March, 2001

Editor's View

New Thoughts for a New Millennium

A variety of books offer new ways of thinking about old things

A few years ago, I bought my father a coffee mug that says "So many books, so little time." It clearly expressed my attitude toward life. But I'm also a busy person and as a result, I'm very jealous of the time I have available for pleasure reading.

Generally, I try to find time during working hours for technical reading. But, every now and then, a computer-related book is appealing enough for me to devote some of my private hours to it. Over the last few months, three such books managed to work their way onto my reading list.

Craftsmen or Professionals?

I've written in this space (most recently in the March 2000 issue) about Microsoft's certification program. Steve McConnell's *After the Gold Rush* (Microsoft Press) argues that vendor certification isn't enough. He proposes a true profession of software engineering that would involve certification and/or licensing much like that required for engineers and other professionals.

I was fairly skeptical when I started reading, but McConnell's arguments are cogent and by the end of the book, he had me pretty well convinced. The analogies with both other kinds of engineering and other professions make a lot of sense to me. Much of the software being developed today is mission-critical and even life-critical. We're all aware of the shortcomings in today's software development process. A movement toward professional development standards along with ways of distinguishing amateurs from professionals and beginners from experts cannot help but improve both the results and the public's respect for the field.

But How Does it Work?

When someone asks you to explain how a computer works, I suspect that you, like me, immediately divide the discussion into hardware and software. Charles Petzold's book, *Code* (Microsoft Press), provides a new way of thinking about this issue.

Code opens by considering two children signaling each other with flashlights and proceeds through electrical circuits, integrated circuits, operating systems all the way to programming languages to show how codes are the common thread in all aspects of computing. The book is filled with interesting historical information and detours into a variety of more or less related topics.

I've studied many of the subjects in this book at one time or another, but much of it was relegated to the dusty corners of my brain. Petzold's well-grounded explanations and diagrams provided a great refresher course, as well as clarifying a few things that never quite sunk in the first time around.

This is a book I recommend not only for you, but to hand to those clients (and family members) who always want to know more about what's going on under the hood.

Interfaces Again

No matter how you feel about the results, the recent US presidential election clearly demonstrated the importance of the user interface. Many of the problems with the election could have been avoided if the users (voters) had been provided with a clear, unambiguous interface for voting (and one that didn't require them to read the directions, since, as we all know, users don't read manuals).

I've been interested in user interfaces for a long time. (In fact, my graduate work was in this area.) Over the years, I've read a variety of books on the subject. Almost invariably, I find myself of two minds as I read them – generally agreeing about the diagnosis of the problem, but often disagreeing with the proposed solutions. Jef Raskin's *The Humane Interface* (Addison-Wesley) is no exception.

Raskin, who was the creator of the Macintosh project at Apple, provides not only anecdotal descriptions of problems with today's user interfaces, but includes formulas for measuring the effectiveness of interface techniques. You can use them to compute both the estimated time for a given operation and the efficiency of the operation.

The solutions he offers are revolutionary, and this is where I found myself shaking my head and talking back. Among his suggestions are doing away with individual applications (and the operating system), as well as unifying the operations available. He also recommends eliminating file names and hierarchical directories, offering full text search instead. One suggestion I found particularly hard to understand

is elimination of the Caps Lock key, on the grounds that it creates a mode. (In general, Raskin considers modes a bad idea, and on the whole, I agree with him there.)

I know that part of my resistance to his ideas is habit. I've been doing things more or less the same way for a long time, so I know how to make them work. I'm also aware that I am what Alan Cooper calls "homo logicus," which makes me different from most of the people ("homo sapiens") who use the software I write. The combination makes me suspicious of the benefits of Raskin's radical ideas.

But whether I think Raskin is right or not, just reading his suggestions helped open my eyes to the fact that the way user interfaces look now is not necessarily the way they have to look down the road. Whether we end up following his ideas or doing something totally, it's like that five or ten years from now, user interfaces will be quite different from today's.

The Bottom Line

At first glance, these three books have in common only that they're about computers. In fact, though, what really ties them together is that they all get "outside the box" and make us think about topics in a different light. Having my mind opened is definitely worth some of my precious reading time.