a need for improved communication during requirements elicitation and analysis.

Software engineering dovetails with technical communication in a number of ways. Effective written and oral communication is crucial in both disciplines. Software engineers use techniques of invention and design that are quite similar to those of the technical communicator. Finally, software engineers must be aware of the use of their software products, as well as other effects such as costs and safety risks. Consequently, they must be sensitive to the same issues of audience that concern technical communicators.

Technical Communication and Rhetoric

For the readers of this journal, it may seem unnecessary to define technical communication. Nevertheless, we have a particular approach that we want to clarify: an approach grounded in theories and practices of rhetoric. We define rhetoric as strategic communication since our project aims to introduce SE students to the rhetorical arts and skills of planning to communicate with stakeholders in a way that bridges the gap between technical imperatives and human needs. There is a clean fit between rhetoric and software engineering through the concept of heuristics. “Heuristic...was a part of rhetoric

and the sciences from Aristotle's time. It was the art of systematic inquiry and provided a method of gathering information about a problem and asking fruitful questions" (Young, Becker, and Pike 1970). Thus, the modern day technical communicator, like the ancient rhetorician, relies on these arts of knowing how to inquire, what questions to ask, in given situations in order to make appropriate communications for various audiences. In our project, we introduce software engineering students to these arts so that they can gather the kind of information they need in order to make their software products more useful to their project stakeholders.

The Problem: Exposing Software Engineering Students to Real Stakeholder

Interaction

At Michigan Tech, an undergraduate degree program in software engineering has been in place for several years. Since software engineering is an attempt to bring to software development a notion of repeatable process and a body of best practices, in the tradition of other engineering disciplines, Michigan Tech students receive traditional instruction in computer science, but they also get a higher-level perspective on the software process. By and large, however, the importance of communication in the software process is not reflected in current computer science and software engineering education. There are practical challenges. Typically, it is impractical to involve large numbers of students in real projects with real stakeholders. Students who do not participate in project-oriented courses get no exposure to the issues surrounding communication and those who do are thrust into a highly risky and sensitive situation with little previous guidance. There also

are problems due to the legacy of computing pedagogy. For instance, in introductory programming courses, collaboration among students is often viewed as plagiarism, not as a natural simulation of real software development. Software engineering research in this area has historically been rigid in nature and documentation techniques typically covered in software engineering classes are of a formal, developer-centered nature not suitable for customers and end users. Only recently have efforts begun to develop more natural and inclusive modes of communication.

As we began our project, we came to understand that software engineering students need exposure to communication with stakeholders different from themselves, and they need guidance in how to conduct such communication: moving beyond the cubicle-centered vision of software development and into a more humanistic process of active inquiry. They need to develop sensitivity to the format, time, and place of communication, and to the perspectives of others. Perhaps most importantly, they need concrete examples of strategies for encouraging and enriching communication.

A Solution: A Cross-Disciplinary Curriculum Development Venture

Our project at Michigan Tech aims to bridge this gap in current software engineering pedagogy by helping students solve not only the technical problems but also the communication problems that reside at the heart of software development. Our goal is to enhance the education of undergraduate students in software engineering by enlightening them about the needs and knowledge of other stakeholders, empowering them to engage

in active interchange with these stakeholders, and enabling them to communicate precisely and effectively about software requirements. To accomplish this, we draw from techniques in rhetoric and technical communication, applying them to live events that have been recorded and then edited to produce realistic case studies. We use qualitative methods to assess our success.

Students Learning About Stakeholders Through Cases

In order to give SE students practice in software production in realistic contexts and with a range of stakeholders, we decided to develop scenario-based cases. Cases can be described on a continuum of complexity ranging from single anecdotal stories to more involved collections of data and information gathered over a period of time and then distilled into significant pedagogical products. Our project resides on this latter end of the spectrum. We prepared a rigorous set of methods to build significant cases for the SE students. The process of doing so has allowed us to experiment with various methods, keeping some, refining some, and discarding some as we have progressed. At the same time, however, we are fully aware of the need to design methods of case development that are transportable to other pedagogical contexts and can be replicated with a less intense process. What follows in this section is an overview of our process as explained through one of the cases we have constructed and how we have worked toward a sustainable model of SE case development by having students participate in the case development process.

Case studies are used widely in the field of technical communication, for software as well as other technological artifacts. In this project, case studies are developed through a rhetorical approach: an approach that combines the arts of solving technical problems with those of creating strategic communications (Johnson and Simpson 1990). Further, this approach aims to create highly realistic case studies that allow students to better prepare themselves for situations they will confront outside the classroom (Couture and Goldstein 1985; Kynall and Stone 1999). Software case studies present stories using the vocabulary of the application domain, thereby conveying subtle but important contextual information. They can serve as the basis of an “inquiry method” (Rosson and Carroll 2002) that enriches stakeholders’ understanding of one another’s needs and constraints. Since they are expressed in plain language and mention specifics, they encourage the kind of constructive questioning that fleshes out important details (Sutcliffe 2003).

Text, audio, and video materials make up our case studies at Michigan Tech. Students can thus read cases, listen to audio recordings, and view animation clips as they answer questions that focus on a particular episode or theme. These “multi-media packages” are designed to convey the experience of interacting with stakeholders as realistically as possible. Information about the project requirements is revealed gradually, and, whenever possible, primary sources are included. Our aim is to expose students to real-world complexities, not predetermined conclusions.

Graduate students in the Rhetoric and Technical Communication program have played a central role in this project by taking the raw materials of each case, organizing them into

meaningful modules, and formulating questions intended to provoke inquiry into the case material. Once this organization and creation was complete, the case study material was encoded in DocBook XML. In this format, the XML file for each case can be used to export multiple HTML files that chunk the information into various modules based on the pre-defined structure of each document. DocBook XML can also easily be transformed into other formats, such as JavaHelp or PDF.

Examples of our online case study material are shown in Figs. 1 and 2. Both screenshots are part of the “Seabase” case study. In this project, Senior Design students worked with faculty and students in Mechanical Engineering to develop a ship-based crane for use on the high seas. Students faced critical communication challenges in this project. Not only did they have to learn the language of mechanical engineers, but a new programming language, MATLAB, as well. In addition, coming new to a project that was already underway, they had to find their place in an established and distinctly different work environment.

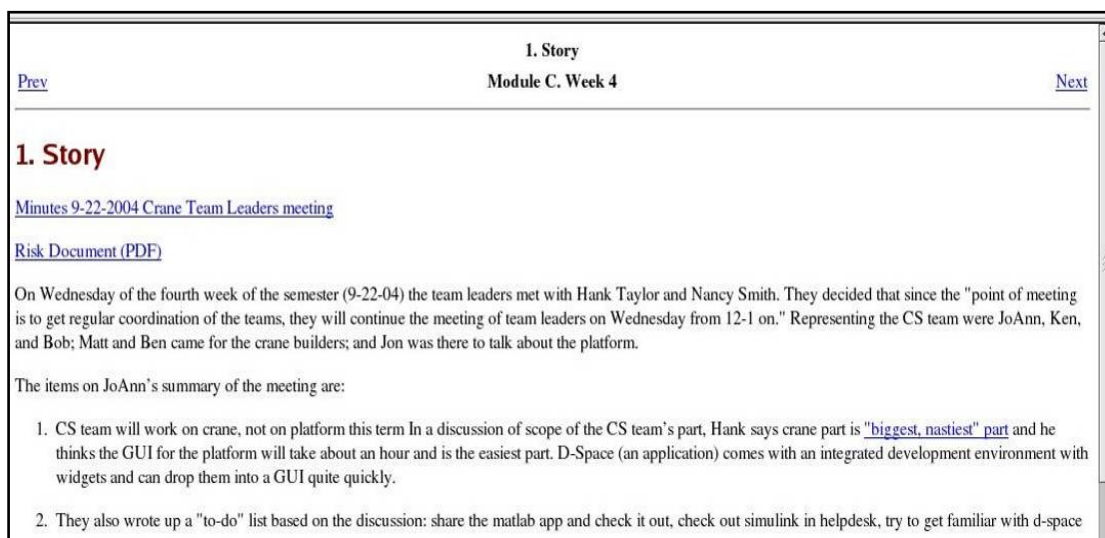


Fig. 1. Case study, modular perspective.

Fig. 1 shows a portion of the Seabase story from the chronological perspective. Here the developments of each week of the project are presented as a unit. Links to three primary sources are included: the meeting minutes for the project team leaders, the risk document of the software engineering team, and an email from a client in Mechanical Engineering. Students and instructors can thus follow the progress of the project module by module, gradually revealing more of the project outcomes.

Week	Date	Synopsis
2	9/6 to 9/10	Hank showed the CS Team simulink, d-base, and matlab
3	9/13	
4	9/20 to 9/22	share the matlab app
5	9/27	discussion: move into coding or working with matlab
6	10/4 to 10/6	LEARN matlab (faculty are starting to ask about it)
7	10/11	learn Matlab/simulink
8	10/18	
9	10/25 to 10/27	"I learned some interesting stuff; matlab is about code " GOALS: Nov. 3 Will have spent at least four hours with matlab
10	11/1 to 11/2	" decided we needed to talk to Hank " GOALS for this meeting: Get empty prototype in matlab going
11	11/8 to 11/10	divided up tasks into Bob = Simulink, JoAnn and Ken = dissect existing code
12	11/15	
13	11/29	
14	12/6	

Fig. 2. Case study, thematic perspective.

Fig. 2 shows a thematic perspective on the Seabase story. Here, all the events are related to learning the MATLAB language. Events are identified throughout the entire project history and links to them are grouped together in a single table. The thematic perspective allows instructors and students interested in a particular issue to move straight to the relevant story elements.

Using Qualitative Methods to Develop and Assess Case Studies

Developing Case Studies

Technical communicators use qualitative methods for a variety of purposes, among them assessing user needs and testing for usability and usefulness of developing products. We imported a number of these methods into our work with scenario-based cases. We used them as we developed the case studies and then as we assessed their effectiveness in increasing students' abilities to ascertain and address the needs and knowledge of their stakeholders. To develop the first case studies and pilot the methods, RTC graduate students gathered written material (*e.g.*, notes, meeting minutes, source code, email) from the Senior Design students. (Informed consent forms are on file.) Using standard qualitative methods (Agar 1996; Kirsch and Sullivan 1992; Lauer and Asher 1988), the graduate students observed and audio-taped students' meetings, took field notes, and conducted interviews with the students and their clients. At the end of the school year, the cases were then written up.

With that experience informing our methods, currently we are testing procedures that will allow us to reduce the active role of the graduate students, thereby reducing cost and time commitments. We have introduced one undergraduate Computer Science student to selected qualitative methods, and he is doing the actual observations and recording (under the supervision of a graduate student). Based on our success so far, we expect the procedures we are developing to be robust, and the project to become self-sustaining for the 2006-2007 school year.

At that time, Senior Design students will create and maintain written histories of their experiences. The documents that constitute these histories include rhetorical analyses of customers and end users, reports on students' communication with other designers and domain knowledge experts, risk documents built based on problem solving heuristics, progress reports and their portfolios. In addition, students will maintain participant observation records designed to capture information about their own software development procedures. During the summer after each Senior Design Project, graduate students and faculty will compile and analyze this data and write the subsequent case studies.

Assessing Case Studies

In the meantime, we are using qualitative methods to determine the degree to which engineering students instructed in rhetorical analysis can communicate with stakeholders in a way that bridges the gap between technical imperatives and human needs. More specifically, we are using qualitative methods to assess our project's three main goals. Our first goal is to enlighten students about stakeholders' needs and knowledge and here we use a questionnaire that indicates the extent to which students view project stakeholders from a rhetorical perspective. We are administering this before and after case study instruction. The questionnaire thus functions as a summative evaluation, assessing the progress students have made in considering stakeholders' needs and knowledge.

Our second goal is to empower students to be in active communication with stakeholders. Here, we videotape interviews simulating discussions among designers, customers, and clients to identify stakeholders' knowledge, needs, and goals. Using prompts, such as "What kind of knowledge, if any, would stakeholders have that might be useful to you as a designer," students develop a set of questions that they use to engage in active communication with their customers and end users. Videotapes are used as both formative and summative measures and are thus recorded before, during, and after case study instruction in rhetorical analysis.

Our third goal is to enable students to communicate precisely and effectively. Here we use portfolios as formative and summative evaluation measures, assessing first the ongoing and then the overall progress students have made in communicating precisely and effectively with a variety of stakeholders as they develop software for them. On a weekly basis, students are required to submit one-page progress reports, which may range from rhetorical analyses of customers and end users to reports on their own communication with other designers and domain knowledge experts. Using a rubric that identifies the rhetorical, mechanical, and organizational characteristics of precise and effective communication, instructors read these weekly reports in order to identify gains and lapses in students' audience awareness and adjust instructional goals for Software Quality Assurance accordingly. Instructors return evaluated reports to students who collect them in a file or portfolio. At the end of the term, students are asked to write a final one-page assessment of their written communication skills and to submit this with

their portfolio of written work. Instructors score portfolios for their rhetorical, mechanical, and organizational strengths and weaknesses.

Interdisciplinary Exchange in Three Stages

Interdisciplinary scholar Julie Klein (Klein 1990), drawing from the work of George Steiner, explains that interdisciplinary exchange is comprised of several stages, among them extraction, incorporation, and reciprocity. In the following section, we explain how the field of scientific and technical communication has benefited from the first two and how *Speaking of Software* suggests ways for us to engage with the third.

In the field of technical communication, we have asked two central questions over the past fifty years concerning the concept of “work”: "How do we work?" and "How do we work together?" To probe these questions, academic technical communicators have engaged in the interdisciplinary acts of extraction and incorporation. That is, we have entered other disciplines, brought back important findings, and then applied them to technical communication practices and pedagogy. As far back as the 1950's, for instance, we have turned to the nonacademic arena, commonly known as "the workplace," for a better understanding of how communication and collaboration function (Allen et al. 1987; Spilka 1990; Blyler and Thralls 1993). Expanding into areas such as interface and information design, we have developed new ideas and best practices based on concepts such as usability. And we have begun to develop explanations for the why, how, and what of our work. Socialization theory, for instance, has helped us understand composing

as a complex process, deeply influenced by the organizational contexts in which it occurs (Anson and Forsberg 1990; Lutz 1989; MacKinnon 1993).

Similarly, researchers in technical communication have explored other academic disciplines in order to identify and develop methods that can benefit our field. We have spent considerable time learning from psychology, human factors, sociology, computer science, law, philosophy, graphic arts, sociolinguistics, and management hoping to provide realistic and grounded pedagogies that will send our graduates into the workplace as properly informed professionals (Brady 2004; Cushman 1998; Dautermann 1997; Flower and Hayes 1980; Johnson 1998).

Using the interdisciplinary acts of extraction and incorporation, we have begun to answer the question: How do we work with other disciplines? Because we have accomplished this initial feat, we now have the opportunity to go one step further and deepen our understanding of how we can work together with other disciplines interactively. We have, in other words, the opportunity to reciprocate. *Speaking of Software*'s first acts of reciprocity, in compensation for the information STC has borrowed, is to develop case studies for use in SE classes and to provide SE students and faculty the methods they will need to develop future cases.

STC programs have incorporated concepts like problem solving from the field of computer science. With its emphasis on planning, designing, producing, and revising, problem solving has influenced the teaching of writing in significant ways. Two of these

are particularly relevant to our understanding of interdisciplinary work because they foreground its reciprocal and compensatory nature.

First, problem solving has offered STC teachers a way to talk about audiences as central to document production and review. Strategic document planning requires that writers familiarize themselves with the needs of their users and the contexts in which users will be working, that they design documents that aim to meet those needs, and that they produce prototypes and revise them based on the feedback their users provide. Those of us involved in the *Speaking of Software* project reciprocate by fostering the growth of SE students in three ways, all of which depend on audience awareness: an increased awareness of the needs and knowledge of all stakeholders, an increased ability to communicate with stakeholders, and an increased commitment to design products that are useful to stakeholders in the contexts in which they use them.

Second, with its inclusion of review and reflection, problem solving provides a rationale for using academic writing portfolios to assess students' writing achievements. In STC classrooms, the portfolio process often includes having students collect their written work during instruction and then review it as the term ends. Completed portfolios include statements that students write explaining their document selections and how their choices demonstrate their rhetorical skills. Taking the portfolio process into the SE classroom, we have used it both to foster and to assess the progress students make in communicating with a variety of stakeholders as they develop software for them.

The Act of Reciprocity: Impacts on The Future of Technical Communication

Programs

We feel that the time has come for the field of technical communication to complete the cycle of interdisciplinary exchange with the third “compensatory act of reciprocity that attempts to restore balance while enhancing and even enlarging the stature of the original” (Klein 1990). In this final section, we reflect on our project and pinpoint several ways that similar interdisciplinary projects can help the field develop in the coming years.

1. *Strengthening the presence of research methods in our curricula.* Research methods, qualitative or quantitative, have been something of an enigma for a number of years in technical communication programs, especially at the undergraduate level. What methods are most necessary? Should they be presented in stand-alone classes, or should they be integrated into other courses, or both? These and other questions can be vexing both in terms of curriculum design and material realities. In the Speaking of Software project, we haven’t been able to solve all of these problems, but we have had some fresh insights into how our students can learn about pertinent research methods.

First, we have seen how the contextualizing of qualitative ethnographic methods seems to make sense to students. That is, techniques of user observation similar to those used in usability research allow us to integrate such methods into the context of case, and eventually, software development. Second, this *in situ* use of the methods — methods meant to achieve tangible outcomes for the project — gives a strong applied “feel” to

ethnographic and qualitative usability research. As many of you are well aware, our technical communication students often express concern (even strong anxiety at times) that academic programs fail to prepare them adequately for the “real world” that they will soon be entering. While our project is not a replacement for internships or co-ops, it does allow for a situated practice that is in many ways reminiscent of the workplace atmosphere. In addition, the opportunity to practice research methods in actual moments of practice, ironically, lets us as instructors address issues of theoretical import more pointedly in the classroom, as we explain below.

2. Making theory more central to curricula. Theory has played a central role in the reciprocity of this project in two key ways. As we mention above, theories associated with the collection of and eventual interpretation of qualitative research data have allowed us to situate the practices of usability methods into case development and curriculum design. In another vein, however, we have been able to bring theories from software engineering (problem frames) and from technical communication (rhetoric) directly into the experiences of the students. In fact, the linking of these two theories comes directly to the fore when the software engineering students begin to see that the solving of a software problem is also the solving of problems for the stakeholders. In other words, the incursion of rhetorical theory into the problem-solving arena of software development enables the software engineering students to see their use of problem solving as an audience problem as well. As Hart-Davidson points out, while technical communicators operate within social, ethical, and political contexts every day, they are

often identified solely with the practical (2001). Projects such as this one can serve as powerful illustrations to students of the reciprocal nature of practice and theory.

3. Integrating external funding into programs. In the humanistic disciplines, where technical communication programs often reside, significant external funding is a somewhat rare phenomenon. By significant funding, we mean levels of support that can release faculty from teaching responsibilities, provide for undergraduate and graduate student assistantships, cover travel expenses for research and conference attendance, and procure computers or other equipment, among other things. Such funding is obviously welcome in our discipline, but putting it to its best uses is something that entails a fair amount of serious preparation on the part of the researchers, the home departments, and sometimes the greater institution.

In addition, external funding inevitably comes to an end. That is, while the funding is available there can be changes made to courses and curricula and initiatives that “spin-off” the funded project. Such bounty has associated costs, especially if these related outcomes have long-term consequences. In the early stages of a project, plans can be made for extending it beyond the original funding time of the grant. However, it is sometimes the case that such opportunities arise extemporaneously. Events like this call for further planning and may depend on garnering institutional support so that the changes can become part of the permanent fabric of given programs and curricula.

4. *Creating cross-disciplinary alliances for internal and external visibility.* Finally, we would be remiss if we didn't mention that part of disciplinary reciprocity involves making technical communication programs more visible, both to your home institution and to the greater arena of academe. While we may be somewhat visible institutionally due to our teaching of writing and design, our "political cachet" can only be enhanced through the procuring of external research funding. Universities are becoming increasingly intent on making external funding a major criterion of program, college, and institutional strength. Cross-disciplinary relationships forged through funded research are a strong way to become more visible and "value-added" to your school.

On the national level, it will become increasingly important for technical communication programs to become "players" in the interdisciplinary research game. We have the potential to become more visible in this way, and involvement in funded research is certainly one way to accomplish a few steps in this direction. There are other avenues to take, too, such as creating alliances among professional organizations and research institutes and centers. The opportune moment is upon us to do this now. Let's take advantage of this moment and think carefully about how to help lead the way in interdisciplinary research, scholarship, and pedagogy.

Acknowledgment. This work is supported by NSF Award #CCF-0417548.

Works Cited

- Agar, M. 1996. *The Professional Stranger*. San Diego: Academic Press.
Albers, M. 2005. Introduction to Special Issue on the Future of Technical Communication. *Technical Communication* 52 (3):267-272.

- Allen, N., D. Atkinson, M. Morgan, T. Moore, and C. Snow. 1987. What Experienced Collaborators Say About Collaborative Writing. *Journal of Business and Technical Communication* 1 (2):70-90.
- Anson, C., and L.L. Forsberg. 1990. Moving Beyond the Academic Community: Transitional Stages in Professional Writing. *Written Communication* 7 (2):200-231.
- Blyler, N., and C. Thralls, eds. 1993. *Professional Communication: The Social Perspective*. Newbury Park, CA: Sage Publishers.
- Brady, A. 2004. Rhetorical Research: Toward a User-Centered Approach. *Rhetoric Review* 23 (1):57-74.
- Couture, B., and J.R. Goldstein. 1985. *Cases for Technical and Professional Writing*. Boston: Little, Brown.
- Curtis, B., H. Krasner, and N. Iscoe. 1988. A field study of the software design process for large systems. *Communications of the ACM* 31 (11):1268-1287.
- Cushman, E. 1998. *The Struggle and the Tools*. Albany: SUNY Press.
- Dautermann, J. 1997. *Writing at Good Hope: A Study of Negotiated Composition in a Community of Nurses*. Greenwich, CT: Ablex.
- Davis, A. 1990. *Software Requirements: Objects, Functions, and States*. Upper Saddle River, NJ: Prentice Hall.
- Flower, L., and R. Hayes. 1980. The Cognition of Discovery: Defining a Rhetorical Problem. *College Composition and Communication* 31:21-32.
- Glass, R.L. 1998. *Software Runaways: Lessons Learned from Massive Software Project Failures*. Upper Saddle River, NJ: Prentice Hall.
- Johnson, R.R. 1998. *User-Centered Technology: A Rhetorical Theory for Computers and Other Mundane Artifacts*. Albany: SUNY Press.
- Johnson, R.R., and M. Simpson. 1990. Bridging the problem solver communication gap: Toward an art of professional case design. *Writing Instructor* 9 (3):109-120.
- Keil, M., P.E. Cule, K. Lyytinen, and R.C. Schmidt. 1998. A framework for identifying software project risks. *Communications of the ACM* 41 (1):76-83.
- Kirsch, G., and P. Sullivan. 1992. *Methods and Methodology in Composition Research*: Southern Illinois University Press.
- Klein, J.T. 1990. *Interdisciplinarity*. Detroit: Wayne University Press.
- Kynall, T.C., and W.K. Stone. 1999. *Scenarios for Technical Communication: Critical Thinking and Writing*: Allyn and Bacon.
- Kynell-Hunt, T., and G. Savage, eds. 2003/2004. *Power and Legitimacy in Technical Communication*. Vol. 1 & 2. Amityville, NY: Baywood Publishing.
- Lauer, J.M., and W. Asher. 1988. *Composition Research/Empirical Designs*. Oxford: Oxford University Press.
- Lutz, J.A. 1989. Writers in Organizations and How They Learn the Image: Theory, Research, and Implications. In *Worlds of Writing: Teaching and Learning in Discourse Communities of Work*, edited by C. B. Matalene. New York: Random House.
- MacKinnon, J. 1993. Developing Writing Ability in a Mature, Writing-Intensive Organization. In *Writing in the Workplace: New Research Perspectives*, edited by R. Spilka. Carbondale, IL: Southern Illinois University Press.

- Mirel, B., and R. Spilka. 2002. *Reshaping Technical Communication: New Directions and Challenges for the 21st Century*. Mahwah, NJ: Lawrence Erlbaum.
- Poole, W.G. 2003. The softer side of custom software development: Working with the other players. Paper read at Conference on Software Engineering Education and Training.
- Rosson, M.B., and J.M. Carroll. 2002. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. San Francisco: Morgan Kaufmann.
- Spilka, R. 1990. Orality and Literacy in the Workplace: Process- and Text-Based Strategies for Multiple-Audience Adaptation. *Journal of Business and Technical Communication* 4:44-67.
- Sutcliffe, A. 2003. Scenario-based requirements engineering. Paper read at IEEE International Conference on Requirements Engineering.
- Young, R.E., A.L. Becker, and K.L. Pike. 1970. *Rhetoric: Discovery and Change*. New York: Harcourt, Brace and World.

Biographies

Ann Brady is Assistant Professor of Technical Communication and directs the Scientific and Technical Communication Program at Michigan Technological University. Her research and teaching focus on the intersections of classical rhetoric, technology studies, and interdisciplinary theory.

Robert R. Johnson chairs the Humanities Department at Michigan Technological University. Johnson's research interests include technical communication theory and history, rhetoric history, and usability research. His book, *User-centered Technology: A Rhetorical Theory for Computers and Other Mundane Artifacts*, was awarded the 1999 Best Book Award from the National Council of Teachers of English for Publications in Technical and Scientific Communication. He is currently developing The International Center for Scholarship in Technical, Professional, and Scientific Communication at Michigan Tech.

Charles Wallace is Assistant Professor of Computer Science at Michigan Technological University. His research interests in software engineering include requirements analysis and formal approaches to specification and verification. He has helped to create and maintain the undergraduate Software Engineering degree program at Michigan Tech.