Enabling Schoolteachers to Participate in the Design of Educational Software

Marian G. Williams
Center for Productivity Enhancement
University of Massachusetts Lowell
One University Avenue
Lowell, MA 01854 USA
Tel: 1-508-934-2630
E-Mail: williams.chi@xerox.com

ABSTRACT
The introduction of new hardware and software can change the way that teachers do their jobs, yet teachers are often neither enabled nor permitted to participate in the design of the new technology. This paper looks at the barriers that keep teachers from such participation and suggests one approach to overcoming them. The approach is to provide a translator who is familiar with the work and workplaces of both teachers and software developers and who can bridge the inter-occupational communication gap. A case study in the participatory design of educational software using a translator is presented. Studying translation for clues to what a translator does and how he or she does it may yield information that is useful to design team members who do not have the benefit of working with a translator. Such information would be useful for design efforts beyond the realm of educational software.

KEYWORDS: participatory design, translation, software design, educational software, case study.

INTRODUCTION

Schoolteachers need to be enabled to participate in the design or customization of software tools that they will use on the job. As with any group of workers, the introduction of a new software tool into the teachers' workplace may impact not only the way work is done, but working conditions as well. Many public school teachers face barriers to participation in design sessions, even though their participation is crucial to the design process.

This paper looks at public school teachers as workers and at the public school as their workplace. This is an uncommon point of view. Schools are most often studied as the students' learning-place, not as the teachers' workplace. Similarly, educational software is most often regarded as a learning tool for students, not as a teaching tool for teachers. As a result, software is often introduced into a school without benefit of teacher involvement and without regard for its effect on how the teachers do their work.

In *PDC'94: Proceedings of the Participatory Design Conference.* R. Trigg, S.I. Anderson, and E.A. Dykstra-Erickson (Eds.). Chapel Hill NC USA, 27-28 October 1994. Computer Professionals for Social Responsibility, P.O. Box 717, Palo Alto CA 94302-0717 USA, cpsr@cpsr.org.

There are, of course, educational software successes. For instance, there are teachers who are computer literate and who, by evangelizing, help some of their colleagues also to become computer literate. Similarly, there are organizations developing educational software who engage teachers actively in design teams. These success stories are far too rare. For graphic descriptions of the unsuccessful introduction of computers into public schools in the US, see [Branscum, 1992; Piller 1992].

In many parts of the USA, including Massachusetts, public school faculties have shrunk because of budget cuts and changing enrollments. The youngest teachers, the ones most apt to have had computer training, have lost their jobs through reduction in the size of the teaching workforce. The remaining teachers, in many cases, have little or no familiarity with computers. For example, in the case study described in the next section, only two of the 10 teachers we worked with had substantial experience using a computer, and their experience was limited to word-processing.

Teachers are domain experts, both in curriculum development and in their subject areas. We need to learn how to give teachers, especially those who are not computer-literate, a real voice in the design or customization of the software they will use on the job, in the classroom. In particular, techniques are needed to secure teachers' participation beginning with the earliest stages of design, well before the formative evaluation of prototypes. This effort complements the work of other researchers who are studying participatory design with workers as diverse as nurses [Bjerknes and Bratteteig, 1987], architects [Peng, 1992], and software engineers [Muller, 1991].

In their retrospective survey of participatory design projects, Clement and van den Besselaar [1993] looked at projects that took place in a variety of countries and a variety of workplace settings. They observe that the five "prime ingredients" in a participatory design project include some or all of the following: (1) access to relevant information; (2) an independent voice in decision making; (3) usercontrolled development resources, including time, facilities, and expertise; (4) organizational and technical flexibility; and (5) appropriate development methods.

Of these ingredients, the first and third are common to all of the projects that Clement and van den Besselaar surveyed. Looking at why these ingredients are difficult to achieve in the teachers' workplace is helpful in illustrating the barriers to participation faced by schoolteachers.

Teachers do not have control over their time. A teacher's workday is tightly scheduled; the schedule is not flexible. A one-hour "prep period" is a luxury, and is not nearly sufficient for preparing a day's lessons and grading a day's assignments. Many teachers spend long hours outside of school on preparation and grading. Getting released from regularly scheduled work (for example, to participate in a training session) requires the expense of paying substitute teachers and the work of preparing lessons for them to teach. As a result, teachers are not apt to be freed up to work on a project such as software design.

Most teachers do not have access to on-the-job computer training. The teachers who have participated in our projects tell us that teachers are asked to solve more and more problems, ranging from the educational to the societal, without being given any additional resources to bring to bear. It is perhaps not surprising, then, that many teachers are expected to learn to use computers on their own time and of their own initiative.

In addition, teachers have no formal voice in how computer technology is introduced into schools. Although teachers in the US are unionized, their unions have been slow to develop policies about giving teachers individual and collective voices in how such technologies are incorporated into curriculum.

Appropriate development methods, in Clement and van den Besselaar's words, must come to the teachers' rescue. Techniques must be employed that enable teachers to participate in software design and customization despite the barriers to their learning about computers. Such techniques must allow them to communicate with software developers even though they are unfamiliar with the software domain and even though the software developers are unfamiliar with theirs.

A key issue here is to provide *translation* between users and software developers [Williams and Begg, 1992, 1993a, 1993b]. The translation involves not only the different terminology used by teachers and software developers, but also the understanding of each other's work and workplace. We have successfully used a former-teacher-turned-computer-scientist as the translator in our projects. The translator understands not only the languages used by teachers and by software developers, but also the detailed nature of their work and the conventions of their workplaces. The translator can employ participatory design techniques to make sure that there is a meeting of the minds between teachers and software developers.

The next section develops the notion of translation further. It is followed by a case study of the participatory design of educational software, and then by a description of the future directions of this research project.

TRANSLATION

Design team participants from different domains use different languages, have different ways of viewing their work, and have different workplace conventions. These differences can make interdisciplinary communication challenging. Translation deals with building bridges to resolve the differences..

The focus of this research is on people who can serve effectively as translators between users and software developers, because they have work experience in both domains. (The degenerate case of translation is in the design of software for one's own use, where one is simultaneously the user and the developer.)

Harrison et al. [1994] speak of the importance for members of interdisciplinary design teams to learn each other's terminology early in the design process. Speaking from experience, they say, "We often called a rose by many names only to later discover we all meant a rose. We also often thought we were all talking about a rose when we meant a petunia, a lily and a dandelion" (p. 129). Members of the design team need a set of common terminology to avoid talking at cross purposes.

A translator can facilitate the convergence on a common vocabulary, since he or she is familiar with the terminology of both the users' and engineers' domains and understands the differences between apparent synonyms. Such understanding comes from detailed knowledge of the work of both domains. Thus the translator in the case study which follows can compare and contrast various names for a piece of writing (essay, file, story) without needing to apply the methods of, say, ethnography [Blomberg et al., 1993] or contextual inquiry [Holtzblat and Jones, 1993] for work language analysis. The effort expended by design teams that have to resolve their own language differences is illustrated in [Katzenberg and Piela, 1993].

More complex than terminology is the view of work and workplace, or what Harrison et al. [1994] call the "definition of the situation." This definition may include the goals of the design project as viewed by various design team members, the nature of the tasks to be performed with the software under design, the conventions of the users' workplace, as well as workplace politics. Design team members view each of these complex issues in different ways. Non-users on the team can make erroneous assumptions about the users' work and workplace, based on the conventions of their own fields. A translator can identify and correct such assumptions.

The description of translation should not be taken to imply that any career-changer who has become a software professional can serve effectively as a translator. One of the interesting issues in studying translation is to consider the characteristics, other than domain knowledge, that make a person a successful translator between users and developers. The idea of establishing a registry of software professionals with expertise in domains outside of software is a provocative one, though.

The case study which follows describes a participatory design project with schoolteachers. The project relied on translation between the domain of writing instructors and the domain of software engineers. It illustrates the issues of translation between terminologies and workplace conventions.

CASE STUDY

This case study concerns the introduction of new computer technology into the curriculum of a public-school English department, which previously had no computer technology available in the classroom.

The computer technology in question is an editorial system, which consists of the networked hardware and software used by reporters and editors on the staff of a newspaper or magazine. A premise of the project is that teachers and students can profitably interact in ways analogous to the interaction of editors and writers in a newsroom.

An editorial system is designed especially for creating, sharing, and editing newspaper or magazine articles. It is not a general-purpose computing environment, but rather is highly customized for writing and editing. It provides basic support for collaboration between writers and editors, though not the sophisticated tools of state-of-the-art groupware. Because newsroom staff create the content of articles but are not responsible for the appearance of the final page, an editorial system does not include a page layout facility.

Through of the generosity of a corporate sponsor (Atex Publishing Systems of Billerica, MA, USA), an editorial system was donated for this project. The grant included not only hardware and software, but also installation, the time of some engineers to perform customizations, two days of training, and hot-line help.

The editorial system was installed in the English department of a four-year public high school in a middle-class suburb of Boston. The school had 10 English teachers (eight full-time and two part-time) and an enrollment of 900 students in grades nine through twelve. The editorial system is currently in its second year of use at the high school. Each sophomore and junior English class uses the editorial system lab, instead of a traditional classroom, for one out of four terms. Thus, the lab is used by a total of 450 students per year. All of the English department's teachers chose to be trained to use the editorial system.

The editorial system's hardware consists of 22 terminals, which are a mixture of Atex terminals and IBM PCs. All of the terminals, whether Atex or IBM, have Atex proprietary keyboards, which are highly customized for writers and which have a large number of dedicated or programmable keys for word-processing functions. The terminals hang off a customized Digital Equipment Corporation minicomputer. Print-outs are done on a laser printer.

The software is Atex's basic editorial system, used worldwide by newspaper writers and editors. The software offers traditional word-processing capabilities, nearly all of which are available via special keys on the keyboard. It also offers elementary features for cooperative work by writers and editors. These features include red-lining (which permits an editor's corrections, additions, deletions, and comments to be distinguishable from the writer's original text), electronic mail, and file sharing.

Whenever an editorial system is introduced into a new workplace, it is customized to reflect local work practices. The Atex engineers had plenty of experience tailoring editorial systems for the individual newsrooms of particular publishing operations. However, customizing an editorial system for a classroom was outside their experience.

Despite the obvious similarities between a newsroom and a writing classroom, an editor's work differs from a teacher's work. The editor's goal is to ensure that high-quality articles are produced for publication. The teacher's goal is to ensure that high-quality writing instruction is delivered to students, who will then, it is hoped, produce high-quality pieces of writing. Yet the editorial system, appropriately customized for each workplace, proves a useful tool for both jobs.

The teachers are domain experts in the teaching of writing. They had no expertise in software development. Their lack of computer knowledge made it impossible for them to direct the customization of the editorial system or even to carry on useful dialogues with the software engineers. They had neither the skills to evaluate the existing capabilities of the editorial system nor the skills to tell the software engineers what customizations they needed. They were unfamiliar with basic computer-ese, such as "file," "directory," "username," and "electronic mail." They also did not have the skills to look at differences in workflow in a paper-based vs. a computer-based classroom. That is, they lacked the formal skills to analyze and articulate their work practices in a way that would be useful to the engineers.

Moreover, the teachers did not control the resources that would have enabled them to develop computer skills and knowledge in preparation for discussions with the engineers. They were given no release time to learn about computers or to work on the customizations. They had no ready access to computers in the school. And they had no in-house expertise in software development to draw on. In addition, the teachers felt inferior to the software developers. The tended to refer to their lack of knowledge as stupidity and spoke about their fear of being unable to understand and use computers.

The engineers, on the other hand, were knowledgeable about the editorial system and about workflow in a newsroom. They were used to newspaper jargon. In fact, a lot of that jargon had been incorporated into the editorial system. Where a teacher would "throw away a paper" and a computer person would "delete a file," a newspaper person would "spike a story," a metaphor for the traditional metal spike on which papers to be discarded could be impaled. In the editorial system, a file is deleted by sending it to the "spike queue." The engineers' language was a mixture of computer talk and newspaper talk, while the teachers spoke the language of writing instruction.

Having spent time in the classroom as students, the engineers had general assumptions about what teachers do, but were not aware of specific activities and conventions of the workplace. They needed to be encouraged away from an I-know-what-the-teachers-need point of view. They had spent plenty of time in classrooms watching teachers, and so had a sense that they knew the nature of the teachers' work, when in fact they did not. (A trivial example of their unfamiliarity with teachers' workplace realities was their suggestion to use "stud" as an abbreviation for "student" in naming some computer accounts.) Their job was to perform customizations, but they had neither sufficient time nor sufficient knowledge of the teachers' workplace to lead the design of those customizations.

The design effort was lead by the author, a university researcher who is a former English teacher turned computer scientist. The author served as a translator to enable the high school teachers to participate in the customization process by translating between their language and workplace conventions and those of the software engineers.

The translation process had these steps: work with the teachers to develop a description of writing-related activities currently used in their paper-based classrooms, including workflow and paperflow; determine how these activities could be carried out and extended, based on the existing and potential capabilities of the editorial system; figure out the implications of the computer-based versions of these activities on information flow, security, and usability; teach the teachers, in their own language and in terms of classroom activities, about features that would extend their writing-instruction activities, in order to enable them to decide if they wanted the extensions; translate the descriptions of the activities from the language and workplace conventions of the teachers to the language and conventions of the engineers; and work with the engineers to specify the customizations to the editorial system.

The customizations to the editorial system address issues such as how to support the teaching of *process writing,* in which students are introduced to the process of evolving a piece of writing through multiple drafts; how to discourage plagiarism; and how to help students have easy access to the materials they need to have at hand.

Consider, for instance, this example concerning the versioning of files. In a typical newsroom, there is only one current version of a "story," the most recent draft. Either the editor has it or the writer has it. When one of them sends the story to the other, the story disappears entirely from the sender's workspace and now appears only in the recipient's workspace. Earlier versions of the story are stored in an archive and are retrieved only if disaster strikes the current version. What matters in the newsroom is the product, not the process by which it is created.

By contrast, in an English classroom where process writing is taught, students are expected to maintain a library of the various drafts of an essay. When a student submits a draft to the teacher, the student should retain all earlier versions, as well as a copy of the current version, in his or her workspace. Both the product and the process by which it is created are important. The versioning capabilities of the editorial system were customized to reflect these differences between the newsroom and the classroom. When a student submits a paper to the teacher electronically, a new version number is automatically assigned, and all drafts are archived in the student's own workspace. Several additional examples of customizations to the editorial system may be found in [Williams and Traynor 1994].

We hope to interest an educational researcher in evaluating the editorial system's effect, if any, on student learning. In the meantime, anecdotal evidence from the teachers suggests the usual motivational value of successfully integrating computers into the curriculum (non-attending students start attending class, students who appear passive in the traditional classroom are actively engaged by the computers, etc.). The teachers also report that students are giving greater attention to the writing task and that they take the teacher's comments more seriously when they are embedded in an electronic essay than when they are written in the margins of an essay submitted on paper.

The project has had workplace side effects for the teachers, right from the beginning. What added most to the teacher's work load was learning to use the editorial system and incorporating it as a tool into the writing curriculum. Also, because course times were rescheduled to maximize the use of the editorial system lab, teaching assignments underwent substantial changes (note that these changes were implemented for the beginning of an academic year, and did not happen disruptively during the year). Thus, teachers found themselves teaching courses that they had never taught before, teaching more different courses at the same time than previously, or not teaching favorite and familiar courses. The project has also brought increased visibility to the department. While the teachers appreciate the good publicity that the project has brought them, they are aware of being in the public eye and have had to deal with a certain amount of jealousy from their colleagues in other departments. The teachers have also had to adjust to the fact that their students are, in many cases, far more at ease with computers than many of the teachers will ever be.

FUTURE DIRECTIONS

Retrospective studies are being performed of two projects that introduced new computer technologies into the classroom (the project described above and a computer science project described in [Williams et al., 1992]). The goal of the retrospective studies is to formulate a set of guidelines for the participants in educational software design teams to help them break down barriers to participation. In addition to teachers and software developers, such teams may include administrators, personnel from sponsor corporations, and university researchers. The guidelines are intended to serve not only as a source of logistical advice to all of these groups, but also as a position paper on the importance of empowering teachers to participate in the design of software that will change the way they do their jobs.

The guidelines will explicitly address the need for teachers to be provided with the resources to participate in design. These resources include release time from their other duties for design sessions, curriculum revision, and training; ready access to new hardware and software outside of training sessions; and participatory techniques that enable teachers and software professionals to communicate effectively.

The guidelines will be used in a new project, in which a Geographic Information System (GIS) will be introduced as a teaching tool in an interdisciplinary curriculum. This new project will be the subject of a pro-active study of participatory design with school teachers. The design team will include a translator, another former teacher turned application developer. Data will be collected both by observation during the design process and by surveying and interviewing participants after the design process.

SUMMARY

Schoolteachers are frequently at a disadvantage during the design or customization of educational software. As with any group of workers, the adoption of a new software tool may change the way they do their jobs, and may have profound workplace repercussions. Yet, like many other groups of workers, teachers face barriers to participation in the design process. These barriers may include a lack of familiarity with computers, and a lack of resources (especially time) to gain that familiarity. Additionally, teachers may feel at a disadvantage to software professionals, and so may perceive themselves to be powerless to influence the design process.

Translation is an approach to enabling users such as these teachers to participate in design. A translator with experience both in the users' domain and in the software development domain can overcome interdisciplinary communication difficulties by translating not only between the terminology of the two domains, but also between the workplace conventions of the domains. A translator can also help to restore a real or imagined power imbalance between users and software developers.

The translator in the case study was vital to the customization of the editorial system, since the teachers and engineers did not have knowledge of the each other's work and workplace, and did not have sufficient resources to develop such knowledge. The translator was accepted by both teachers and engineers as somebody who spoke their language and who knew how their work responsibilities were being affected by the project. Studying the contribution made by the translator should reveal additional techniques for empowering teachers to participate in software design.

In addition, studying the nature of the translation activity may lead to techniques that other, non-translator, facilitators can use to mediate the design activities of interdisciplinary groups. Such techniques would be valuable beyond the domain of educational software

ACKNOWLEDGMENTS

This research is funded in part by U.S. Department of Education grant number R117E30237, "Improving the Success Rate of Adapting Technology for the Classroom."

The project that forms the basis of the case study could not have taken place without the generosity and commitment of Atex Publishing Systems and the vision of vice president Jo Hoppe, nor could it have happened without the courage and dedication of the teachers who embraced the editorial system as a new teaching tool.

REFERENCES

Bjerknes, G., and T. Bratteteig. (1987). Florence in Wonderland: System Development with Nurses. In G. Bjerknes, P. Ehn, and M. Kyng (Eds.), *Computers and Democracy: A Scandinavian Challenge*, Aldershot, England: Avebury, pp. 279-295.

Blomberg, J., J. Giacomi, A. Mosher, and P. Swenton-Wall. (1993). Ethnographic Field Methods and Their Relation to Design. In Schuler, D. and A. Namioka (Eds.), *Participatory Design: Principles and Practices*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 123-155.

Branscum, D. (1992). Conspicuous Consumer: Educators Need Support to Make Computing Meaningful. *MacWorld*, September 1992, pp. 83-88.

Clement, A., and P. van den Besselaar. (1993). A Retrospective Look at PD Projects. *Communications of the ACM, Special Issue on Participatory Design*, **36**(6), pp. 29-37.

Harrison, B., M. Mantei, G. Beirne, and T. Narine. (1994). Communicating About Communicating: Cross-disciplinary Design of a Media Space Interface. In *Proceedings of CHI '94 Human Factors in Computing Systems* (April 24-28, Boston, MA), New York: ACM, pp. 124-130.

Holtzblat, K. and S. Jones. (1993). Contextual Inquiry: A Participatory Technique for System Design. In Schuler, D. and A. Namioka (Eds.), *Participatory Design: Principles and Practices*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 177-210.

Katzenberg, B. and P. Piela. (1993). Work Language Analysis and the Naming Problem. *Communications of the ACM, Special Issue on Participatory Design*, **36**(6), pp. 86-92.

Muller, M. (1991). PICTIVE – An Exploration in Participatory Design. In *Proceedings of CHI '91 Human Factors in Computing Systems* (April 27-May 2, New Orleans, LA), New York: ACM, pp. 225-231.

Peng, C. 1992. "Participatory Architectural Modeling: Common Images and Distributed Design Developments," In *PDC'92: Proceedings of the Participatory Design Conference*, (November 6-7, Cambridge, MA), Palo Alto, CA: CPSR, pp. 171-180.

Piller, C. (1992). Separate Realities: The Creation of the Technological Underclass in America's Public Schools. *MacWorld*, September 1992, pp. 218-230.

Williams, M. G., and V. Begg. (1992). Translation in Participatory Design. In *PDC'92: Proceedings of the Participatory Design Conference*, (November 6-7, Cambridge, MA), Palo Alto, CA: CPSR, pp. 113-14.

Williams, M. G., and V. Begg. (1993a). Translation between Software Developers and Users. *Communications of the ACM, Special Issue on Participatory Design*, **36**(6), pp. 102-103.

Williams, M. G., and V. Begg. (1993b). Translation in Participatory Design: Lessons from a Workshop. In *Adjunct Proceedings of INTERCHI '93* (April 24-29, Amsterdam), pp. 55-56.

Williams, M. G., C. Theriault, A. Stowe, and J. T.Canning. (1992). Revitalizing High School Computer Science. In *Proceedings of the National Educational Computing Conference (NECC '92),* (June 1992, Dallas, TX), pp. 181-186.

Williams, M. G. and C. Traynor. (1994). Participatory Design of Educational Software. In *Proceedings of the National Educational Computing Conference (NECC '94)*, (June 11-15, Boston, MA), Eugene, OR: International Society for Technology in Education, pp. 334-339.