A User-Driven Approach to Better User Manuals

John Maynard

"Damn you! Damn you! Damn you! I spent $100 on your user manuals and I can't understand a thing they say!" So said an irate university student recently in a letter to a mainframe computer manufacturer. Such criticisms, while perhaps not often so vehement, are not at all uncommon. And yet, we can produce good manuals—if we carefully analyze the problems users are having. I propose three strategies: (1) reorganize the manual content and format; (2) create a new role for manual writers and the company organizations supporting them; and (3) speed up manual production and distribution. Each strategy solves a specific set of user problems.

How we attacked the problem.

The problems discussed here represent a consensus of the views of many hundreds of users surveyed during a 15-year period by Control Data Corporation, Scientific Data Systems, and Xerox Data Systems. The users' responses suggested many of the solutions as well. These surveys were among several approaches our research took.

One approach consisted of response forms placed in the back of each manual. Using these responses, we first established the problems users were having with specific manuals. We then determined the problems users were having with whole classes of manuals, such as reference manuals and user guides. Finally, we examined the problems in light of different classes of users.

A second approach consisted of surveying the members of user groups. These surveys, which involved a smaller number of users, were particularly valuable because we used them to monitor overall user reaction as we made various changes.

Another approach involved person-to-person contact with users. Although we made a few trips to user sites, our participation in documentation committees of user organizations was the most productive form of contact. It provided us with continuing user guidance, and it allowed us to modify the surveys of user group members to better measure their reaction to our changes.

Finally, we looked at other manufacturers' manuals and at commercially available publications. For the most part this yielded little useful information, simply because the same problems with manuals prevail throughout the industry.

Organize for learning and retrieval

One of the most frequent criticisms we encountered was that the manuals appear to be written by and for development programmers. Since many user manuals are not significantly different from software specifications (which are written by development programmers), this criticism is not surprising. However, such complaints are symptomatic of a fundamental problem—we have failed to write and organize manuals according to what users need. Manual writers have instead based their work on specifications.

Users repeatedly asked for function-oriented rather than software-oriented manuals. Software-oriented manuals are the norm today; each manual is organized around a software structure—file manager, linker, loader, job control language. Function-oriented manuals, however, are organized around user job functions—logging on, building files, modifying files, printing files, logging off.

Our users also desired reference manuals containing fewer words. Most manufacturers have supplied many reference manuals but few training manuals. To meet the frequent requests for more tutorial information, writers have merely added wordy examples, thereby decreasing their effectiveness for quick reference. Users clearly believe that reference manuals should be concise and organized for quick retrieval of information.

Some users want manuals that are less wordy; others want manuals with more explanations.

Users frequently requested manuals with more explanation of how to use a particular software product. Although this conflicts with the demand for less wordiness, it shows that different users need different information. Experienced programmers need quick access to information such as syntax rules. Novices need tutorial information in addition to quick access. The apparent conflict indicates that the user manual has both a learning and a retrieval function.

Although this brief analysis of user needs is not exhaustive, it gives us enough information to start reorganizing
manuals. First, manuals should be aimed at users according to their function, e.g., site administration, system programming, application programming, computer operation, etc.), so they can quickly find the information they need. Second, every manual should contain two separate parts—a learning section and a retrieval section.

Depending on the audience level, the learning—or "how to use"—section could range from a brief tutorial covering just a few aspects of a larger function (e.g., a short tutorial on computer operation), to a full-blown training text on the entire function (e.g., a complete introduction to all aspects of computer operation).

Although the tutorial material can be presented in many different ways, I believe the most effective approach is one that divides it into chapters based on degree of complexity and frequency of use. The first chapter would then contain the simplest, most frequently used information, the next more complex and less frequently used information, and so on. Each chapter would be self-contained so the user can apply the information without consulting other parts of the manual. Every chapter should itself be arranged so that the material flows from simple to complex and, as much as possible, follows a sequence similar to the way the systems described will be used. Brief programming exercises, presented so that the user interacts with the material every few pages, would also reinforce the "how to use" approach.

This type of tutorial has proven effective. A book published by Wiley has some of its elements. This book, Fortran Autotester, by Robert E. Smith and Dora E. Johnson, was first published in 1962 and has sold over 40,000 copies. It is still in print and is one of Wiley's most successful programming books. Its success is due to its superior self-instructional style. Bringing this tutorial style to manuals should enable us to satisfy users' needs.

The retrieval part of the manual, called a reference digest, would completely define the user interface for the systems described, but would do so in a tabular format for rapid retrieval. It would be presented as much as possible in text-free form, i.e., with syntax statements and abbreviated descriptions of error messages. Similar types of information, such as error messages or functional groupings of commands, would be grouped together.

Writers should be programmers

The traditional role the writer plays in software development hinders the writing of good user manuals. The typical writing scenario in most publication shops is as follows: Development programmers prepare descriptions (specifications) of the software during the design phase of development and turn these descriptions over to programming writers, who research and write the first draft of the manual. The extent of the writing varies all the way from literary editing to extensive original writing. The first draft of the manual is then reviewed by various groups throughout the organization, such as development programming, marketing, training, and hardware engineering. The writer then incorporates the review comments and turns the draft over to production for final typing.

The traditional role of the manual writer ignores user experience.

Various other roles have been tried with little success. Some people believe writers should work alongside development programmers so as to become more knowledgeable about the software product. The trouble with this argument is that it is user programming knowledge, not development programming knowledge, that the writer needs.

Another suggested role for writers is as trainer. Proponents of this approach argue that if writers must teach from the manuals they write, the manuals will be better. Although I have more sympathy for this approach than the preceding one, it too has its drawbacks. One is that this role makes conflicting demands on writers' time—he must teach and write. Moreover, my proposed type of manual may eliminate the need altogether for formal training organizations in companies.

The major difficulty with all of these roles is that they ignore user experience. For example, what problems arise when particular commands are used together? How does the user recover from a specific error condition? The lack of this type of information forces users to discover it for themselves—a time-consuming task, as our surveys showed.

The new role. The scenario for the role I propose is as follows: The development programmers write the functional descriptions as before. The writers, however, do not start researching and writing upon receipt of these descriptions. Instead, they become the first users of the software product being developed. In effect, they become the first system analysts, site administrators, system programmers, application programmers, and computer operators. As soon as they gain sufficient knowledge about the simplest aspects of these functions, they write a tutorial guide and reference digest for them. Then they go on to the more complex and less frequently used aspects. This procedure continues until all the functions are entirely documented. The review and revision procedure is then the same as before.

The primary advantage of this new role is that the writers themselves will use the product and incorporate their experience into the manual.

In the past, it was difficult to implement this approach because the software was not operational soon enough. With top-down software development methodology, however, writers can start using the software early enough to make the approach practical.

Selection and training. The selection and training of writers is critical to the improvement of user manuals. In the early days of the computer industry, companies tried to find writers with "a flair for words." We soon realized that such writers—unless they also knew something about programming—tended to produce polished but useless manuals. Unable to determine what should be suppressed and what was missing, they merely edited the specifications they were given.

Today the pendulum seems to be swinging toward programming expertise. We have at last begun to realize that thorough technical knowledge of software products is one of the most important criteria for writer selection. To be sure, the change is not coming easily. The industry, perhaps in an effort to hold down costs, has been slow to set up writing classifications with salary ranges comparable to programmers. Yet every time a writer adds needed information to a manual or structures a manual based on actual use, he saves time for everyone who uses the product.

Of course, attracting the right kind of people is not the whole answer. We need to put the programmer-writer in a new role, combining communication
skills, technical expertise, and user experience. Only then will we be able to produce significantly improved manuals. User comments supported this analysis.

Elevate the programming publications group. The ultimate success of programmer-writers, even in their new role, also depends on the organization of the programming publications group. Because managers seldom appreciate the influence of documentation on productivity, they frequently set the publications group up as some low-level support group within software engineering. As a result, the publications manager has little influence on user manual development.

If we wish to improve user manuals, the publications group manager must be elevated to the same level as the managers of the software and hardware engineering groups. At least one major mainframe manufacturer already takes this approach. This organization puts the publications manager in a position to influence manual development, it gives him the power to take a non-traditional approach.

Make production and distribution timely

Users have been plagued for years by the lack of timely distribution of manuals. The problem is twofold: manuals are frequently out of stock and manual orders are often filled far too slowly.

Why ship tons of paper when we can distribute manuals on tape?

The problem of stocking manuals that are frequently revised is a difficult one. Even with good inventory control, there is always a tradeoff between the user's need to get manuals quickly and the supplier's need to avoid overstocking manuals that are quickly made obsolete by new editions. In addition, suppliers—particularly computer manufacturers—have not responded well to customer orders. Some suppliers believe that every problem related to an order must be solved before the order is shipped. As a result, it sits in a queue for several weeks.

Aside from these problems, it simply does not make sense to print and ship thousands of tons of manuals a year.

Instead, we should ship user manuals on tapes, the way we ship software today. Users will then be able to modify manuals to meet their own needs and to select the format and quality of their printed copies. For example, users could have multilith masters typed out on a Diablo or Qume terminal and then print inexpensive distribution copies on a multilith press. Another option might be to produce the distribution copies on a printer/plotter or to use the printer/plotter output as offset masters.

Recent advances make distribution on tape even more attractive. Previously, only a partial automation of manual production—mainly text entry—has been feasible. Most graphic material had to be added by hand. Fortunately, several low-cost devices introduced in the last few years have made complete automation of manual production possible. Printing/plotting terminals and computerized photocomposition devices can now intermix text and graphics. We can now convert tape files to camera-ready copy with no manual processing.

Automated production and distribution on magnetic media, then, can alleviate problems associated with stocking and ordering. Suppliers need not worry about large stocks of out-of-date manuals or handling large volumes of expensive printed materials; users can easily modify manuals (in the form of tape files) and print copies inexpensively for local use.

Can we afford to do a good job?

Can computer companies afford to improve user manuals? Although most have certainly had a difficult time trying to remain profitable, I believe they cannot afford not to.

A fundamental economic fact of the computer industry is that communication has a dramatic effect on productivity. Yet budget cuts nearly always reduce the size of publication organizations. The irony is that we in fact decrease our productivity during a budget crunch!

We fail to recognize the hidden cost of poor communication because we do not attempt to measure the productivity of professionals. Most managers acknowledge the problem but appear unwilling to address it. One simple way to measure this productivity loss is to ask development programmers and customer engineers how much time poor manuals cost them.

Even if we ignore the potential savings to in-house users, my approach will still cost the supplier no more than the traditional method. This may seem improbable since I propose substituting highly-paid programmers for technical writers, but let me offer a reason why this may be so.

No one seems to pay attention to the cost of the time lost because of poor manuals.

A common estimate of programmer output is five to ten debugged instructions per day. A comparable estimate for technical writers is about two 500-word camera-ready pages per day. This includes incorporating revisions, reviews and proofing as well as writing. I am, of course, talking about writers who write the traditional style user manual, not those who merely edit it. It is my experience, however, that only two hours per day are spent physically writing, revising, and proofing copy. So what is the writer doing the other six hours? Learning! Either by trying to understand the specifications or by asking the development programmers questions.

In experiments with the new approach, I found to my surprise that my output nearly doubled. I had always believed that the new way would take longer. But after analyzing the experiments I now believe that I produced more because I learned faster—and I learned faster because I functioned as a user. In addition, I could then organize my experience around a proven framework, the tutorial text/reference digest arrangement. So the user role and new style of manual promoted productivity. This should outweigh the cost of the larger salaries.

Suppliers normally set manual prices to recover production and distribution costs. Since writing costs are grouped with software development costs, the higher writer salaries will not in any case impact manual prices. I expect that under the new approach suppliers will set prices equal to the cost of producing and distributing manuals on tape, just as they now set prices to recover the costs associated with paper copies.

Up to now I have discussed only supplier costs. And yet, inadequate manuals cost users far more than they do suppliers. I would not be surprised to find that more than half of users' programming costs are caused by inadequate manuals.
How we did it at Xerox

While I was at Xerox Data Systems, we implemented several of the ideas after receiving some very uncomplimentary feedback from our users. In a survey taken early in 1971, 93 percent of 60 user group members thought our manuals were inadequate.

In response, XDS management first made the publications manager a permanent member of the Documentation Committee of the Exchange Users Group. This gave us the user guidance we needed to upgrade the manuals. Next, we were given a larger budget for improving specific manuals and for hiring programmers as writers.

Our first major effort was to accelerate the reorganization of operating system manuals. We had already started arranging one such manual by function (e.g., system management, batch processing, timesharing). We continued this reorganization and began work on the manuals for two other operating systems.

As our next major effort, we increased the number of tutorial-type users' guides. These were not combined with the reference digest as I have proposed here. But they were a step in that direction, and the combined tutorial/reference digest manual would have been the next step, had Xerox stayed in the mainframe business.

Early in 1972, with only partial implementation of our program, another survey showed that users felt we were making progress. Of the 242 users responding, 52 percent said the manuals had improved, 45 percent felt they had stayed the same, and 3 percent said they had deteriorated. The survey also asked users about the adequacy of the manuals. Only 6 percent complained that they were totally inadequate, while 22 percent said they were somewhat inadequate. Thirty-nine percent said they were adequate, 15 percent good, and 14 percent had no opinion. Four percent had no response.

Six months later, with 74 users responding, we obtained similar results. This time the survey simply asked the users to rate the manuals on a scale of 5, where 1 was excellent. The results were as follows (Note that the percentages shown here and for the next survey are composites—the surveys showed results for specific manuals.):

1 11 percent
2 40 percent
3 31 percent
4 15 percent
5 3 percent

By the time of our 1974 survey, two-thirds of our writers were experienced programmers, system analysts, programming instructors, or operators. They had experience in Fortran, Cobol, Jovial, assembly language, and other languages. We had also partially reorganized most of the manuals for two additional operating systems. The results of this survey, again with 74 responding, were even more encouraging. Sixty-seven percent checked that the manuals were "okay as is," 13 percent marked "needs to be rewritten," 1 percent indicated "needs less detail," and 19 percent chose "needs more detail." Here we tried to make the question less emotional by eliminating words such as "excellent" or "inadequate." As I expected, our programmers were doing a good job of sifting out extraneous material such as implementation information, as reflected in the 1 percent response to "needs less detail." The 19 percent response to "needs more detail" and the 13-percent response to "needs to be rewritten" indicated the users' need for more experience-derived and more tutorial information. Specific comments from the survey and from response forms in the manuals appeared to support this. Although the 1974 results were dramatically better than the 1971 results, they nonetheless indicated we still had much to do. Although we hired programmers, we did not alter the traditional role of the writer. What few user guides we supplied were in addition to the traditional reference manuals. I believe that adoption of the tutorial/reference digest manual and of the role of writer as user would have overcome the remaining shortcomings.

Can we improve user manuals? I believe we can—a dramatically. What is needed is clear—an awareness of users' needs by manual publication groups.

The approach I have proposed, which arose out of that kind of awareness, can accomplish this.

John Maynard is an analyst in the Tools and Techniques Section of Software Engineering at Honeywell's Large Information Systems Division in Phoenix. Prior to joining Honeywell, he was manager of Programming Publications and of the Software Library at Xerox Data Systems. His other experience includes work at Bendix, Packard Bell, Control Data, and Scientific Data Systems.

Maynard received a BBA from Golden Gate University, San Francisco, and an MA in economics from UCLA. He also holds certificates in data processing and writing.

SR. SOFTWARE SCIENTIST

We're seeking a recognized authority in the field of software design—a senior technical professional who can lead the design and development of advanced software tools for a world leading company in telecommunications. PhD supported by 10 years extensive applicable experience is the minimal requirement. Salary $50-70,000. Write in strict confidence to Albert E. Powell.

ABBINGTON ASSOCIATES
P.O. Box 826
Huntington, NY 11743.

An equal opportunity employer m/f.

January 1979