PREFACE

This is a book for people who want to make a lot of money, and the reason I say this is that there can be no doubt that people who can deliver computer documentation that enables users to find what they want in less than 25 seconds at least 80% of the time, are going to be rewarded well. As they should be, considering the market value that this quality alone will add to their companies' products.

This book describes a simple, straightforward method for making software *usable* — not necessarily always *pleasant*-to-use, but such that every user in the class of intended users can find step-by-step instructions for performing any task made possible by the system without having to *figure out* where the instructions are. That is a giant step forward, as I think you will agree if you recall what you typically go through to learn how to use software, in particular software which is not in the computer culture in which you have been working.

The method described in this book enables computer-human interface designers, for the first time, to *guarantee their work*, in the sense of guaranteeing that every user, after his or her first use of the system, will be able to find the information he or she wants within a specified time limit. I am using the conservative figure of 25 seconds as this time limit in the case of paper implementations. The on-line limit is a function of the speed of the software used to implement the method. The term "zero-search-time documentation" is intended to emphasize that with this method, the user always knows *where* to find the information that he or she needs, and *how* to retrieve it. The fact that the time limit is not actually zero is simply a reflection of the fact that we cannot turn pages instantaneously, and that we cannot point and click with a mouse instantaneously, and that software does not execute instantaneously.

Two warnings: First, you should not expect to find much discussion of existing software that can be used to implement the method, e.g., various on-line documentation programs. The emphasis of this book is on the *What*, not on the various *How*s, which in any case change almost daily. The distinction is important. Ever since the dawn of the computer age, the thinking of the average computer professional has been technology-oriented, i.e., centered on the *How*. In fact, one of the requirements for acceptance in any of the computer subcultures — for example, those centered around machines (PCs, Macs), or programming languages (C, C++) — has been knowledge of the current technology. In general the rule has been, *solve today's problems with tomorrow's technology*. Among engineers and programmers, there has been little questioning of the idea that the more features a product has, the better; the faster a machine goes, the better — the idea that all we need to do is improve the technology and everything else will take care of itself.

There are certain parallels to this way of thinking in the first few centuries of the revival of learning in the Western world during the late Middle Ages. The "new

technology" then was the ancient Greek and Roman classics, and the mark of an educated man or woman was knowledge of the Greek and Roman languages. (I specifically include women here because, at least in the aristocracy of Elizabethan England, there were women educated in the classics). In essence, the more words a person knew, the better — the more advanced he or she was. It is fair to say that, until the twentieth century, a good prose style was what we would now call elaborate, flowery, making use of big or obscure words. Now, because of the democratization of knowledge, a simplification of language has occurred: if you want to be successful as a writer, you write for Everyman and Everywoman.

Software design is now on the verge of the same stage of development. The notion that computers are for use only by the select few — namely, engineers and programmers — has been obsolete since the introduction of the Macintosh, at the very latest, although, as you will realize after you read this book if you don't know it already, this notion is not dead yet. But I think we in the computer-human interface design field are now mature enough to step back from our habit of thinking in terms of technology and to start thinking more abstractly, more in terms of function, of *task*, and of certain key parameters of these.

The second warning is that this book is critical of certain attitudes and practices of the technical writing and documentation community. One reviewer remarked that the book "will offend most of the leaders of our field." I could only think, "Then God help our field!" Of course, no profession can hope to remain healthy without being open to criticism, including the kind of criticism that no one in the profession wants to hear. On the other hand, criticism without a criterion for measuring the success of what is being proposed is empty, and that is why the 25-second rule is so important: it forces counterarguments into two distinct camps: (1) that the rule is not as important as I claim, in which case the burden is on the counterarguer to show that documentation systems built with some other criterion of success, or none at all, will have equal or greater appeal to users; or (2) that there exist better ways than the Environment concept described in this book to meet the criterion.

In any case, it will take only one successful implementation of the ideas in this book to convince the skeptics, because that company will be selling so many of their product that the technical writers will have to work in Shipping on the weekends. And this *even if* the software doesn't run as fast as the competition's, since it will be obvious that the throughput, the productivity of the typical user, will be so much higher than the competition's. As my mentor in computer science used to say, "You do it your way and I'll do it mine, then I'll race you around the block!"

An implementation of at least some of the ideas in this book is under development at the time of writing.

When you finish this book, I think you will agree that usability is a solved problem — *if* you define the term as it should be defined, namely, not in the hand-waving, psychological terms of pleasantness-of-use, but in terms of whether or not users can find how to do what they want to do in some minimum time, like 25 seconds. That we can deliver today.

I would like to thank Tiffany Yates for her thorough editing of the first draft of this book, and Chris MacIntosh, who may be the best production editor in the business, for her patient attention to the thousand details that are a necessary part of the publication of any book.

— Peter Schorer,
Berkeley, California, 1995