

WHY DO WE REALLY WRITE PROCEDURES?

A writer I work with recently came into my office with a huge book on Microsoft *Project*. The book was published by a third party (in other words, not Microsoft). “Look at this book,” my co-worker said. “I challenge you to find the procedures in it.”

I flipped through the book. Sure enough, I saw pages and pages of paragraph text and plenty of headings, tips, and screen shots, but few stepped procedures.

“What I want to see,” my colleague continued, “are the procedures. I want to see how to do something and I want to see it spelled out step by step. I can’t waste my time reading all of this. How can they call this task-oriented documentation?”

At first, he seemed to have a point. But the more I thought about it, the more I saw why the book might have been written as it was. *Project* is powerful, complex software. To get the most value out of it, you have to be familiar with a host of terms and concepts that are seldom used outside the discipline of formal project management. What is slack? What is leveling? What are baselines, actuals, and variances? What is the relationship between work and duration?

Now, I’m not privy to the decisions

BY DOUGLAS R. WIERINGA
Senior Member, Puget Sound Chapter

that were made when that book was written, but I’d like to think that the authors realized that the biggest hurdle they faced was getting readers to understand the concepts of project management. (I’m not singling out *Project*, and Microsoft didn’t make up these terms. They’re commonly used in other project management software. And you can perform simple tasks in *Project*, like setting up a timeline, without being familiar with these concepts.) Once the users understood the concepts, the manipulations they would perform in dialog boxes, forms, and tables—the procedures—would be comparatively straightforward.

My colleague’s critique, on the other hand, was based on his feeling that technical documents should give prominence to procedures. This little episode got me to thinking about why procedures are so important to technical communicators, why we really write them, and why we should write them. Don’t get me wrong: I like procedures. I’ve co-written an entire book on them, in fact (*Procedure Writing: Principles and Practices*, Battelle Press, 1993). But it

seems to me that sometimes, technical communicators write procedures for the wrong reasons.

Why We Write Procedures

To begin with, there is a huge bias in our profession to write procedures. Our documents (and I’m talking primarily about software documentation here) don’t have readers, they have users, and we call them users because they don’t have time to read. They want to accomplish a task without having to read much, and we oblige by writing an easy-to-digest procedure. Do this, do that, do this, you’re done.

Sometimes users need conceptual information, though. I won’t go into the conceptual information-versus-procedure debate here. We’ve all wrestled with it. Instead, I would like to focus on something else that I think is going on: I think we sometimes write procedures because they’re easy to write and because we can write them quickly.

Conceptual Approach

Here’s an example. I recently wrote an installation guide for a voice mail system that stores voice messages and faxes in Microsoft *Exchange*, alongside e-mail messages. The audience for the installation guide consists of informa-

tion technology professionals, people who work with computer networks every day. During preliminary testing of the document, they gave us a clear message: The procedural topics were of relatively little value to them; once told what they needed to do, they knew how to do it. What they wanted to know was how our product worked, how it interacted with Windows NT and *Exchange* on their network. To take one example, in some configurations, the product requires that they set up *trusts* between different NT domains. You may not know what a *trust* is, but they do, and they don't need to be told the steps for setting one up. (And someone who doesn't know how to set up a trust shouldn't try to do it, because improperly set up trusts compromise network security.) But they didn't know the pros and cons of the different configurations, some of which require trusts and some of which do not. That's conceptual information.

In response to this feedback, I wrote several topics that explained our product's architecture, its principles of operation, and how it shared information with Windows NT and *Exchange Server*. Those topics were the toughest to write in the whole book. I could poke around the interface and knock out a decent procedural topic in an afternoon, but the conceptual topics required extensive interviews with and reviews by subject matter experts. These topics took many more hours to write and were rewritten more times than any of the procedural topics in the book.

That stands to reason. The product has a good graphical interface, and any reasonably skilled user could sit down and figure out enough to write a procedure for accomplishing a given task. True, such a procedure may fall short of editorial standards, but the fact remains that I was

adding much less value to the information when I was writing procedures than when I was writing conceptual topics. The procedures were virtually self-evident; the conceptual topics were not.

Least Resistance

Sometimes we write what's easiest. The reality of the documentation business is that writers can be thrust into situations where they are writing about something they don't fully understand. A typical (and understandable) response is to write what you can. On the project described above, I had access to people who could explain concepts

to me and I had the time to talk to them, so I could take on those difficult topics. Had those experts been unavailable, however, or had I been under a tight deadline, the conceptual topics might not have been written. I would have written procedural topics instead.

There's more going on, too. Procedures of questionable value aren't written only because writers find it easy to write them. Much writing is done by teams, and teams like procedures, perhaps because of a kind of group-think. As I've already said, the task-oriented documentation paradigm puts a lot of pressure on us to write procedures. It's



also easy to decide that users need procedures. User analysis and empathy will lead the team to hypothesize some user, somewhere, who may have forgotten the steps necessary to perform a task. So each person on the team will advocate procedures to one extent or another, and procedures become something that everyone can agree on. The team writes them because it is following the path of least resistance.

Further, I think that teams like procedures not only because everyone can agree that they should be written, but also because everyone can agree on the rules for writing them. We all know those rules. Procedures should break tasks into steps, each containing a single action or two closely related actions,

One of the most important things to understand about procedure writing is that the audience always brings a certain level of knowledge to the procedure.

and each step should begin with an action verb. Procedures should have headings that begin with infinitives (or gerunds). They should use cautions and notes in a particular manner. The list goes on. These rules are important—but that's not the point. The point is that writing good documentation goes way beyond following the rules, into the intangibles of subject matter knowledge and audience analysis. But sometimes we stick to criteria we can quantify, and we write documents that we can evaluate using those criteria. We don't go beyond those quantifiable criteria to evaluate how well our document communicates. We stay in a comfort zone.

In other words, teams write procedures because they feel comfortable working with them. Writers know the guidelines for writing procedures, reviewers know the guidelines for providing feedback on them, and graphic artists know the guidelines for laying them out. Problems arise when the team becomes too internally focused and, in following those guidelines, ignores the users' needs. That's especially likely to

happen when documentation teams don't fully understand what they are documenting or the audience they're writing for, and take refuge in what they do know—the rules for writing a good procedure. When the project is done, everyone's worked hard, followed the rules, and feels good. But the rules they have been following have arisen from the team's comfort working with procedures, not from the user's needs (assuming, of course, that the user needs something besides a procedure).

Procedures When Appropriate

I'm not saying we shouldn't write procedures. But we shouldn't automatically write procedures, or automatically write *just* procedures. We need to figure out what is best for a particular document. As the authors of the Microsoft *Project* manual did, we should figure out what types of information our readers need and include that information.

Some things to consider are the following.

Are the tasks self-explanatory?

Sometimes you need procedures. My early experience with was in the nuclear power and chemical industries. I don't think anyone would disagree that the interface to a nuclear power plant is more complex than the interface to any piece of software. You can't walk into the control room of a nuclear power plant and figure out how to operate the plant. You need procedures (and they have thousands of pages of them). However, a competent user can sit down with a piece of well-designed desktop software and figure out how to use much of its functionality. The user may not need procedures. What audience are you writing for?

Do users want to read conceptual topics?

The information technology professionals I mentioned previously are techies. They're curious: They want to look under the hood and see how something works. They want conceptual information. Data entry clerks, on the other hand, whose jobs are task ori-

ented and fast paced, may not. They want procedures. Again, what audience are you writing for?

Are users familiar with the concepts that underlie your software?

I recently looked at some timekeeping software that my employer was evaluating to track the hours that should be billed to different clients. That software had a concept called portfolios, which had something to do with how different clients or categories of work are grouped. I say "something to do with" because the documentation never told me what a portfolio was or gave an example of how one might be used—although it did include a procedure for setting one up. The documentation told me something that I had already figured out—that I clicked the New button to create a new portfolio—but little else.

Are you writing the procedure at the right level?

Are you leading your readers through something mouse click by mouse click when what they really need is a high-level procedure that outlines general steps? One of the most important things to understand about procedure writing is that the audience always brings a certain level of knowledge to the procedure. Write to that level of knowledge.

Are users likely to have problems navigating?

Do users have a hard time finding something in the interface, but know what to do once they get there? If so, an interface map may be as useful as a procedure.

The procedure habit can be tough to break. When part of me thinks that the audience knows every step in a procedure so I don't need to write it, another part of me thinks that the procedure *has* to be there. But sometimes it doesn't. Next time, take a good look at why you're really writing procedure. Are you doing it out of habit or because your audience really needs it? ■

Douglas R. Wieringa is an independent contractor in Seattle. He can be reached at dwieringa@email.msn.com.