Change Your Goal, Extend Your Role

By Pradeep Henry

You developed the manuals and online help for a software product. How did you react when you saw the product's user interface labels (names of menu options, fields, and other objects on the screen)? And how did you respond when you read some of the product's application messages (textual feedback, such as error messages)? Did you think that you could have created them much better?

As a technical communicator, you already have the skills to create good labels and application messages. If these elements are currently being written by programmers or engineers, maybe it's time you stepped up to the plate and took a more active role. That's not to say that programmers or engineers can't write—just that, as a professional technical communicator, you may be able to do the job better.

This requires you to do two things: Change your goal and extend your role. You may meet some resistance as you do this, but the effort is worthwhile. I know from personal experience.

Background

Back in 1989, I was the only technical writing specialist on a software project team. I trained and guided programmers to write manuals, edited their work, and managed the user documentation development effort. Everything went fine until we lab-tested the usability of the product prototype (see sidebar). Our client in Sweden was not happy with the product's online help, which was inconsistent with the printed manuals my team had developed. The solution was to integrate the programmers, who developed help on a part-time basis, into my team so that we could work together more closely.

Just when the quality of help was getting better, our client found problems with the product's user interface labels. The client suggested that I review the labels. I changed them to simpler terms—terms the user was familiar with.

We then delivered a large number of application messages that the product might display. Dissatisfied, our client turned to our software project manager, who turned to me. I reviewed the messages, too—and had to rewrite them in consultation with the programmers.

That project involved a lot of revising—first the online help, then the interface labels, then the application messages. In retrospect, it's logical to ask, Should I have designed these elements myself to start with? That would have saved us a lot of trouble.

We have two important lessons to learn from this project. Lesson one: online information elements (e.g., help), paper-based information elements (e.g., user's guide), labels, and application messages should all be designed together, with improved software usability as the quality goal. Lesson two: Good technical writing skills are needed for the effective design of all these elements. In other words, lesson one is about changing your goal and lesson two is about extending your role.

What Is Software Usability?

The major concerns for the acceptability of a software system are its capability, usability, reliability, performance, and supportability. "Usability" describes a system that is easy, quick, and pleasant for users to work with. Today, software usability is a critical quality factor because it not only benefits users but also brings business benefits to software houses and user organizations that invest in it.

Enhancing Software Usability: Your Ultimate Goal

There are two reasons to change your goal:

1. *Standard approaches are inadequate.*

Technical writers have used three main approaches to develop software user documentation: the product-centered approach, the writer-centered approach, and the document-centered approach. The problems with these approaches can be traced to inappropriate goals.

In the *product-centered* approach, the technical writer's goal is to "document the software," so writers try to write everything about the software. The good thing about this approach is that all the information that users need is available. The bad thing is that it's presented along with a lot of information that users don't need—or want.

In the *writer-centered* approach, the goal is to "write well," so the focus is on elements such as grammar, rhetoric, style, and typography, and on activities such as word processing, copyediting, printing, and binding. The resulting manual is strong in these areas. For software users, however, such "well-written and -produced" manuals can be about as useful as advertisements that win awards but don't sell products.

In the *document-centered* approach common today, the goal is to "write usable manuals," so writers treat information elements as separate entities that deserve their own independent set of methods for development, evaluation, and so on. Thus, among other things, writers organize usability tests for testing information alone. This approach improves the usability of individual information elements. Software users, however, are likely to find information elements developed with this approach inconsistent with the rest of the product. For example, online help may have its own user interface, which is different from that of the product.

2. *Information is a software usability component.*

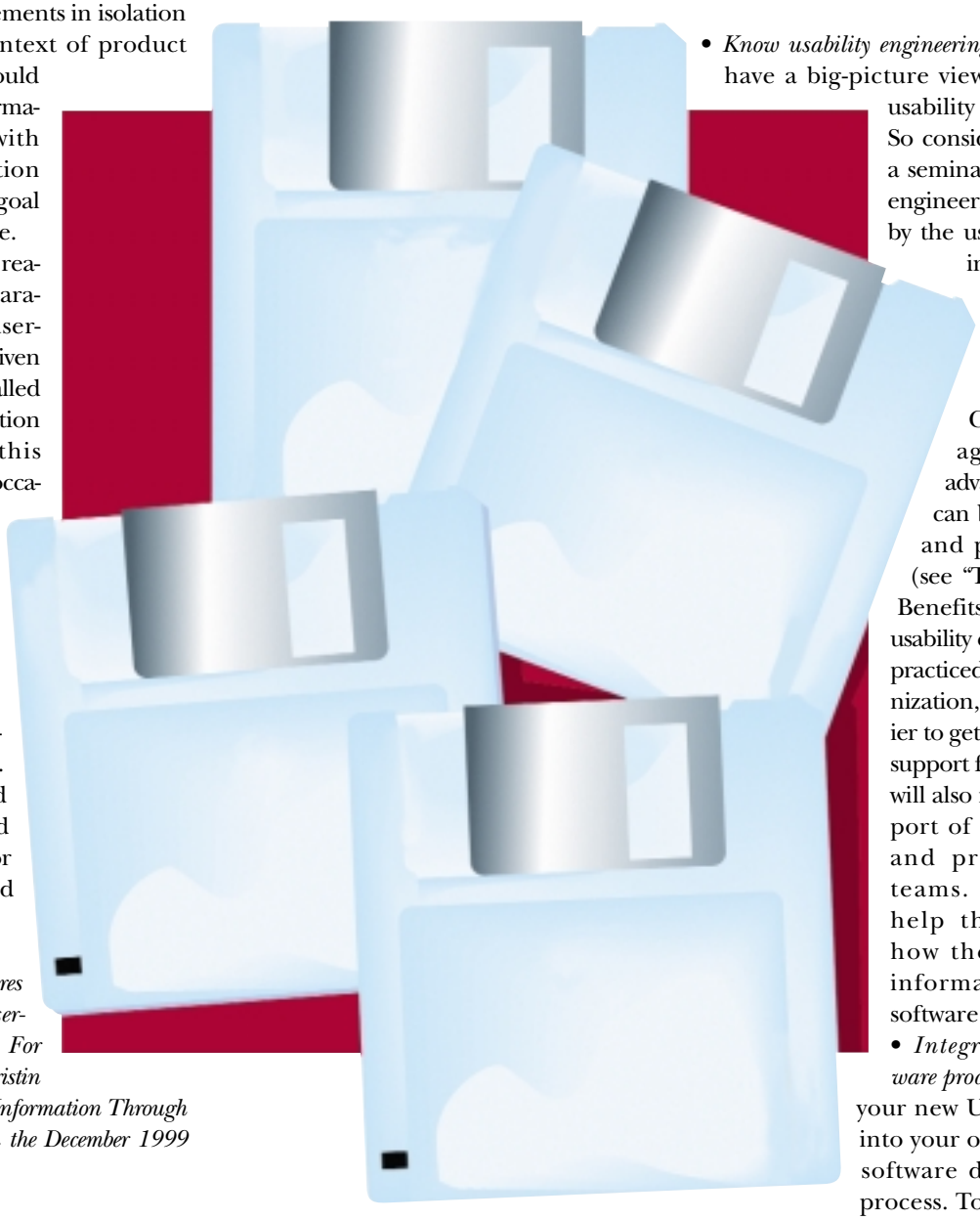
Well-designed menus and navigation are not the only elements that make it easy for users to interact with

software. Various textual elements—such as user interface labels, application messages, online information, and printed information—also make the interaction easier. Simply stated, information is a component of software usability.

Software technical writing, therefore, comes under the heading of software usability. This means that you should not design information elements in isolation from the broader context of product usability. Your goal should be to make every information element work with every other information element to meet the goal of easier-to-use software.

In light of these two reasons, the following paragraphs describe a "user-centered" approach driven by product usability. Called user-centered information design (UCID), this approach enjoyed my occasional and informal patronage for about ten years. In January 1999, I made UCID formal by launching a usability-engineering lab that brings the technical writing function under its umbrella. Experience has proved that the integrated approach works well for the benefit of users and businesses.

(Editor's note: UCID shares some properties with User-Centered Design (UCD). For more on this subject, see Kristin J. Zibell's article "Usable Information Through User-Centered Design" in the December 1999 issue of Intercom*)*



UCID and Your New Role

UCID is an integrated approach to designing all information elements so that they contribute to improved software usability. Integrated design is key to UCID. It means packing into each element just the information users will need or expect from that element, so that all the elements together contribute to the product's ease of use. Integration decisions are made based on knowledge of users, their tasks, and the design of the user interface.

Obviously, you—the technical writer—are the primary contributor to UCID, enjoying an expanded role that includes the design of labels and application messages. In the UCID approach, you are a "designer of product usability" as you collaborate with usability engineers and programmers to improve software usability. If your organization recognizes the importance of good user interface design, you are almost ready to seize your new role.

Preparing for Your New Role

Good technical writers write information well because they see things from the users' viewpoint, have good written communication skills, and understand technical and domain area details. You will use the same skill set to better design labels and application messages. To get started, however, you need to do a few things.

- *Know usability engineering.* It helps to have a big-picture view of software usability engineering. So consider attending a seminar on usability engineering organized by the usability group in your organization or by an external institution.
- *Get support.* Clarify to management the advantages UCID can bring to users and practitioners (see "The Resulting Benefits" section). If usability engineering is practiced in your organization, it will be easier to get management support for UCID. You will also need the support of the usability and programming teams. Be sure to help them realize how the quality of information affects software usability.
- *Integrate into software process.* Integrate your new UCID process into your organization's software development process. To do this, you need to work with the usability and quality (or process) heads in your organization.
- *Prepare to collaborate.* Set up a working relationship among technical writing, programming, and usability teams. Let everyone know what each is expected to do.

Designing Labels

Labels should communicate meaning, content, action, or the state of the object they identify. Unfortunately, poor labels abound in the software world. Some are vague; they leave users uncertain about what to do next. Some are misleading; they promise one thing, but do something else.

Good labels are crucial. Effective labels can lead users—correctly—through the steps required to complete tasks. Good labels can also mean shorter manuals.

In typical software development projects, labels are first created by analysts and programmers. Used in the early requirements specification stages, these labels soon get into the software system. Some of them communicate effectively with users; many communicate only with other members of the software project team.

Spend the time to make labels effective. Your extra effort on labels could save many person-weeks of writing explanatory manuals and help. More important, it could save users many hours spent attempting to understand the interface.

Labels with the following characteristics can help improve software usability:

- Clear
- Distinctive
- Short
- Consistent

To design labels with these characteristics, you need to be creative and may have to do some research. Also, you should work closely with analysts, programmers, and usability engineers.

Developing Application Messages

An application message is textual feedback from the software system. Users need such feedback primarily in three situations: when they need to know what has happened, when they need to be warned of a possible undesirable situation, and when they have just made an error requiring their action. Like labels, many messages are poorly written. Some are cryptic, such as "TRAP0002." Such messages force users to search for helpful information elsewhere.

Changing your goal and extending your role means benefits for you as well as for the users.

Good application messages are important because they make two-way communication possible in a user-software interaction. Communication significantly affects user performance and satisfaction with the software system. Good messages reduce the number of problems users have and make it easy for them to correct problems that do arise. The result is reduced user support costs.

Traditionally, designing application messages has been a programmer's job. According to a study quoted in the book *Online Help: Design and Evaluation* (written by Thomas M. Duffy et al, published by Ablex Publishing Company, Norwood N.J., 1992), only 22 percent of experienced technical writers indicated that they were responsible for writing application messages. Perhaps the most common problem with this approach is that the messages are written from the programmer's viewpoint, which makes them unclear and too technical for users.

Application messages with the following characteristics can help improve software usability:

- Complete
- Conceived from user's viewpoint
- Context-specific
- Clear
- Correctly toned
- Attention-getting
- Readable

As you can see from this list, there are two aspects to designing application messages: the design of message content and the design of message display.

Design the application message content in consultation with usability engineers and programmers. First, identify message categories. You are likely to have the three standard categories: informative messages, warning messages, and error messages. Decide what types of information will go into each category. For example, will error messages only mention the error, or will they give the corrective action as well? For writing the content, coauthor with programmers; you cannot do it well on your own.

Design the message display before you begin the development of a substantial number of messages. How will messages be displayed on the users' screen? Will error messages appear in a pop-up window or at the bottom of the screen? Will there be an accompanying audible feedback too?

The Resulting Benefits

Changing your goal and extending your role means benefits for you as well as for the users.

For you, it establishes your identity as designer of product usability. It puts you on the track toward maximizing software usability—a worthy goal. By providing information that meets users' needs and expectations, you ensure easier-to-use products, which leads to a better bottom line for your organization.

For users, the change helps maximize ease of use. Users' productivity and satisfaction with the product improves. In the traditional approach, different groups design labels, application messages, online user documentation, and printed user documentation at different times, often from different locations. The problems with this approach are missing information, repetition, useless information, and inconsistency. The integrated approach, by eliminating these problems, lets users perceive the software system as a single usable system.

Pradeep Henry is the author of User-Centered Information Design for Improved Software Usability *(Artech House Publishers 1998). After a decade-long technical communication career, he moved into usability engineering and subsequently set up a Usability Engineering Lab at Cognizant Technology Solutions, a former Dun & Bradstreet company. He has been profiled as "Distinguished Technical Communicator" in* IEEE Professional Communication Society Newsletter *(March/April 1994). He can be reached at* ***henry@tech-bridge.com***.