View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Illinois Digital Environment for Access to Learning...

CHARLES T. MEADOW

Professor
Library and Information Science
University of Toronto
Toronto, Ontario

Tailoring System Design to Users

We seem to have finally reached the point of almost general acceptance of the concept that computer software should be designed for ease of human use. There remain questions, however, of whose ease and of what is meant by tailoring design to users.

Design for Whom?—Some Fallacies

Thirty years ago software had to be designed first to accommodate hardware. Whether it was speed or reliability at issue, the foibles of the computers had to be catered to. They were expensive, more so than their programmers or operators, so operating speed was a critical factor. Software designed in this manner is mainly gone now.

The convenience of the programmer is another possible design goal. It is simply easier to write a parser of syntactic statements that says "syntax error" than to have that program explain what the error was, how it was detected, or what the user might do about it. A spreadsheet program does not have to wait until you have used up all available space and then announce a fatal memory overflow error. It could warn you ahead of time, but it is easier to wait.

Yet a third object of design may be an erroneous assumption about user characteristics. Program authors are not always thinking selfishly of their own interests. Sometimes they mean well but do not understand those who are not professional programmers or, horrors, those who do not have much interest in programming. This author began his career as a professional programmer, a profession to be respected. I understand and still share the fascination with computers and getting them to do one's bidding. But one of the aspects that was always the most fascinating was the

different levels at which one could deal with a computer. There was never much interest in the hardware. I was always content to view the machine as defined by its programming languages. But the programming languages do not normally include any way of dealing with intricate timing problems or unreliable discs. To some extent, a user has to know the hardware, but for me it is always the least that can be gotten away with and never the most that can be learned.

The point here is the belief that some programmers assume that all users are as interested in the detailed operation as they themselves are. They do not reply to an error with the message "syntax error" out of meanness, but because they truly believe that all users: (1) know the syntax or the general language structure thoroughly, and (2) will not rest until they have worked out the nature of the error. That, after all, is how the hypothetical program author would have behaved. This seems to be the basis upon which UNIX was designed—an operating system that generally assumes everything is being done correctly and either does not respond to errors or does so only minimally. (UNIX does not require that it be used in this way—organizations can build in their own user interfaces, but apparently not many do so.)

In fact, however, many simply lack the time and interest to pursue intricate details of program malfunctions. They may go to the extreme of abandoning use of the program rather than spending an entire evening reading the users' manual. It is a defensible point of view, but one that does not often enter the consciousness of software designers.

The so-called end user is more likely to have this last point of view than he or she is to share the programmer's point of view and fascination for computer details. The key is to design software for the user and take the trouble to find out who that user is.

Who are End Users?

Who then are end users and what do they want from their software? The very term is something of a negative descriptor. It is not truly a positive description of a person. It is merely an assertion that this is a person who is not a search intermediary, or, in programming terms, not a professional programmer. Whoever they are, the world is only now discovering them and making the usual noises of discoverers enchanted with their finds. But they have been around forever. An end user is simply a consumer rather than a provider of information service. We all play the role at some time, even if information service providers by profession. Hence, an end user is someone operating, at least temporarily, outside his or her own profession.

To design for end users, it must first be determined who they are, what their interests are, and what they do when they are using the particular type

of software in question. In other words, tailoring design to users does not, and cannot, mean treating all users alike. We should try to negate the earlier statement that end users are temporarily operating outside their own professions and bring searching into their various professions.

Often software is described in such terms as natural language interface, menu-driven, or user friendly as if these words alone guaranteed proper user support. But the first two are technical terms descriptive of the means of representing information. They are not sufficient to describe a means of communication between user and system. It has to be known what is to be said in natural language, or what information is to be represented in a menu, how it is represented, and in what context it is presented. The term *user friendly* has no meaning at all to this author.

Why is it Important to Design for Users?

Learned language affects the understanding of the world around us. That is essentially the hypothesis of Benjamin Whorf (1956) the linguist. If that language is a computer programming language (which domain includes database command languages), then a user is going to understand computer functions in terms of the language learned. If the language is low level, difficult for the person to learn, and limited in functional capability, then the user's perception of what can be done in a database search is necessarily limited. If users are to treat database searching as a routine part of their own professions, they must be allowed to "see" database searching in their own professional terminology.

Some Cases

In-Search

Probably the best known intermediary software for online database searching, until about a year ago, was In-Search (Newlin, 1985). It originally appeared in 1983 and was intended as an end user intermediary for DIALOG, presumably for any user of any of DIALOG's then about 200 diverse databases. Its advertising stressed ease of use compared to use of the DIALOG language.

My personal assessment was that it would not work out. This, of course, is stated with the benefit of hindsight. But In-Search had a rather thick users' manual, came with six discs (including four holding the content of DIALOG's "blue sheets" or brief database descriptions), and many found that installation was relatively difficult. The original plan was for the database descriptions to be updated by direct downloading of the corrections to the user's computer. This plan was never realized.

It had basically only one way to search. True, there was no need to learn commands, but a user did have to use Boolean expressions, previous set numbers, truncation, proximity, etc. Hence, In-Search forgave only use of the name of a command for searching. It had no active help; it merely simplified the command language.

On the good side, the programming craftsmanship was superb. There was imaginative use of windows, and the method of searching one of In-Search's files, such as that of database descriptions, used simulated index cards. Everyone knows how to do that without any real instruction. Each card had a tab with a short title or code, and the current card popped up as the user riffled through the "deck." This is what happens when a manual file is searched. Truly, here was a form of searching that needed no instruction. But this was not how DIALOG was searched. It was only how the database descriptions were searched. The telecommunications were good. This part of In-Search later served as the basis for DIALOGLINK (Witiak, 1986).

The idea to convert In-Search to Pro-Search (Quint, 1986) was probably a good one. It reduced the market but ensured a better fit of product to users. When it was converted, a new accounting function was added which is of great use to institutional searchers who must charge services back to clients.

Info Globe

Info Globe is an online database containing the text of the Toronto *Globe and Mail* newspaper (Ross, 1981). Its basic search language is somewhat cumbersome. What is interesting is the array of front end techniques that Info Globe has developed to overcome this, tending to be fitted to the situation. An example is a method of scrolling of headlines based on selection of articles by use of menus and a few key words. The menu portion might get the user to the point of asking for a search based on company name. The user then enters the name and scrolling begins, showing date and headline.

The advantage is that if you know what you are looking for and approximately how to get it, it goes very fast and places little demand on the user. The versions this author tried were developed for securities analysts; people who probably can be relied upon to know how to spell the company name without much difficulty. In this sense, it is fitted to the user's knowledge of the field in which he will search and his assumed lack of interest in having to get too involved in a search.

A disadvantage for anyone who knows much about searching is that a complex search can be difficult to express. But choices must be made if products are to be designed for users, and no product is going to be well suited to all potential users.

Stock Quotation Systems

While having no personal experience with this class of system, they are intriguing because at least some of them operate on the basis of a single type of query—i.e., one may ask for current information on the shares of a given company, or not ask—the question is constant. The single variable is the name of the company entered as a stock market symbol. The output is fixed, consisting of latest price, recent high, low, sales volume, etc.

For persons not familiar with the market, this system can be useless because they may not know the ticker symbols and they might not know how to interpret the data. A highly specialized language is in use, but the people who work in this field or who trade in the market all know the language, so its use causes no trouble. It takes time to develop the skill, but this is not time taken out of the regular job—it is part of the job and the skill would have to be learned with or without the computer system.

Aspects of Systems to Consider

What aspects of a system can be tailored for the user? Realistically, expect that there are limits and that software people have always been able to conceive of and promise far more than could be produced.

Databases

In general, the search system is considered and not the database. If the database does not have the kind of informaiton the searcher wants, there is relatively little the search system can do about it, except to recommend or help find another database. This author has tried several times to find material in *MEDLINE* that can give information in layman's terms about some medical condition a personal physician has not fully explained. It does not work well; *MEDLINE* does not generally include such material, and the database cannot be faulted for not having what it was not intended to cover, nor can any intermediary find information that is not there. On the other hand, in the early days of the industry, ERIC on DIALOG did not permit limiting a search by publication year. Instead, the searcher had to use ERIC accession numbers. The real pros kept a table showing the first number occurring in each year. A search service or a search front end could easily have done the same and made the conversion for the less experienced searcher.

Display

Dealing with full text is another example of where the search service or mediator can overcome inherent database problems. The search software, whether centralized or front end, can divide the text into smaller segments and can use highlighting to help the user see the context that would lead to

retrieval of any text segment. For example, the option to display only the individual paragraphs that contain search terms can be offered. Without this, if only the full item were retrieved, browsing would be more difficult.

Time and Cost

Fiscal and time controls can be built in which are often necessary especially when a library allows patron access to its accounts and must exercise some sort of control over use. A very simple example is the practice, now fairly popular, of allowing the user to prepare and store a query before automatic logon to the search service. A similar technique used by EasyNet and Search Helper is to run a search, download some records, then log off. While this detracts from the full interactive potential of most retrieval systems, it does accomplish the objective, important to some users, of tight cost control.

Types of Question

Most accommodations to user style, interest, and knowledge in use of software can be made in language and type of question asked. First, just as in the stock market systems, there are often quite standardized questions in any profession that require only the addition of one or more values at search time. Such questions are coming to be called scripts. Selective dissemination of information (SDI) is a form of this; the search is defined once and then run repeatedly, changing only the date or database update code each time. The same can be done for company information searches—for example, defining what kind of information is sought beforehand and filling in the name of the company at search time. It can be done in searches for articles about the toxic effects of a substance where the substance name is provided at search time and the toxic effects portion defined well beforehand.

As an extension of this concept, professional searchers might prepare for their company or laboratory an extensive library of search segments or scripts, carefully and completely defining key subjects of interest. These could then be assembled by the user who could add his/her own information—such as dates, corporate sources, or authors of interest—at the time of the search. The tailoring process comes with the design of a system that can store and assemble search fragments. The creation of the fragments is up to user organizations.

Vocabulary and Syntax

There are two sets of vocabulary and sometimes syntax to consider—e.g., that used to describe the content of records and that used to tell the retrieval system what to do. First consider the content descriptive language. There is no substitute for the searcher's understanding of the vocabulary of

the field in which he or she is searching. There are ways to help find terms, but it is hard to imagine successful searching of, say, chemical or legal files not based on a good grasp of the vocabulary of the fields.

A number of computer assistance programs were developed to aid in selection of the vocabulary or its improvement based on preliminary trials (Marcus, 1983). But in general they depend on the user to be able to recognize which of the offered "associated" terms would be most helpful where there is a host of definitions of word association. This is quite reasonable: if the person does not know the vocabulary, there is hardly any way to judge the value of—hence to select—retrieved items.

When it comes to the language that is used to tell the computer what to do, rather than what the desired information looks like, it is important to avoid bringing in new vocabulary and usages. These can intrude on the search process or require the investment of significant amounts of time before the search process can begin. Longitudinal studies of how users have adapted over time to a new information system are rare to nonexistent. Lacking formal proof, one must make do with informal studies and intuition.

Experience and intuition indicate that users do not want to spend much time learning to use systems. They want to start right in using them. Then, as they become more experienced, they will gradually recognize what they cannot do and look either for improvement in the system or for greater knowledge of how to use it. To designers, this means making the system simple to learn but expandable. A requirement for learning is acceptable when users have a precise goal, know what they want to do, and seek instruction in how to accomplish their goals. In other words, when a user wants to be able to change the sequence of records in the output, he will be amenable to seeking instruction in use of the sort procedure. But seeking instruction in information before using the system will not meet with ready acceptance.

Of course people vary. The suggestions just stated apply to myself and to many others, but there are also those who want to start a new endeavor by reading all about it first.

The Expected Output

One of the great fallacies of online searching is that everyone wants high recall or even high precision. It is a fallacy to believe that the average searcher even thinks in these terms let alone uses them for setting a goal.

There are, however, recognizable classes of objectives which have system design implications. There are users who want "everything" on a subject, one implication of which is high volume output. Another possibility is that the results will be used as part of a document, hence the need for processing of the retrieved records in the correct format. Yet another

common possibility is that the searcher wants only "a few good items," which implies no particular need for any further processing or storage beyond the ability to print or download a few records.

These goals can be designed for fairly easily. Downloading is now common and often the records can be in a format acceptable to word processing software. For those needing programmed reformatting there is such software as Pro-Cite (Hoyle & McNamara, 1987). For those needing no special handling, none need be provided thereby avoiding asking the user cumbersome questions about what is wanted and how to get it.

How to do This

In any activity, more options mean more decision-making; this, in turn, means that more information must go from the system to the decision-maker and from the decision-maker to the system. This is true when doing an online search, buying a computer, or buying a shirt. When buying a shirt, however, the task is usually simplified by such measures as adopting favorite brands, shops, or colors. Every type of fabric or style need not be investigated because we know about them or know as much as we want to know. There is a willingness to risk a mistake to avoid the time required to find out about possible new fabrics each time one shops.

In computer buying, the buyer is probably doing the corresponding thing—using brand preference or looking for a clone of a preferred brand and a fairly standard configuration. If an experienced buyer, one may well be fatalistic and take the attitude that any selection made may well be outdated by a new announcement next week—and not worry about it.

In online searching, we are to some extent still expecting customers to consider everything, to invest the time and energy required to make the most intelligent decision each time. This is just not realistic. Just as there is a limit on how much we care about an error in shirt buying, most users have a limit on how much they care about online search results.

Hence, a critical design feature of any search service should be that it not ask more of the user than the user wants to invest in it. Command language systems tend to ask too much of end users. There is too much to learn before effective use. This means that search systems should be designed to have a different appearance to different classes of users.

How can this be done? There is a need for more than merely the option to replace long computer messages with short ones. There should be systems that truly behave differently for different users. Someday there may be computer programs that are able to determine the level and intensity of use the user wants and to respond accordingly. This lies in the domain of artificial intelligence. Human beings are not universally adept at this. It may even be suggested that those with great depth of subject matter

knowledge will be most reluctant to adapt themselves to the unknowing client, and that those with the greatest adaptation skills may lack depth of subject knowledge. It is not even clear, if this hypothesis is accepted, that there is a good model of human behavior to try to replicate in a computer program.

A personal preference for a short-term approach to this problem (the problem, remember, is designing a system that has the right appearance or makes the right demands on its user) is different programs for different situations. There is no universal reference book. It is accepted that different subjects and depths of presentation demand different typographic and contextual designs. Users do have to invest some effort in finding the appropriate book, but repeat users will return to the same sources again and again and not have to go through the selection process each time.

We can do this—continuing to borrow a design concept from the print world—by allowing individual authors who can visualize a problem situation to design for that situation. A "situation" is a combination of a set of available information and potential users. The physician interested in current information about drugs on the market has a different need than the research pharmacologist; the average undergraduate has a different need than the average professor and so on. In all these cases, the different groups may have not only different needs, but different searching or computer-using skills and available funds as well.

There are a number of search "front ends" or computer intermediaries available, and they do reflect different user needs and skills. Pro-Search has high expectations but allows the skilled user to exercise all the many logical search options DIALOG (and now BRS) can offer. EasyNet (O'Leary, 1985) requires no preparation before use but limits the precision with which the user can control a search. My own system, OAK (Meadow et al., 1989), which was developed for the Department of Energy, falls between these. The user is expected to be well educated but not necessarily in searching techniques. OAK, like EasyNet, requires no prior instruction, but it is offered and better results can be expected if it is used. The system is designed for bibliographic database searching. It would not serve well, in the same form, for numeric database searching, nor would the others mentioned. There are different relationships to be stated, hence different language requirements for doing so.

A program like DIALOGLINK is not a front end in the sense that this author means it. DIALOGLINK does not change the language of communication with DIALOG. It helps a great deal with telecommunications and allows for local storage of the search statements, but users still talk DIALOG's language.

To accommodate the diversity of front ends that are needed, the central services should be designed to expect their use and not merely tolerate

them. No software change is required to do this. The central system does not know whether it is receiving its command language statements from a person or a computer since the normal means of operation has the front end translating the user's expressions—however they may be stated—into the target system's commands.

A central service could design a lower level command language, one requiring less effort on its own part to parse, and then require that all use be via a front end. One form of front end, of course, could deal in the traditional command language. But others need not use anything resembling the "native" languages. An incentive for the central services to do this is that the parsing function would be largely carried out at the user end in the user's computer. The user is connected for the same amount of time, but the work is done outside the central system thereby increasing its work capacity.

Another requirement of this mode of operation is that central services be prepared to warn independent front end designers of impending changes. This is not a characteristic of the industry now. Both search and communications services are willing to make changes without warning other software vendors who are dependent on them.

In this way, any central retrieval service could be accessed by any number of front ends, each designed to suit its users' needs and capabilities, likes, dislikes, and budget. This is fundamentally no different than a computer manufacturer planning for users to use a wide variety of software, not all produced by itself. In the long run, this would make the search services more attractive to more users.

Summary

What has been attempted to be covered in this discussion can be summed up briefly. Computer software, regardless of the application, is something worked with and not merely consulted. Some of it is very complex. Some of it is not well suited to the needs and skill levels of its intended users. There is no reason why this should continue, at least not in the database search field, because it is relatively easy to separate the user interface from the main system.

People will use systems that are easy to use. A professional goal should be to make systems easier to use without trivializing them. A major way to do this, at least in the short run, is to design systems around specific user groups who share common professional jargon and use of databases, and to arrange for standardized mechanical interfaces with the major retrieval services. This approach has the added advantage of allowing for totally different points of view on how to search databases to be available to users.

The point is to give the users the choice of method and not reserve it for the distributors.

REFERENCES

Hoyle, N., & McNamara, K. (1987). Biblio-Link and Pro-Cite: The searcher's workstation. *Database, 10*(February), 73-78.

Marcus, R. S. (1983). An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science, 34*(6), 381-404.

Meadow, C. T.; Cerney, B.; Borgman, C. L.; & Case, D. O. (1989). Online access to knowledge: System design. *Journal of the American Society for Information Science, 40*(2), 86-98.

Newlin, B. B. (1985). In-Search: The design and evolution of an end user interface to DIALOG. In M. E. Willimas & T. H. Hogan (Eds.), *Proceedings of the 6th national online meeting* (pp. 313-19). Medford, NJ: Learned Information.

Niehoff, R.; Kwosny, S.; & Wessells, M. (1979). Overcoming the database vocabulary barrier—a solution. *Online, 3*(4), 43-54.

O'Leary, M. (1985). EasyNet: Doing it all for the end-user. *Online, 9*(4), 106-113.

Quint, B. (1986). Menlo Corporation's Pro-Search: Review of a software search aid. *Online, 10*(1), 17-25.

Ross, N. M. (1981). Newspaper databases. In M. E. Williams (Ed.), *Proceedings of the national online meeting* (pp. 415-20). Medford, NJ: Learned Information.

Whorf, B. J. (1956). In J. B. Carroll (Ed.), *Language, thought and reality: Selected writings* (p. iv). Cambridge, MA: MIT Press.

Witiak, J. (1986). Dialoglink: A review of DIALOG's search assistance software. *Online, 10*(6), 39-42.