Session 3232

Improving Quality in Software Engineering Through Emphasis on Communication

Barbara Mirel
DePaul University/ University of Michigan

Leslie A. Olsen, Atul Prakash, Elliot Soloway
University of Michigan

Abstract

We will describe the integration of a communication component into a senior-level design course in software engineering, the structure of the component, and methods for testing its effects. The goal is to improve the usability of the software product, to overcome some problems due to time constraints of a school term, and to bring the course more in step with industry approaches by the following:

- educating students on techniques for defining a vision of the product (what is it doing and for whom),
- placing greater emphasis on the client's and user's perspective, the interface design, and interface's effects upon the rest of the code, and
- conducting iterative usability testing, starting early in the project cycle.

From inception to completion of the software, these important issues are addressed by teaching students to write well-reviewed specifications and user documentation, by beginning this early in the term, and by using these documents to inform the design.

Problem With Software Design Courses

In the computer industry today, it is rare to find a firm or in-house Information Technology department in which the dominant emphasis is not on user requirements, user needs, user involvement, and interactive software. Ideally, students should be prepared for this emphasis on user-centered design and development through their academic courses, especially through their culminating project course on software design and development. Without experience in these activities, students will not be in step with industry approaches.

Unfortunately, students rarely leave a software design course skilled in user-centered design. User-centeredness is not an orientation that comes naturally or even easily to most computer

engineering majors. Rather they have to learn it as an alternative perspective on program functionality and flexibility. This learning is hard to come by in a culminating software design course because of the time pressures of a single semester. In such a course, students largely become taken up with developing collaboration skills: how to divide and then consolidate work, how to conceptualize ideas into a focused product that they can create manageably in 10-12 weeks, and how to solve engaging technical issues during coding and testing (issues that motivated them to major in computer science or engineering in the first place).

Given the collaboration and technical issues they must deal with, students and professors have traditionally not had the time in one semester to add the activities and resulting documentation that user-centered design requires. These documents include the following:

User and task analyses based on contextual inquiry, activity-based planning, and scenarios.

A vision statement tying the product to a market niche and what it takes to fill that niche.

High level specifications reflecting users' points of view, including plans for interfaces.

User test plans for prototyping to guide the construction of instruction and interfaces that users need.

Progress reports on key trade-offs resulting from negotiating technical and user issues and the rationales behind them.

Specifications that "grow" with the product to reflect cuts and changes—or iterative prototyping that similarly reveals the evolving product.

User documentation aimed at users' actual tasks, not at descriptions of program functions and features.

Overview of Proposed Solution

We are adding these activities and documents to the capstone software design course by integrating a communication course with it. The goal is to improve the usability of the software products, to overcome time constraints, and to bring the course more in step with industry approaches.

The course described in this paper is designed to train software engineering students in user-centered issues at the same time as they are taking their software design course. This course is a 3-credit Technical Communications course fully coordinated with the pacing and assignments of the Software Engineering Design course, and is restricted to students concurrently registered for the software design class.

The course "Technical Communication for Software Engineering" was jointly conceived by technical communication and software engineering faculty, and is based upon the assumption

that writing the documents listed above will help software engineers become sensitive to user requirements and users' computing perspectives. Software engineering students need to realize that in the real world their program documents present their program not only to their teammates but to users, managers of users, marketing specialists, trainers, and others. The hope and experience is that once students learn to write for the needs and expectations of these audiences, they will transfer their insights about these needs and expectations to their programs and gain greater user-centeredness in their products.

"Technical Communication for Software Products" has the following three goals:

1 To improve students' team product, assuming that a user perspective will help them focus earlier and more sharply on a manageable design.

2 To use the writing of required documents as a means for generating ideas comprehensively and for negotiating tensions between technical issues and user requirements, with a clear sense of their resolution and the rationales for decisions.

3 To write and manage effective project documents as World Wide Web documents which are intended to "grow" with the product and to help keep it focused and unified. (Project documents include specifications, user documentation, progress reports, test plans, etc.)

To show how the course strives to achieve these goals, we will first briefly describe the process of its creation and then describe the strategies by which it engages students. We will conclude by presenting our methods for assessing its effectiveness.

Process of Creating the Course

The University of Michigan's Engineering College is moving to a comprehensive Communication-Across-the-Curriculum (CAC) program as part of a major curriculum revision, Curriculum 2000.[1] The CAC program targets engineering courses where students write significant documents as part of their engineering work and then works with the professors of these courses to extend and improve the communication that occurs in the engineering course. Typically, a technical communication expert meets several times with a technical professor to assess the professor's communication goals and needs, and—with input from the technical professor—designs ways in which the communication might be improved. This process was followed in creating the communication course described here.

Key Needs: The needs analysis with the software engineering professors identified several areas that required additional attention in the software engineering course:

- Team dynamics, negotiating conflicts, and following up face-to-face meetings with e-mail summaries.

- Design discussions that were thorough enough to avoid groupthink and to explore a full range of options and criteria for making decisions.

- User documentation that focused on real world tasks, not just on functions, features, and design rationales.

- Better program requirements specifications with a sense of real life users and needs.

- Talking to actual people who would use the product.

- More in-depth analysis in progress reports about problems and issues in the design process and strategies for dealing with them.

- More attention to effective World Wide Web documents, especially for texts that will undergo change.

These needs reflect those identified in the literature as indicated by the following:

> Software engineering is "an argumentative process in which the concept of problem and solutions emerges gradually ... produced by incessant judgment, and subject to critical argument...[For instance,] activities mean different things in programmming and in written instruction"[2]

> "Specification-in-the-large" is an activity in which there are many participants—clients, systems analysts, engineers, domain experts and so on. Each has differing perspectives on and knowledge about the object system, as well as a variety of skills and roles....In some cases, these perspectives may be based on contradictory understandings. To construct a specification the participants must cooperate, that is, contribute to the achievement of a joint goal."[3]

> "[I]f only the...features of work are considered in designing... and...the importance of learning is left out, there will be negative consequences in the conception and implementation of design."[4]

> "The continuing failure of the software industry to adapt and change is due to a fundamental flaw in the software development model. The flaw is an emphasis on engineering that discourages change and flexibility during development instead of an...attitude that fosters change and innovation during development."[56]

Description of the Communication Course

The curriculum in undergraduate computer science and engineering rarely concentrates on *integrating* effective communication and software engineering for the good of the product. Our curricular efforts strive to achieve this integration in a two-step process:
(1) some instruction within the software design course in designing for the World Wide Web and requiring that written documents be published as web documents and (2) a three-credit-hour course in relevant communication activities for software engineers taken concurrently with the

software design course. Each of these two steps includes a series of lectures, workshops, and team reviews of both written and technical student work in progress.

The course, “Technical Communication for Software Products,” is based on the assumption that writing things down is essential for managing a software project and for enhancing the usability and conceptual integrity of students’ team-developed products. This integration of writing and developing takes the following forms:

- Using writing strategies and user-centered perspectives to improve software.
- Designing hyperdocuments for change.
- Monitoring and assessing how writing works hand-in-hand—not as an afterthought—with programming and design.

Students are introduced to key distinctions between programmer and user perspectives, they learn strategies and methods for analyzing users and tasks (contextual inquiry, activity-based planning, scenarios of use), and they apply their insights to the software and the documents associated with it.

Documents such as program requirements specifications and user documentation keep teams focused on users—on their needs and on the real-world problems that the software aims to solve. The software that students develop will only achieve its aims if users find the meaning they need in the program. Meaning, conceptual integrity, and usability reside in actual use. They are not, as this course emphasizes, inherent traits of functions or features.

The writing that students produce and that they tie to their design and development includes the following:

- A vision statement.
- A user analysis and task analysis.
- A program requirements specifications which relies on the user and task analysis.
- Two progress reports, one as an oral briefing and one as a written document.
- User documentation.
- A user test plan and its implementation.
- A report of test findings or of the user documentation that is used for the test.
- An essay analyzing changes in the product and teammates’ individual contributions.

As these assignments suggest, students need to take multiple perspectives on their software and its presentation to audiences. They look at their product from the perspective of teammates, users, users' managers, program implementers, and testers. Though they rarely if ever do so in their software project class, they interact in this communication course with potential users, trying to understand users' tasks in context and to recognize their real world uses of the software. Ultimately, this enriched view should help the students to create better products.

Another communication aspect of this course is helping students to develop effective World Wide Web documents. The course particularly emphasizes some of the most recent findings from current research on the different demands of presenting *functional* web documents and of designing home pages or other single-screen web sites[6] .

Instruction highlights strategies for selecting appropriate content for various audiences and for developing the hyperstructures that support the tasks, purposes, and reading styles associated with each functional text. Students are guided in constructing effective links within and across documents. As they "grow" their documents in synchrony with their program, students maintain a record that keeps visible previously identified user needs and interface design issues. In addition, the skill of creating functional World Wide Web documents should readily transfer to students' future professional contexts where they are likely to communicate in this way over intranets and other electronic networks.

Key Logistics: The two courses avoid repetition but encourage overlap and reinforcement of concepts in both classes. This approach demands extensive coordination of the two courses and the many student projects and necessitates a number of changes to the normal logistics of each individual course.

- Most of the assignments are identical and due at the same time in both classes, which reinforces the idea that a single document must satisfy multiple needs for multiple users (and which avoids complaints that students have to do two different versions of a document). At the end of the semester, there is a slight deviation: the software engineering class requires a press release, while the communication class requires a user test and its documentation (either a test plan or a report of test results).

- The communication class has 10 teams; these identical teams (with the same students on each team) also function in the software design class.

- Only students in the software engineering course are allowed into the communication course.

- One person acts as World Wide Web administrator for a website shared by both the software design course and the communication course. The website includes all course notes, instructional materials, class exercises, student projects, and project reviews by professors and student peers.

- The communication course meets in a computer classroom since all documents and reviews of documents are Web based.

Assessment

The aim of our assessment is to discover if focussing on the audience and needs for software positively affects the nature of the software produced. If it does and if software engineers-in-training realize and apply these insights to their future professional projects, the course will contribute to user-centered efforts in industry. It also will establish a mutually beneficial partnership between faculty in software design and technical communications.

Continuing Methods for Assessment: There are several basic issues which need to be assessed in detail:

- In general, does the course improve the software product and development process?

- Do students have a greater appreciation for the role of user requirements and needs in the software development process? Does such an appreciation translate into improved software products?

- Do the iterative processes in the communication course make a difference, especially the rewriting/updating of specifications and the iterative prototyping?

- Does the writing in the communication course sharpen team discussions, design decisions, and trade-offs in the software design process?

These questions are being addressed through a series of methods as outlined below. The results from these assessments will be available at a later time.

Improvement of the software product and development process: We are comparing the team products produced by teams in the communication class with the products produced by additional teams in the software engineering class, measuring (1) completion rate (does writing help the team focus early on and help them complete their design), (2) ability to stay on schedule and meet milestones throughout; (3) frequency and quality of user testing or prototyping, and (4) evaluation—by expert evaluators—of user interfaces for user-centeredness.

Impact of rewriting/updating of specifications or iterative prototyping: We are comparing (1) students who develop their specifications iteratively with those who don't and (2) the final products of both groups. It's not a common practice in business today to update specifications—but failure to update is probably a bad practice. Findings from this class may help to shed light on that question for industry.

Impact of writing on sharpening team discussions, design decisions, and trade-offs: We are analyzing transcripts of e-mail meetings to see the type of design discussions the students have. We are also analyzing final essays by the students to find connections between the documents that were due, the ideas and tensions present in student discussions, and the reasons students had for the decisions they made.

Impact of user requirements and needs in the software development process: We are collecting retrospective comments from students at the end of the term from both an essay and a focused questionnaire.

Differences in the quality, scope and reader-centeredness of the documents: We are conducting expert evaluations of a sample of specifications and reports from the communication class and from non-communication students in the software design class and from an earlier software design class.

Conclusions

A communication component taught by a technical communication specialist can help a software design professor

- maintain a focus on goals, actual users, effective presentation to the user, real work tasks and problems, and usability;
- enforce a broad view of the project so students don't commit to coding a specific design too soon; and
- thus improve the conceptual integrity of the product.

The hope and experience is that once students learn to write for the needs and expectations of these audiences, they will transfer their insights about these needs and expectations to their programs and gain greater user-centeredness in their products.

[1] Undergraduate Curriculum Task Force, College of Engineering, University of Michigan. *Michigan Curriculum 2000: Proposed Changes to the Undergraduate Curriculum*. April 2, 1996.

[2] Bruce Blum (1996), *Beyond Programming*, pg. 138.

[3] A. Finkelstein and H. Fuks (1990), "Conversational Analysis and Specification," in P. Luff, N. Gilbert, and D Frolich (eds.) *Computers and Conversations*, New York: Academic Press, pg. 175.

[4] Patricia Sachs (1995), "Transforming Work," *Communications of the ACM* 38 (September), pg. 36.

[5] Tom Dayton et al.(1993), "Skills Needed by User-Centered Design Practitioners in Real Software Development Environments," *SIGCHI Bulletin*, pg. 18.

[6] Andrew Dillon (1996), "TIMS: A Framework for the Design of Usable Electronic Text." In H. van Oostendorp and S. de Mul (eds.*) Cognitive Aspects of Electronic Text Processing*. Norwood, NJ: Ablex, 99-119.

[6] Andrew Dillon (1996), "TIMS: A Framework for the Design of Usable Electronic Text." In H. van Oostendorp and S. de Mul (eds.*) Cognitive Aspects of Electronic Text Processing*. Norwood, NJ: Ablex, 99-119.