1 Online Help in Context


Arnold Penzias, in his book *Ideas and Information: Managing in a High-Tech World* (1989) has accurately assessed the pervasive influence and the importance of technology in our society.

Throughout the ages, technology has helped shape the facts we humans think about. As our knowledge has increased, so have our tools and the ways we employ them. Today, technology is so complex and pervasive that it dominates much of the environment in which human beings live and work. For this reason, I feel we need a better understanding of how technology affects the ways in which we now create and explore ideas (p. 180).

And one of the more complex and pervasive tools we have at our disposal today is the computer. But what are the implications of the “computer age” for the designers of computer systems? And what, in particular, is the impact on the status of help systems? To answer these questions, and to begin addressing the topic of this book, we must start with a consideration of how the uses and the users of computer systems are changing.

Computers are becoming an integral part of virtually all aspects of our day-to-day activity. Services are being automated with computers. We see automatic tellers, computerized libraries, automated offices, computerized stock management, computerized entertainment and tourist information services in hotels, and so on. Importantly, all these technical services share one thing in common—in each case an individual operates a computer to obtain a particular service that otherwise would have been performed by a human.

Perhaps an even greater impact has been the use of the computer to complete tasks that we would have ordinarily done ourselves or would have had to hire someone to do for us, that is, personal computing. The first microcomputer, Kenbak I, was introduced in 1971. The first personal computer store did not open until 1975 and companies like Apple, Commodore and Tandy did not introduce their products until about 1977. Since that time we have seen an exponential growth in the sales of microcomputers. Whereas, in 1978, there were approximately a half million computers in use, by 1983 that number had grown 20 times to over 10 million computers. Some researchers, in fact, predict that this rapid growth will continue and they expect over 100 million computers to be in use by 1995 (Juliussen & Juliussen, 1988).

The number of software packages, designed to do virtually anything we can imagine, has also grown exponentially during this period. Not only has the breadth of applications grown, but the number of alternatives available for any one application area as well. The March 1989 issue of PC Magazine, for
example, reviewed 49 statistical packages, 9 scientific graphing packages, and 16 math toolboxes—a total of 74 different programs available for just three application areas on one platform, that is, the IBM-PC (and compatibles). Add to that overwhelming selection, applications for the Macintosh platform. MacWorld in February 1989 reviewed 8 spelling checkers, 6 personal finance programs, 8 investment packages, and 5 tax preparation applications. And we’ve neglected to list the various graphics, word processing, music, payroll, database, spreadsheet, et cetera programs, or to consider alternative platforms such as the array of available operating systems, workstations, minis, and mainframes (see, for example, Frank Catalano’s amusing article regarding our “headlong rush into the operating system Tower of Babel” in the March 14, 1989 issue of MacWEEK, p. 24). The number of potential programs an individual may encounter is truly staggering.

Moreover, the escalation in software packages is effecting, not only end-users, but programmers and system designers as well. Increasing emphasis is currently being directed towards producing cross-platform connections, for example, using HyperCard as a front-end for a database application residing on a Vax 785. In this respect, programmers and system designers are being exposed to an increasing number of platforms, not to mention being forced to decide how best to link or to port different applications across various technologies.

Perhaps more importantly, however, is the proliferation of programming languages over the last 15 years: C, Ada, Lisp, Pascal, Prolog, c++, t, Scheme, and a host of languages specific to particular applications, for example, HyperCard™ (HyperTalk™), database application languages (Lotus 1-2-3™, Excel™ Macros, Dbase III™, etc.), word processor macros (emacs™, Xedit’s Rexx™, etc.), instructional design languages (Natal™, CAS/DAL™, etc.), and fourth generation languages (Zim™, PowerHouse™, etc.). And, as with the applications, variations exist for the majority of these languages. The February 1989 issue of Byte magazine reviewed 12 different C compilers, for example.

We have gone on at length with these examples because we feel that the consequence of this growth, in both the breadth and depth of applications, has tremendous implications for the development of user support systems. We would argue, in fact, that these developments will alter current characterizations of the computer-using populous in the following ways:

**Fewer Experts**

The first implication is that fewer individuals who are considered experts on any one program will be available at any one computing site. In the past, it was probable that everyone at a given location used the same word processor, the same statistical package, and the same programming language simply because so few alternatives were available. However, as the number of alternatives has expanded, and people are able to select applications based on personal preference, it is less likely that two people in the same room will have experience with the same application. In effect, this means that the most popular source of help or assistance is much less available—“Excuse me, but do you know how to...?” is no longer a foolproof problem-solution strategy.

**Fewer users developing expertise**

The second implication is that people, in general, will not develop and maintain expertise on the majority of applications they use—at least not the in-depth, comprehensive knowledge we often associate with “computer expertise.” Certainly, in the early days of computing, conceptions of the computer user differed significantly
from today's conception. That is, historically, the primary users of computers were programmers, engineers, and technicians. They programmed daily and generally used the same language and the same word processor to do it. As one would expect, continuous, frequent use led to expertise. However, many of the programs we use today are used only infrequently and for short periods of time. The sales person might put together a presentation on a monthly basis, or the secretary might use a spreadsheet application only on Fridays, or the accounting department might run its corporate financial system every quarter. In this respect, we would argue that users never develop lasting expertise since their knowledge, goals, and strategies don't necessarily transfer from situation to situation and from application to application.

More transfer users

The third implication is that we are more likely to transfer to different applications in the same application area, for example, from one word processing package to another. It is not unusual for users to choose to upgrade or to advance to newer, more advanced applications than those they're currently using. Nor is it unusual to change jobs or to have to work away from the normal facilities and to find it necessary to use an unfamiliar product. Thus the number of transfer users—people attempting to apply their knowledge of one program to another—is increasing.

More novice users

Finally, the growth (both current and projected) in the number of computers in use suggests that there will be a continued growth in the number of first time computer users.

Indeed, the dramatic changes influencing the characteristics of the user population will have profound implications for software design in general and online help design in particular. That is, software designers have become increasingly aware of how important it is to build applications that are easy to use or “fool-proof.” They recognize that, given the highly aggressive nature of the software industry, it is less likely that features and power alone will sell a product since several competing products will potentially share the same general capabilities. “Usability” is the bottom-line.

The development and success of the Unix operating system provides a technical, but excellent example of the tension between system complexity and system ease of use. At the time that Ken Thompson and Dennis Ritchie began to develop Unix, most operating systems allowed programmers numerous ways of moving, arranging, and storing information. Thompson and Ritchie, however, decided that emphasizing ease of use for the users of operating systems—that is, programmers—was the ultimate goal. In response to this criterion, they set about creating a small but comprehensive list of functions (that is, modules or utilities) for simplifying the programmer’s interactions with the computer system and the result was the highly successful, commercial operating system, Unix (Penzias, 1989, pp. 129-130; Ritchie & Thompson, 1974).

Perhaps a more publicized example will further clarify the recent emphasis on creating “user-friendly” applications. In 1979, Visicalc, the first spreadsheet program, was created by a business student at Harvard and allowed users to transfer their accounting tasks online. Soon after, a flurry of
competing products appeared on the market, for example, Lotus 1-2-3. By 1983, Lotus 1-2-3 held the largest share of the spreadsheet market and, as Shneiderman (1987) points out, part of its success was clearly due to improvements to the user interface (pp. 185-186). Another obvious advantage of the emerging spreadsheet applications of the late 1970s and early 1980s was that they allowed users familiar with traditional accounting tools—pencil, ledger, and calculator—to quickly recognize that similar features were available in the online versions. Clearly, then, complex applications don’t always necessarily gain the popularity of programs that perform fewer functions but are easier to use.

Let us emphasize that we do not believe usability is an important issue simply because an increasing amount of computer users are novices. Usability is also an issue of increasing importance to people who consider themselves “expert” computer users. As we suggested earlier, experienced computer users will still be exposed to—from time to time—a wide variety of applications spanning numerous application areas. While they may have a good general knowledge of their preferred word processing package, experienced users are not likely to require a personal finance application or tax preparation program anymore frequently than novice computer users.

Let us also re-emphasize the number of alternative programs the expert has to use, limiting them to his or her particular area of expertise. Earlier, we noted that there are over 49 statistical packages for the IBM-PC, excluding the Macintosh and countless other technological platforms. Imagine a consultant on statistical analysis trying to work with this variety of programs if they were all as poorly designed and poorly supported as the statistical packages of the early 1960s. Add to the scenario the growth in the availability, variety, and power of peripherals and add-on cards which enhance the functionality of software even more. Clearly, the number and sophistication of potential computing tasks is escalating dramatically. The consultant must deal with more diversity and complexity and will, not surprisingly, vary in levels of expertise depending on which program he or she is using, even in his or her area of expertise.

**Designing Usable Systems**

How do we make computer systems more usable in order to accommodate changing user characteristics? The initial, and still dominant response of the industry, has been to focus on the software interface. The buzz words of the past were “user friendly” and “transparent.” The goal was to make the capabilities of the program, as well as the methods for using these capabilities, fully clear to the user by carefully devising what the user sees on the screen. That is, the user should be able to intuit what can be done and how to do it simply by looking at the screen. As we shall discuss shortly, there has often been a naive expectation with regard to the level of transparency possible; a naiveté that fails to recognize the importance of the user’s knowledge. Nonetheless, the relatively recent focus on the interface has had a very positive impact on usability.

The shift in focus to interface design required a very significant re-orientation in the industry. Computing, in the 1960s, had always taken place in isolated laboratories and been carried out by highly specialized technicians, mathematicians, and so on. Given this setting, it’s not surprising that maximizing the power and capabilities of machines outweighed human-computer interface considerations. Certainly, while power and capabilities are still very important, this new set of goals—interface goals—suggests that engineers and system developers may have to trade off capabilities or functionality for usability. They have to consider how the (internal) code will be presented (externally) to the user and how it will be used by the user (the relationship between internal and external).
The introduction of the Xerox Star user interface in 1981 (Cattell, 1979; Learning Research Group, 1976; Smith, Irby, Kimball, Verplank, & Harslem, 1982) and of the Apple Macintosh computers in 1984 (Juliussen & Juliussen, 1988) placed even greater emphasis on usability. The challenges of designing direct manipulation interfaces and the integration of metaphors in the design of the interfaces (see, for example, Carroll & Thomas, 1982; Hutchins, Hollan, & Norman, 1986; Shneiderman, 1983) brought new software engineering prestige and excitement to the goal of usability.

Today the human-computer interface is a well-established discipline with a major annual conference (CHI or Computer-Human Interaction) held under the auspices of the Association for American Computing Machinery and sponsored by numerous special interest groups including ACM's SIGCHI, SIGCAPH, SIGGRAPH, SIGOIS, the Human Factors Society, the Computer Society of the IEEE, the Cognitive Science Society, the British Human-Computer Interaction Specialists Group, the European Association of Cognitive Ergonomics, and the Software Psychology Society. There are also innumerable guidelines available for the designers of interfaces. Perhaps the most extensive listing is the 679 guidelines presented by Smith and Mosier (1984) covering both hardware and software aspects of the interface. A sample of additional sources that provide a comprehensive consideration of interface design issues include the following:

- Gardiner and Christie's (1987) book presenting a cognitive perspective of interface design and 100 guidelines derived from that perspective.
- Dumas' (1988) book in which guidelines are embedded in a presentation of principles for designing interfaces.

In addition to these "how-to" books, there are also several very important books and articles presenting conceptual or theoretical perspectives on interface design (for example, Breuker & de Greef, 1985; Card, Moran, & Newell, 1983; Norman, 1984; Norman & Draper, 1986; Shneiderman, 1987). These books and articles focus on general considerations for interface design—interfaces for personal computing programs. As well, numerous books and articles can be found for the designers of specialized applications like instructional software (see, for example, Bork, 1983; Bork, Franklin, Von Blum, Katz, & Kurtz, 1983; Hannafin & Peck, 1988; Heines, 1984).

Clearly, interface design is now receiving significant attention from computer industry practitioners and researchers. However, even given this new focus, is a transparent interface an attainable goal? Don Norman, in discussing the ultimate interface, argues that "...the very best computer systems will be ones where you don't know there's a computer" while at the same time noting that computer interfaces will never be able to fully anticipate all user tasks and problems (MacWeek, 21 March 1989, pp. 19-20). Indeed, it's not uncommon to hear designers argue that interfaces should be so "transparent" to the user that documentation is unnecessary, as if to suggest that the existence of documentation reveals a flawed interface design. The interface, some designers believe, should be the only help a user requires. In fact, the argument continues, any plan for documentation may negatively impact the interface design. Once interface designers know that there will be supporting documentation they may be tempted to sacrifice ease of use for power or elegance. The strongest version of this argument, then, is that usability rests with the interface alone.
Is it possible to design an interface where all users can intuit all possible tasks and all of the operations necessary to execute those tasks? We think not. That is, we would argue that how an interface is perceived, how it is interpreted, how it is understood, depends on the knowledge users’ bring to the situation. And that knowledge, we hold, is not exclusively factual knowledge about computing and about the topic area—though variations in knowledge of these facts would be sufficient to make a transparent interface infeasible. In addition to factual knowledge, experience in a knowledge domain and experience with a particular type of technology significantly affects how you view the world: what you notice, how you interpret what you notice, and how you link concepts together in an attempt to represent a situation or a goal. Therefore, even if interfaces are simple and clear—which is always a goal—we may expect users with different knowledge bases to interpret the capabilities and perhaps even the procedures differently (see Norman, 1990, for a more comprehensive discussion of how design must take into consideration the user’s goal and the user’s knowledge—and some grand examples of failed designs).

Our point can best be illustrated by considering an example often used by those who propose that the interface should be so transparent that it needn’t require supporting documentation. They will point to the pencil as a technology with just such a transparent interface. The pencil does a great many wonderful things and we do not require support materials on how to operate it. The pencil’s function is apparent simply by looking at it—or so the argument goes. The goal of these advocates, then, is to design their application’s interface so that it is as transparent as the pencil interface—thus obviating the need for any other usability support.

Let’s take a closer look at the pencil’s “transparent interface.” First, consider a basic pencil and the basic applications of that pencil, analogous to the most basic computer and computer programs. Is the use of the pencil really without instruction and without assistance? We argue emphatically not.

In reality, we have received significant training and significant assistance in learning to use the pencil. And the more sophisticated the pencil, the more help we have received. Children in school learn how to hold a pencil. Even in later grades there may be error correction when they hold it improperly. This includes not only generally holding the pencil, but also holding it at the correct angle and moving it in the right ways to form particular symbols. The training necessary for learning to form those symbols using the pencil technology is significantly different from how we would train the students to make those very same symbols using a typewriter or a keyboard. The training, therefore, is specific to the technology of the pencil. Indeed we would wager that you too can remember the laborious hours in school mastering how to use the pencil to make just the right symbols. Some of you may even have used templates to guide your initial practice.

Moreover, many of us have had to learn the functional difference between #2 and #3 lead pencils—which pencil to use for which task. Others of us have also had to learn that there is an important distinction between #2 hard and #2 soft. Frequently we’ve had to learn that the hard way—by finding the #2 pencil we’ve been using is the wrong #2 and having to interrupt the task (usually a test) we’re doing to search for the proper #2.

Of course, we have also had to learn the difference between pencils and pens (fountain and ball point). This included matching the proper instrument with the marking goal and with the particular surface we wish to mark (for example, consider the problems in using a hard pencil on a glossy surface, writing graffiti on bathroom tiles using a fountain pen, etc.). In the most unpleasant scenario, our learning included recognizing the consequences and error correction procedures available to us when the contents of one of our writing instruments got free.

Thus far we have only been dealing with the everyday use of pencils—alogous to the most basic computer platforms and applications. These are well-learned skills. However, let us now extend the
technology, much as computer technology is being extended. Who would consider the use of calligraphy pens to be intuitive and not require any support beyond the natural interface? The same can be said about the selection and use of pens for graphic art—knowledge of the technology of graphics tools is an integral part of the expertise of the artist.

There can be little doubt that the “transparent” interface of the pencil actually requires a significant amount of training and support in pencil tasks—including the selection of the appropriate tool for the job, the actual use of the tool to accomplish specific tasks, and troubleshooting when the tool is not functioning properly. However, the impact of using the pencil goes well beyond this development of fact knowledge. It is part of the technology of writing, a technology that has allowed us to form grapholects such as standard English (with nearly 2 million words) and to develop fully analytic and abstract modes of thinking (Ong, 1985). Being part of a literate society—being experienced in the technology of writing—has transformed our way of viewing the world.

So too, we may expect computer technology to impact our representation of the world. For example, accountants who once balanced books using ledgers and adding machines can now interact with electronic spreadsheets. As a new tool, the spreadsheet requires the accountants to perform their original tasks in new ways; as the accountant gains experience her understanding of the accounting tasks will undergo transformations (compare the way packages like Lotus 1-2-3 allow us to access our data-sets in multiple ways, including graphs, pie-charts, ledger-form, etc.).

We have explored the technology of the pencil because it nicely illustrates the importance of knowledge in using the technology. Using technology requires a knowledge base, training and, most importantly, it’s continued use has affected our way of thinking. If the use of the pencil and the technology of writing requires this training and has this impact on problem understanding and problem representation, how can we possibly assume that computer technology can ever be transparent. Certainly, computer programs are growing in complexity and, even now, they amplify the complexity found in the technology of graphics and calligraphy (for which extensive training and help is required). We find it difficult to conceive of an interface that is transparent to users with different conceptualizations of the tasks they are attempting to perform. Surely there must be additional help available to link the user, who has a different perspective of the task, to the appropriate perspective exemplified in the interface.

In sum, we see little doubt that users will require help in using a software package of any substance regardless of how much effort designers put into trying to make the interface transparent. This is certainly not to deny the importance of interface design. An easy-to-use interface is obviously a critical goal. We are simply stressing that additional support or help is an equally critical goal.

Helping the User

What basis shall we use for classifying help systems? What are the basic types of help systems? Since the focus of this book is on the design of help systems, our classification system should reflect basic differences in design requirements.

Help systems have typically been classified in terms of the skill of the user or in terms of the types of document being developed (Brockmann, 1986; Hayes, 1982; Houghton, 1984; O’Malley, Smolenskey, Bannon, Conway, Graham, Sokolov, & Monty, 1983; etc.). We will discuss these approaches to classification later. For the moment, let us argue that the most useful classification of help systems—from a rhetorical perspective—is in terms of two dimensions: the delivery medium and the goal of helping. Crossing these two dimensions results in a matrix containing six cells, each representing a “different way of helping” (See Figure 1). Each of these cells may be further analyzed into sub-systems.
requiring different design strategies and this more detailed view will be our goal when we focus on the cell labelled **Online Help**.

**Dimension 1: The Medium of Delivery**

Assistance to the user may be provided through a variety of media. Most commonly, the contrast has been between delivering information online or in hardcopy form, but examples exist in both video and audio tape. With the growth of CD-ROM capabilities we may expect that video and audio delivery to also be “online” in the near future. For now, however, we will not consider the audio and video issues, but will restrict ourselves to the more traditional and more readily available text and still-graphic online information.

There are two reasons for considering the delivery medium as a critical dimension for discriminating between help systems. First, our research has indicated that there is a considerable difference between online and hardcopy in respect to information design. We will discuss this research in some detail in Chapter 2. In many respects, though, content and design decisions that are affected by the delivery medium are a basic consideration throughout this book. Some of the fundamental differences between the two mediums in terms of design include the following:

<table>
<thead>
<tr>
<th>User’s Goal</th>
<th><strong>Medium of Delivery</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Hardcopy</strong></td>
</tr>
<tr>
<td>I want to <strong>buy it</strong></td>
<td>A sales Brochure. A fact sheet.</td>
</tr>
<tr>
<td>I want to <strong>learn it</strong></td>
<td>A tutorial manual.</td>
</tr>
<tr>
<td>I want to <strong>use it</strong></td>
<td>A user’s manual.</td>
</tr>
</tbody>
</table>

Figure 1: A Classification of Different Types of “Help Systems.”

- the lack of permanence of the online display;
- the restrictions in screen size of the online display;
- the inability to interact with the user and to provide dynamic, animated displays in hardcopy;
• the restrictions in cross referencing or multiple presentations of information in hardcopy;
• the different requirements for navigation aids in hardcopy and online, for example, menu design
  and search mechanisms.

The second reason for identifying the delivery medium as an important dimension in classifying help
is because online help offers numerous advantages over hardcopy delivery, and we expect the
importance of those advantages to increase over time (Brockmann, 1986; Shneiderman, 1987; Walker,
1987; Duffy, Gomoll, Gomoll, Palmer, & Aaron, 1988). Therefore, we anticipate a significant movement
of help systems to online delivery. Just what are those advantages? Online delivery supports the
following:

**Greater availability**

With networks and portable computers becoming more prevalent, we anticipate that it will become increasingly unlikely that adequate hardcopy documentation will be available for all software applications at all delivery sites. Online information can provide a reliable source of information for all software packages and on all delivery platforms.

**Easier access**

Online, the system can provide mechanisms for efficient access to the relevant information, especially in cases where that information might span many volumes of hardcopy documentation (see, for example, Walker, Young, & Mannes, in press, for a discussion of the use of documentation for the Symbolics platform).

**More interaction**

Online, both the user and the system can interact with the information. For example, the system can use the state of the application (its current context) to determine what information to provide to the user, or a monitor capable of plan recognition could help debug a user’s faulty or inefficient procedures.

**High accuracy**

Hardcopy documents require much longer production cycles. As computer companies adopt shorter and more efficient software development cycles, the pressure to adequately document a product increases. The time it takes to produce a book after it is written—layout, formatting, and printing—becomes a bottleneck. Either manuals go into production well before products are stable (resulting in manuals that are inaccurate or incomplete) or a company incurs costs in order to make the necessary changes and to begin the production cycle again. This production bottleneck is minimized with online documentation, allowing for a more accurate representation of the final software product. In addition to the initial production, documentation and system updates are more easily supported for online documentation. Therefore, once again, accurate information is more readily available.

**Low cost**

In general, online information is less expensive to store, reproduce, and distribute. This is partially a matter of the size and weight of a disk as compared to a manual. Of course, the reproduction process is also more easily accomplished for electronic information than for print information.
Online information can exploit multiple media, such as video, sound, and animation, and can apply techniques from Artificial Intelligence (AI).

Of course, there are disadvantages to the online medium as well. First, with most computer systems, users can’t work in the primary application while using the online information. In contrast, users can work comfortably with an application while using an hardcopy tutorial or manual. Second, it is well documented that most computer displays diminish the readability of text and the legibility of characters, two factors which make reading from screens more difficult than reading from a book or manual (Haas & Hayes, 1987; Kruk & Muter, 1984; Muter, Latremouille, Treurniet, & Beam, 1982). In general, these types of problems are technological: and advances in computer hardware and software will certainly reduce or eliminate these difficulties in time (see Haas & Hayes, 1988, for a study comparing hardcopy to standard and advanced displays: as displays improved, so too did the similarity between reading online or from hardcopy).

Importantly, however, there also remain unsolved conceptual problems. For example, the familiar strategies for navigating through books do not apply to online information (see, for example, Elm & Woods, 1985; Robertson & Akscyn, 1982). Users, therefore, must learn how to interact with the new medium. And this is not a simple problem, especially since the technology of hardcopy text is an integral part of our culture. However, well-designed navigation systems can improve considerably the user’s ability to adopt the conceptual framework necessary for this new technology. Egan, Remde, Landauer, Lochbaum, & Gomez (1989), for example, compared their online system, SuperBook, to its hardcopy equivalent, and found that subjects performed better with hardcopy and better with the online version depending on the task. Overall, however, users reported that they preferred the online system over the hardcopy text, a finding which suggests a positive future for similar, advanced online documents. Similar findings are reported by Walker, Young, & Mannes (in press).

Certainly, however, in discussing current online systems versus hardcopy delivery we must consider the issue of acceptance by the user. Not all studies report the same success as the Egan, Remde, Landauer, Lochbaum, & Gomez (1989) paper. There are some critics who would argue that users reject online assistance systems. Users simply don’t accept and refuse to use online information. So why should we bother considering online delivery? Why not just present everything in the hardcopy form? We have two reactions to this proposition.

First, there are indeed situations where the user will prefer hardcopy documentation. For instance, hardcopy will be preferred and necessary when computing power is unavailable such as in attempting to start-up the machine or when fatal crashes occur. Also, if the use of the information is going to involve reading for a long time, users are likely to prefer relaxing in a chair with a book rather than paging through a file. Similarly, if studying is the goal, again requiring a significant amount of time with a particular text, the user will once again probably prefer a manual. And finally, if the delivery system does not have windows available and forces the user to actively remember a great deal of information in going from the aiding system to the application, then a manual will be preferred in 99% of the cases (though even here the availability of efficient search mechanisms in online delivery could encourage users to prefer it over hardcopy for large systems, for example, nuclear power plants or mainframe operating systems).

Our second reaction to the argument that hardcopy is a preferred form of information delivery is that rather than simply abandoning online delivery because of user comments, the comments should be a basis for re-thinking the design of online delivery to better meet the user’s need. Users’ may be rejecting online assistance simply because it is poorly designed—not because of inherent weaknesses. For
example, one reason users might prefer hardcopy texts to online text is because the online versions don’t support windowing. Who can blame users for not being willing to use a system where getting information requires exiting the application, entering the aiding system, getting the appropriate information, existing the aiding system, and re-entering the application?

The technological problems are being overcome—windowing is generally available and large screen monitors are more and more common. However, an additional problem exists: most help systems are very poorly designed. As designers, we need to begin re-conceptualizing the design process and the design principles for online aiding if we are to make online the preferred delivery medium.

Issues in design are perhaps best understood by examining the history of hardcopy documentation. Originally, hardcopy documents were written by experts for experts. The primary goal was to document the system. However, when the personal computing market exploded dramatically in the 1970s, few designers realized the need for a new design perspective; that is, very few changes were made in the design process or to the well-established principles driving that design—and, in turn, users began rejecting the types of documentation being produced by the computer industry. Since then, the development process and the design principles have evolved, improving the quality of manuals so both manufacturers and end-users consider manuals an asset to the software product (Mills & Dye, 1985). Moreover, manufacturers are well aware that, in today’s competitive industry, poor documentation can spell financial catastrophe for a computer company (see, for example, the headline “Hundreds of Coleco’s Adams are Returned as Defective; Firm Blames User Manuals” in the “Wall Street Journal,” 30 November, 1983; Schriver, Hayes, & Langston, 1986).

Online help is still in its early stages and, like the early days of hardcopy documentation, still carries the vestiges of outmoded beliefs. While the understanding of design requirements is growing, many designers and developers still consider that the development of online help involves simply putting the manual online or providing some quick reference information. Fortunately, there is a growing recognition that the design requirements for online presentation are different from those of hardcopy documents. The industry is now re-examining the design process (see Chapter 4) to adjust to the new demands of creating online help, and they are searching for effective online help design principles (see Chapters 3, 4, & 5). The move is afoot to improve help systems and to change the preference of the users.

Another reason to pursue online delivery is that, in our viewpoint, the online medium is the only viable delivery system for the future. Many of the advantages of online delivery are outlined above. However, let us summarize what we argue are the four most important issues for the future.

- Technological advances are removing the barriers to online delivery and presenting enormous communication advantages for online presentation, for example, through digitized video or intelligent systems. Communicating online is quickly becoming the preferred alternative.
- The proliferation of software and hardware products makes document management and hardcopy storage virtually impossible for the end-user. We might imagine, by the year 2010, that each corporation would require a special warehouse reserved for documentation alone.
- The increased networking that allows software to be centrally stored and distributed amongst numerous facilities (perhaps located across the country or in different countries) make it difficult—if not impossible—to provide adequate documentation to every terminal or workstation.
- The increased cost efficiency of storing, distributing, and updating online materials (and the concomitant increase in the cost for hardcopy materials) make online delivery the cost-effective alternative.
In sum, because different media require fundamentally different design considerations and because media differ dramatically in delivery capabilities, we consider the medium of delivery one of the two primary dimensions for classifying aiding systems. Indeed, we expect user preferences for media to change as both the design and computing technology advance.

**Dimension 2: The Goals of the User**

Any classification of systems that aid users must focus on the goals of users. After all, the goal of any aiding system is to meet users’ needs, that is, to help them achieve their goals. This dimension in the classification of help systems ensures that the user’s task significantly influences subsequent specifications for design as well as the contents of a help system (Smolensky, Monty, & Conway, 1984). Indeed, specifying user goals as a dimension assumes that different user goals lead to fundamentally different aiding systems. We feel that this is a reasonable assumption.

So what kinds of information do users want or expect? What questions should information systems be able to answer? Note that we view the function of information as it applies to the context within which that information is being used, not the writer’s intentions or the static description of its form (compare with Bethke, Dean, Kaiser, Ort, & Pessin, 1981). The core idea behind our effort is to match the information provided to users with the different kinds of knowledge that they require.

We can describe three user goals in terms of the expressed statement of need:

**I want to buy it**

The exemplar for information that meets this goal is the sales demonstration, sales book, and the specification sheet. The audience is usually prospective buyers. Their goal is to (perhaps) buy the product or to understand how they can use its services; ultimately, the information reflects a persuasive aim.

**I want to learn it**

The exemplars for this category include tutorials and guided tours (which may emphasize a successful first experience—“the affective response of the user”—as an adjunct to the goal of learning). The users’ goals are to learn what they can do with an application and how they can perform some set of important or fundamental tasks. Constraints on attention generally limit this set of tasks to a group of basic skills, although more elaborate forms of instruction, much like a course or curriculum, also exist. From the user’s point of view, however, the distinguishing feature is the goal of learning rather than performing. Typically, the context is a set of artificial situations constructed so that the user can practice using the application; the user is not actually performing real job tasks. The audience includes both the prospective buyer and the novice or transfer user.

**I want to use it**

The exemplars for this category include reference information and procedural information. Critically, the context of the request is the actual work situation. The user is trying to perform a task with the application. The users’ goals are to overcome impasses that prevent them from proceeding on their task. The audience is all users, depending on the type of knowledge they require. For example, the
novice typically needs task-oriented information, while the expert wants access to reference material.

**Dimensions We did not Include**

The reader may note that our classification of aiding systems does not include some dimensions that others have typically associated with such a classification process. In particular, we did not include a dimension that describes the document type nor did we include a dimension of user expertise. Let us consider each of these dimensions in turn.

**Document or Information Type** Generally, document design researchers have tended to classify types of aids according to document type (see, for example, Gleason & Wackerman, 1984; Mack, Lewis, & Carroll, 1983; O’Malley, Smolensky, Bannon, Conway, Graham, Sokolov, & Monty, 1983; Palko, 1986). For example, Schriver, Hayes, & Langston (1986) have suggested that four categories of texts dominate the literature: tutorial, user/operations guide (procedures), reference, and quick reference guides and manuals. This list might be extended with other types of documents such as the “open me first” document or the first experience document.

Implicit in any categorization by document type is an assumption about the “function” of the information being presented. That is, we might rather choose to describe four information functions: instructional, procedural, explanatory, and facts (or specifications). In this respect, document type (or description) and information function generally correspond to one another in terms of the core information contained in any one particular document.

We rejected this as a dimension primarily because it seems to focus on the wrong issue. It, in fact, seems to “put the cart before the horse.” That is, analysis of user needs should lead to the specification of types of information and types of documents that will satisfy those needs.

We should emphasize that there is not necessarily a one-to-one correspondence between user needs and information elements. Thus, supporting “I want to do it!” may require facts, procedures, and perhaps even some explanation. The selection and organization of information elements is determined by considering what is required to fulfill the need.

**User Expertise** Even more common than the classification by document type is the classification by user type. At the basic level, this is a dimension that is identified by its extreme points—expert and novice. Recently, however, an intermediate point, the transfer user, has been of focal interest (see, for example, McDonald & Schvaneveldt, 1988; Shneiderman, 1987).

Kearsley (1988) goes beyond levels of expertise and identifies three dimensions to which those levels may apply. In his “conceptual models of help,” he indicates that users may be defined in terms of their expertise with the computer, with the particular task domain, and with the particular application software. If we just use the extremes (expert and novice) of each dimension, Kearsley’s (1988) model would yield nine user types (“novice with the application, experienced with the computer, and experienced with the task area” being just one type of user). Kearsley (1988) further suggests that “each of these types of users is likely to need slightly different types of help” (p. 5).

We certainly agree that knowing the skill level of the system users and providing support consistent with that skill is critical. However, this issue is only one of many other information design issues. For example, we also have to be certain that the information is written at a level of syntactic complexity that the user will be able to understand and in a language he or she can understand. In sum, while user skill level is a critical design issue—and a major focus in any task analysis—it is not a dimension along which to classify the goals of helping. It is a design issue that applies to every help system.
In short, we have several reasons for believing that defining help systems in terms of user expertise can lead to ineffective and impractical design.

First, a system of documentation based on user expertise does not take into account the user’s perspective.

The very task of selecting the appropriate help system shifts the focus of the user’s thinking from “how do I?” or “I want to learn about....” to “Let’s see, how much do I know about this product, about computers, and about this task domain?” Therefore, having different help systems categorized according to user expertise forces the user to focus not on his or her task but on the task of attempting to select the appropriate help system.

Of course at some point in the future, expert systems might be able to reduce some of this burden. However, the expert system would still have to be able to identify the user’s level of expertise (on three dimensions—and maybe more) for the particular task he or she is working on. Additionally, the user, at some point, would have to identify his or her overall level of expertise on each of the dimensions—so that the system could provide a beginning place for providing help. We question whether people can accurately assess their expertise even on one of the dimensions (for example, computer or task expertise), much less on all three (and we must presume accurate placement is necessary, otherwise there is no need for different help systems).

Help systems based on expertise seem to us impractical from a motivational point of view. While people may be willing to classify themselves as “beginners,” we do not believe that they will select the beginner’s help—especially if they are experts in other domains.

Second, a system of documentation based on user expertise is impractical from a development point of view.

Kearsley’s (1988) expansion from one to three dimensions of expertise illustrates the combinatorial explosion that is possible. Should experience with particular platforms be a defining factor? That is, do we need a dimension of general computer expertise and then another dimension for experience with a particular platform? Certainly Macintosh experience is important if you are working on a Mac—but so is an overall level of computer experience. How, for example, does experience with a mainframe statistical package transfer to the task of using Dbase III on a microcomputer. So far we’ve only considered the extremes of the dimensions—wouldn’t we need help systems for the intermediate level user? It would seem that this level is very important on each of the four dimensions. We are now facing the horrid task of designing 80 different help systems.

Third, categorization by user expertise classifies individuals rather than the individual’s knowledge of particular parts of an application.

Few people are experts on all aspects of an application. The power and diversity of applications is such that, in many cases, the “expert” has not explored particular capabilities. Moreover, new peripherals and boards often open new capabilities of existing applications. An experienced individual may well require a tutorial or procedural information on unexplored features of a program. Similarly, the relative novice may develop proficiency for particular uses of an application—in these aspects of the application, then, he or she could behave like a “power user.” His or her information goals in that domain may be similar to that of the more general expert. Importantly, therefore, the user’s goals—rather than his or her overall skill level—is the determining factor.
Fourth, categorization by user expertise ignores the tremendous overlap in user knowledge and information goals.

Duffy, Ackerman, Grantham, & Kelly (1985), for example, interviewed business users of microcomputers, system operators of minicomputers, and engineers using programmable controllers. The goal was to understand, in some depth, how people used system documentation. Duffy et. al. (1985) classified the users into expert and novice users based on their relative amount of experience and training, though, in fact, none of the users were novice in the sense of having no experience.

As part of the interview, the users were asked to think back about their prior use of manuals and consider the kind of information they were searching for. In particular, they were asked to classify their information searches into searches for facts (specifications), procedures, or explanations. They were then asked to estimate the proportion of their searches that fell into each category. Interestingly, none of the users had any difficulty with the task: they all had a clear sense of each of the three categories of information.\(^1\)

The findings, presented in Figure 2, show that there are indeed differences in the information needs of experts and novices. However, most importantly from our perspective, a tremendous overlap in user goals is also evident. Experts do, in fact, look for explanation as well as procedures. And novices are not just searching for procedures; they too want facts.

<table>
<thead>
<tr>
<th>Expertise</th>
<th>Expert</th>
<th>Novice</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information</strong></td>
<td>fact</td>
<td>proc</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>microcomputer</td>
<td>.36</td>
<td>.28</td>
</tr>
<tr>
<td>minicomputer</td>
<td>.33</td>
<td>.20</td>
</tr>
<tr>
<td>programmable</td>
<td>.40</td>
<td>.21</td>
</tr>
<tr>
<td>controller</td>
<td>.17</td>
<td>.57</td>
</tr>
</tbody>
</table>

Figure 2: Proportion of tasks devoted to searching for facts (fact), procedures (proc), and explanations (explan) as reported by more or less experienced users.

In closing this discussion, let us re-emphasize that understanding the user’s knowledge is critical to the design of an effective help system. However, there are multiple user types asking the same questions for much the same purpose—and there is a continuum of expertise. How one deals with those multiple users is a critical issue in the design of each and every help system (see Chapters 3 and 4).

---

\(^1\) The documentation did not contain tutorials, which is why Duffy et. al. (1985) could not ask the extent to which they looked for tutorials.
Online Help Defined

Online help is the online delivery of performance-oriented information. It is information presented online that is designed to answer the question “How do I?” (see Figure 1). The difference between a learning-oriented aim and a performance-oriented aim is critical. We restrict our use of the term “online help” to systems that support performance. The user of online help is trying to complete some task in an application. The information required may be facts, procedures, or even explanation. However, because the individual is in the midst of the task, the information must be

- targeted to the tasks
- accessed efficiently
- written in a style that leads to efficient transfer from the help system to the task.

In contrast, online tutorials and training materials support the goal of learning. There is less urgency to the learning situation. The goal is to build generality and to represent the learned information in long term memory. In such a domain, it is acceptable to take the learner through a series of contrived learning tasks to illustrate a point or to facilitate learning.

In the following paragraphs we will attempt to address some parameters that frequently lead to confusion in distinguishing online help from other types of assistance.

*Error correction versus error detection*  Online help systems provide users with information that allows them to continue their task. Error notices presented in the application are not help.

*User versus system initiated information*  Online help may be initiated by the user, or the system may detect an error and present information to correct that error. Either system would be classified as an online help system since the information is efficiently answering the question “How do I?” However, since systems that initiate information are as yet in their developmental infancy, our focus is on user initiated quests for help.

*Primary versus secondary sources of online information*  Online help is a secondary source of information. The information supports the use of a tool (the application) which itself is used to accomplish a primary task. In contrast, online databases, for example, online encyclopedias, or databases of newspaper articles, research reports, et cetera, are not online help systems. These are sources of information directly applied to the solution of a real world task.

*Online documentation versus online help*  The distinction between online help and online documentation is one of design strategy. Both attempt to provide answers to the question “How do I?” and—with effective search tools and text (see, for example, Walker, 1986)—both online help and online documentation can be efficient sources of information. As we will discuss in Chapter 3, the limitations of the delivery platform may well dictate the appropriateness of online help or online documentation for a particular application.

Conclusions

The goal of this chapter has been to define online help and to place it within the context of other support systems and other types of computer systems. We began by examining the growth of the computer industry and alternative ways of supporting computer users. This, in turn, led us to a discussion of the factors we believe will lead to the growing importance of online help, for example, cost, growth
in networking, the ability to individualize or tailor information for computer users, et cetera. Finally, we defined online help in the context of other “types” of help systems.

An Overview of the book

Chapters 1 through 3, in general, answer the question “What is online help?” Chapters 4 and 5 emphasize the design of online help systems, and Chapters 6, 7, and 8 focus on the evaluation of online help systems. We believe that, given our emphasis on design implementation and on system evaluation, the book will be of interest to both practitioners and researchers. Researchers will find chapters 3, 7, and 8 of particular interest; in chapter 3, we outline our model of the users of online help systems and, in chapters 7 and 8, we show how that model can be used to design an evaluation tool for rating the effectiveness of help systems that support a variety of software applications across different computer platforms. Practitioners will find chapters 2, 4, 5, and 6 of particular interest; chapters 2 and 4 summarize research relevant to designing online help systems and provide advice on how to apply this knowledge to different design situations; chapters 5 and 6 present the concerns of actual online help designers and outline strategies for effectively evaluating online help systems. A brief summary of the chapters in this book are as follows:

• In Chapter 1, “Online Help in Context,” we argued that the medium of delivery is one of two dimensions relevant to identifying issues in the design of online help systems.

• In Chapter 2, “Medium of Delivery and the Design Process,” we will elaborate on this issue, discussing how medium affects design. The primary focus of the chapter will be on presenting original research that examines how designing online help differs from designing hardcopy manuals. The data is derived from interviews with individuals in the computer industry who have developed commercial online help systems.

• In Chapter 3, “A Task Model for Online Help,” we will present a theoretical perspective towards online help. Our objective is to provide a framework for structuring one’s thoughts about online help as it applies to the research literature, as a means of guiding the design process, and in order to effectively evaluate help systems. Our emphasis will be on the user’s perspective. Hence we will consider the context of the user when he or she needs help and the task of the user as he or she attempts to obtain and apply help information.

• In Chapter 4, “Designing Interactive Online Help Systems,” we will present a review of the research literature on interface and online help design. Rather than a broad literature review, we will explore several topics in depth. There are two goals in this discussion. First, we want to place the research discussion in our conceptual framework. We want to address the complexity of research findings and provide a framework to assist readers in organizing their thinking about, and understanding of, the complexity of help design literature. Second, we want to derive design principles from the research. We discuss and evaluate alternative strategies for aiding designers, focusing on the traditional strategy of providing design guidelines. The shortcomings of design guidelines will be presented and then we will present our alternative “modelling” approach as a means of providing a framework for interpreting guidelines. Topics covered will include menu design, menu item selection, navigation aids (including hypertext issues), help content, screen format, and the comprehensibility of the help text.
• In Chapter 5, “Understanding the Design Process,” we present original research on the design process. The work is based on a quasi-Delphi study of the design process conducted with 19 help designers representing 14 different computer companies. Our goal is to describe the design process by identifying the major tasks or process involved. These major tasks are then further analyzed by looking at particular subtasks, the relative importance of each subtask, and the major problems or constraints encountered by the designers of online help systems.

• In Chapter 6, “An Overview of Evaluation Requirements and Options,” we will outline our approach to evaluating online help systems. As well, we will discuss the goals of evaluation, that is capturing system usability across different help systems. This will entail a detailed discussion of usability, particularly, the distinguishing of accuracy and completeness issues from comprehensibility and accessibility issues. We will consider alternative evaluation strategies (for example, benchmarking, usability testing, expert judgement, rating systems, etc.) and, finally, we will present our cognitive task analysis as an effective means of assessing (a) accuracy and completeness, and (b) access and comprehensibility.

• In Chapter 7, “The Help Design Evaluation Questionnaire (HDEQ),” we will present the details of our instrument for assessing the design of online help systems. HDEQ itself (17 pages) will be provided as an Appendix to the chapter. Hence the chapter will contain a brief synopsis of the rationale for our approach to design evaluation and a detailed presentation of the rationale for particular items in HDEQ (including examples of high and low assessment scores).

• In Chapter 8, “Findings from the Evaluation of Help Systems,” we will present findings from our evaluation of 28 online help systems for commercially available applications. Data will include the reliability of HDEQ scores, the overall rating of each help system, and the rating of each help system on how effectively it supports each of our eight user tasks.