Chapter V
Speaking of Software: Case Studies in Software Communication

Ann Brady
Michigan Technological University, USA

Marika Seigel
Michigan Technological University, USA

Thomas Vosecky
Michigan Technological University, USA

Charles Wallace
Michigan Technological University, USA

ABSTRACT

We describe our recent efforts to generate and use case studies to teach communication skills in software development. We believe our work is innovative in several respects. The case studies touch on rhetorical issues that are crucial to software development yet not commonly associated with the field of software engineering. Moreover, they present students with complex, problematic situations, rather than sanitized post hoc interpretations often associated with case study assignments. The case study project is an interdisciplinary collaboration that interweaves the expertise of software engineers and technical communicators. Our software engineering and technical communication curricula have been enhanced through this cross-fertilization.

OVERVIEW

We argue that the art of *communication*, in its oral and written forms, is given relatively little attention in software engineering education, despite its fundamental importance in software development. Two major problems appear to prevent a more thorough treatment of communication issues. First, although software engineers may be effective communicators, they typically

do not have practice in articulating what it is that makes communication effective (or ineffective). That is, their knowledge remains at a tacit level, from which it is difficult to impart it to students. Second, part of what makes communication in the software workplace difficult is its intricacy and subtlety—"the devil is in the details". Students will not be convinced by toy examples; only realistic stories of software development will suffice. Yet the prospect of creating a communication setting of appropriate scale seems overwhelming.

Our ongoing interdisciplinary work seeks to address both of these problems. It utilizes the expertise of technical communicators, who are well versed in discussing and analyzing communication. Equipped with examples from software engineering, empirical techniques from ethnography, and analytical techniques from rhetoric, we have created *case studies* for teaching communication skills in software development, and we have used the case studies in upper-level courses in both software engineering and technical communication. Here we use the term "case study" not in its sense as a research tool in the social sciences, but rather in its sense as a *pedagogical* tool, currently used most prominently in law and business schools. Our case studies are based on the experiences of real software engineering students engaged in their capstone projects. The associated instructional materials touch on rhetorical issues not usually associated with software engineering: audience, active listening, critical analysis, timing, and planning. Moreover, they present students with complex, problematic situations, rather than sanitized *post hoc* interpretations often associated with case study assignments.

The case study project is an interdisciplinary collaboration that interweaves the expertise of software engineers and technical communicators. Our software engineering curriculum has been enhanced through this cross-fertilization—both by the insights into communication and by the qualitative methods employed in generating the cases. We report on the success of the project to date and describe some of the future directions we envision for this work.

MOTIVATION

We believe there is a significant gulf between the skills that students practice in academia and the skills they must use in the workplace. In this section, we show that practicing software developers acknowledge the importance of communication skills and expect new employees to have them. We then turn to the current state of software engineering education and comment on the status of communication skills in academia.

Communication in the Software Workplace

Within the lifespan of a single project, software engineers must engage with a wide range of stakeholders, with very different perspectives and goals (Poole, 2003). They must carefully elicit requirements from clients and keep them apprised of budget or scheduling changes. They must consult with end users to design products that provide both ease and value. They must also communicate within their development team, to maintain a clear vision of how to divide the labor and how to handle the project risks.

Stepping up from the level of individual projects to survey the software development landscape, we find an astounding variety of applications. No other engineered product has such a diverse set of potential uses. With this diversity of uses comes a diversity of stakeholders. In the span of a career, a software developer moves from project to project—and most likely from firm to firm—at each step negotiating a new application and a new set of stakeholders with widely varying knowledge, requirements, and communication styles.

Several studies point to deficiencies in requirements as the primary cause of large-scale