APPLIED RESEARCH

SUMMARY
- ◆ Reports findings on user avoidance of the Help menu
- ◆ Identifies five characteristics that may make any help access method attractive to users
- ◆ Examines dialog-box help, pop-up help for tools, and several other help access methods

Fear and Loathing of the Help Menu: A Usability Test of Online Help

TREVOR GRAYLING

My company, MDL Information Systems, Inc., produces database-searching software for the pharmaceutical, biotech, and similar industries. We are also a supplier of various chemical databases. Our end users are mainly Ph.D. chemists who search commercial (and their own proprietary) databases for potentially useful chemical compounds. Because the databases are so large (perhaps 600,000 compounds), the chemists must develop complex and powerful search queries to retrieve a manageable number of useful database records.

Our software has a drawing component where users actually draw a molecule or a reaction. This graphic, which is drawn using the normal chemical conventions for atoms, bonds, and so on, becomes the search query. This query is then transferred to the database-searching component of the application.

Version 2.0 of our software has the typical Windows-style GUI interface; that is, an unordered collection of function-oriented tools and menu commands. Despite the software's complexity, the technical communications team working on the project had decided for a number of reasons that all end-user documentation would be in online help only for Version 2.0, with no printed manuals.

NEED FOR A USABILITY TEST

With "all our end-user documentation eggs in one basket," we understood that we had to build the help system correctly. Thus, we first conducted a thorough user analysis and task analysis. We also spent time designing the help system to avoid the typical complaints about online help (gathered from our surveys, the literature, product reviews, and our own experiences as end users):

- ◆ Information is missing (help is not comprehensive).
- ◆ I cannot find information (poor indexing).
- ◆ I always get lost in hyperspace (poor or ambiguous hyperlinks).
- ◆ I have difficulty manipulating the help and application windows at the same time.
- ◆ I prefer paper for long, complex material that needs to be studied.
- ◆ I don't like long, scrolling help screens.

We then built a paper prototype of a representative "slice" of the help system and conducted two preliminary usability tests. We first tested with six test subjects and then, after changes and corrections, with an additional five test subjects. These tests—with a vaporware version of the application, which was not yet available—were encouraging, and we felt confident that we could proceed with the online-only strategy.

The test also showed us that our users overwhelming preferred to use the index over the table of contents as the information-access tool, so we were careful to schedule several weeks of work to simply edit the index (which ended up containing some 2,000 entries).

As we approached the 2.0 release date, we knew that a usability test of the complete online help was essential: We needed to test the help with the now-completed application. We were particularly concerned to identify any missing information or missing index entries. We also wanted to see whether our design efforts had indeed eliminated or minimized the typical complaints about online help mentioned earlier.

Given our thorough design effort and paper prototyping, we felt fairly confident going into the test.

PARAMETERS OF THE TEST

How many test subjects?

After reviewing the literature and following a number of threads on the UTEST usability listserv, we determined that we needed to test at least eight actual end users to obtain results that we could trust. We subsequently found 10 test subjects (five "novices" and five "experts") from local companies who fit our user profiles. The *novices* had never used any of our products before, whereas the *experts* had used previous versions of our software. They were all screened for suitability: all were Ph.D. chemists with experience using Windows applications.

Manuscript received 11 April 1997; revised 25 November 1997; accepted 26 November 1997.

Emulating the user environment
In our test, we wanted to emulate the actual user environment as far as possible. Thus, our test subjects would:

- Work alone
- Have access to the database application (containing the online help) and a printer
- Complete a series of user scenarios (database-searching questions), which we gave to them one at a time
- Be given no hints or assistance of any kind (unless they encountered bugs)
- Receive an answer of "no comment" to any questions they asked

Because they had been invited to take the test by the MDL Technical Communications group, we were concerned that the test subjects would, in deference to their hosts, use the online help more than they would normally and hence skew the test results. Thus, we did not ask them to specifically use the help: We told them that we wanted them to use the help "only as they normally would." Drawing attention to the help in this way could also skew the data, but it seemed the lesser of the two evils. Because the test subjects subsequently ignored the help as much as possible during the tests, it was clear that they certainly made no attempt to humor or flatter their hosts in any way.

There was no time limit placed on the test and no expectation of the number of tasks that would be completed. On completion of a search, we informed each test subject of the correct number of database records that should have been retrieved. If test subjects had retrieved an incorrect number, we asked them to improve their search query.

The one "artificial" element we introduced that would not be present in the typical user environment was our encouraging them to verbalize their thoughts. We requested that they do this before the test, and we also reminded them from time to time during the test. Use of this "think aloud" protocol often enabled us to determine what the test subjects were trying to do and what they were thinking without having to constantly ask them or guess.

Given our thorough design effort and paper prototyping, we felt fairly confident going into the test.

Defining criteria for success
How would we know whether our usability test was "successful"? That is, how could we measure the "success" of the online help? We needed to determine some test criteria up front. To do this we asked ourselves, "What kind of help is of real importance to our users?" Because they were searching very large databases for compounds that were potentially very valuable to their companies, we determined that the users needed to be able—without outside assistance—to find all the help information that they needed to enable them to create search queries for the given user scenarios that would retrieve *all* those database records and *only* those database records that were chemically relevant to the given scenarios. Ph.D. chemists are not paid piece-rate, so we determined that time was not a factor, only the chemical relevance of the records retrieved.

We needed to determine some test criteria up front. To do this we asked ourselves, "What kind of help is of real importance to our users?"

Creating the user scenarios
The user scenarios obviously had to be based on the test criteria: Could the test subjects find the information that they needed to create search queries that would retrieve *all* and *only* the relevant records in large databases?

Because we asked the test subjects to use the help "as they normally would," we were worried that it might be used only lightly and that we would collect only a limited amount of data. Thus, we needed the scenarios to be sufficiently complex to make it more likely that the test subjects would need the online help. To do this, we looked for areas in the software that were either difficult for us to understand when we first saw them or were difficult to explain when writing the help. We also asked the Customer Service Department to identify difficult areas. Thus, it was easy to write scenarios that would contain difficulties that would in turn make it very likely that the test subjects would need the help.

Here is a typical user scenario:

Find all reactions in which one of the carbonyl oxygens on a reactant is replaced with any atom, but where the carbon atom that is ortho to the replaced oxygen is not methylated.

If you don't understand the chemistry, please accept that this task is typical: It is expressed entirely in our users' terms, it does not use any company jargon, and none of the terms it contains are found on any of the application menus. The scenarios were of "average" difficulty.

Conducting the dress rehearsal

Having conducted usability testing within our company since 1990, we knew the importance of a dress rehearsal. We recruited several in-house personnel and ran through an entire test. We wanted to:

- ◆ Check the user scenarios for inaccuracy and chemical ambiguity
- ◆ Determine that the test subjects do indeed need the online help
- ◆ Ensure that the application prototype could perform the tasks required by the scenarios without crashing
- ◆ Test the equipment
- ◆ Gauge the time needed
- ◆ Rehearse the roles of test supervisor and observer-recorder

Before the test, the test supervisor would work to reduce potential stress in the test subject. During the test, this person would remind the test subject to continue to verbalize and also assist with recovery from any software bugs.

The observer-recorder would have no communication with the test subject. This person would take notes on test-subject behavior (such as the navigation paths chosen), index entries tried or looked for, and assumptions the test subject made about the help or the application. The observer-recorder would also note any inaccuracies and typos in the help.

It is very natural to want to help test subjects who are having difficulties with your software, particularly when you know *exactly* what they need to do. It is also natural to feel frustration when a test subject takes a wrong navigation path through the help for the umpteenth time. Thus, for those technical communicators on the team who were new to usability testing, we also wanted to practice not giving verbal or body-language cues to the test subjects that might skew the results.

It is very natural to want to help test subjects who are having difficulties with your software, particularly when you know *exactly* what they need to do.

RESULTS OF THE USABILITY TEST

Given our dress rehearsals, the upfront paper prototyping, the careful design, and so on, we felt fairly confident in going into the tests with real end users. Here's what actually happened.

How users tried to get work done

Confronted with a searching scenario like the one outlined above, test subjects scanned the various menus, tried out the tools, and peered at the dialog boxes. They were looking directly to these elements of the user interface for inspiration: How did the chemistry terminology of the scenario relate to the application jargon shown on the numerous menus and tools?

What we did not expect were the extreme lengths to which our test subjects would go to avoid using the Help menu.

This type of "trial and error" behavior was expected. In 1993, we had conducted a survey via telephone of 51 of our end users. In that survey, 55% of the users indicated that they used trial and error as their first strategy in learning how to use our ISIS applications. Only 29% indicated that they would first try the manuals or online help. (The remaining 16% first relied on in-house assistance via help desks, colleagues, training classes, and so on.) Because the users we had surveyed knew that they were talking to the authors of the manuals and online help, we thought that the results could be skewed in favor of the manuals and online help. Thus, we assumed that the percentage who actually used trial and error may be even higher.

This bias toward trial and error behavior conforms to Carroll and Rosson's (1987) notion of the "active user" who, even as a novice, will jump right in and attempt to tackle the task at hand, armed only with previous experience of other software systems, which may or may not apply to the software they are using.

What we did not expect were the extreme lengths to which our test subjects would go to avoid using the Help menu. Carroll and Rosson do suggest that users are—paradoxically—biased against using manuals or online help to learn the very computer application that will enable them to complete their tasks. However, we had assumed that as soon as users did run into significant trouble—after first attempting a trial and error strategy—they would cheerfully change strategies by pulling up the Help menu and browsing the help index.

In reality, this was apparently the last thing on their minds. After carefully reviewing all the tools, menus, and dialogs, and not finding any commands or functions that would seem to help them with the scenario task at hand, the test subjects would sigh (audibly), and then start back at the beginning, reviewing all the menus, tools, and dialogs again. The verbalized thoughts of the test subjects at this stage indicated that they were still looking directly to these user-interface elements for assistance. *That is, they preferred to revert to the as-yet-unsuccessful trial and error strategy rather than go to the Help menu and review the very material that was designed to assist them.*

What the test data shows

Because of their reluctance to use the online help, test subjects often simply guessed at what they needed to do, with predictable results: Only occasionally did they perform a successful search on the first attempt.

For the 10 test subjects, we had three searching scenarios available. Thus, the test subjects potentially could have successfully performed a total of 30 searches. However, because of the difficulties in figuring out what to do or where a particular function was located, only 23 searches were actually completed in the total time available for each test subject (2.5 hours). Of these 23 successful searches, only 7 were completed on a test subject's first attempt; 2 were completed on a second attempt; 6 on a third; and 8 on a fourth. These results are recorded in Table 1.

Six of the seven queries that were formulated correctly on the first attempt were created by experts, a result that is not unexpected. However, apart from those, experts often performed as poorly as novices.

TABLE 1: TEST DATA

Number of Attempts at Search Query	Number of Search Queries Correct	Percentage of All Correct Queries
1	7 of 23	30%
2	2 of 23	9%
3	6 of 23	26%
4	8 of 23	35%

What user attitudes toward the online help were observed

When test subjects were told that the number of records they retrieved was incorrect, and hence that their search query was wrong, they needed to correct it. It was sometimes only as a result of repeated failures that test subjects finally turned to the online help.

When they did finally go to the help for assistance, their behavior was remarkable.

They read help information hastily They misread even the simplest instructions. Where a procedure might state, "Double-click the XXX object," the test subjects would single-click instead and then stare at the screen and wonder why the application was not behaving as they expected.

Because of their reluctance to use the online help, test subjects often simply guessed at what they needed to do, with predictable results . . . .

They made poor and inaccurate hyperlink choices We had gone to great lengths to ensure that the hyperlinks on topic screens were clearly differentiated from each other and were non-ambiguous. However, test subjects glanced over the links very quickly with seemingly little thought and simply clicked links that "might" be right but frequently weren't.

They bailed out early If they didn't find information within two or three screens, or if they didn't feel they were "on the right track," they would simply bail out of the help, even without the information they needed. They then went back to desperately browsing the menus and tools again.

They didn't look at "general information" We had created an informative "Overview" topic and had even written a comprehensive "How to Use Help" topic. These, and similar screens, were ignored completely.

Their whole attitude seemed to be: "I don't want to be here! I don't want to read all this information that you have so carefully designed and written!"

Because I don't wish to reopen the sporadic debate of paper manuals versus online help, I will note that manuals, being just another medium for imposing a learning burden on the user, typically fare no better than online help. That users resist reading manuals is widely noted (Norman 1988, p. 191; Carroll 1990, pp. 5, 8; Rettig 1991). A recent usability test involving both online help and manuals revealed identical user hostility and impatience with both (Ness and Butler, 1997).

On the positive side, the test subjects:

- Did use dialog-box help (context-specific help screens available via a "Help" button on a dialog box)
- Did use pop-up help for tools (our homegrown version of Microsoft's tooltips, where a small panel of help information provides a brief explanation of a tool or icon)

However, the testing showed that our work in both of these areas was entirely inadequate to the test subjects' needs.

What the test results reveal vis á vis the criteria

The key criterion was that test subjects would be able to find the help information they needed to create search queries that would retrieve all and only the relevant database records. They were able to achieve this goal *but usually only after being told that their initial search query was incorrect*. If we hadn't told them, they would have assumed that their initial queries were correct. We found this result worrisome because it means that many users must often use the application in an ineffective and inefficient way.

The key criterion was that test subjects would be able to find the help information they needed to create search queries that would retrieve all and only the relevant database records.

The second issue—eliminating or minimizing the typical complaints about online help—was successful overall. When the test subjects finally, grudgingly decided to look in the help in a careful and methodical manner, they were able to find the information they needed (for the test scenarios at least), they found that the index was adequate, and so on.

However, the "success" of the online help was by now a moot point: The major problem was that our test subjects wanted to avoid looking at it at all costs. Was this a sampling aberration? Was our help file *that* bad? Were our test subjects strange and untypical?

CONFIRMING OUR DATA

I talked with Jared Spool, whose company, User Interface Engineering, Inc. (UIE), does usability testing of computer applications on approximately 400 end users each year. Over the last few years, they have gathered an impressive amount of data on user behavior. Spool confirmed that our findings were not an aberration:

- Users go to the Help menu only as a last resort.
- They bail out of help early if a "clue" is not seen in the first one or two screens.
- They are reluctant to even scroll down a help screen.
- "Overview" and similar material is ignored.

Spool also noted that:

- Dialog-box help is used.
- Pop-up help for tools is used.

Spool pointed out that this data is from *their* tests and that others should not assume that results will be the same for every audience. However, we found that UIE's data matched our test subjects' behavior exactly.

For MDL's Technical Communications group responsible for the online help, general depression set in. We had spent over a year carefully designing and writing something that our users didn't want to read. Not only that, but our entire existence as technical writers seemed in jeopardy: Why would companies employ people to write material that customers will go out of their way to avoid?

OUR RESPONSE TO THE DATA

With the release imminent, we didn't have time to reinvent our careers or redesign the application user interface. After fixing the typos, errors, missing index entries, and so on that the test had uncovered, we decided to concentrate our efforts on the dialog-box help. This help component had been used heavily, and we had clearly seen how it could be improved.

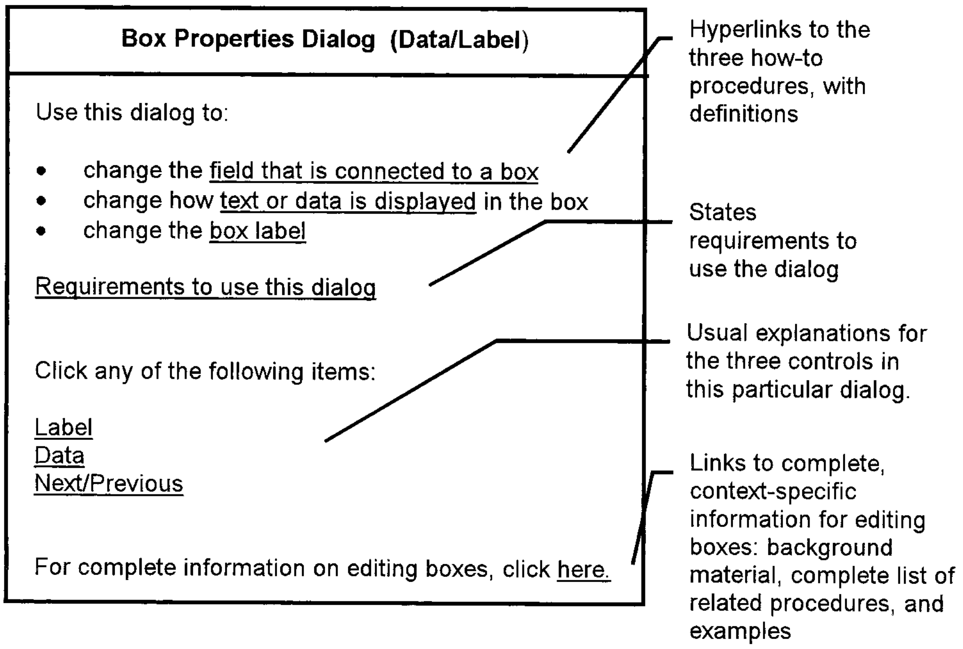
Reinventing dialog-box help

For dialog-box help, we had followed the typical model: We had provided explanations of each of the controls in the dialog. For example, if a dialog box had two lists from which the user needed to pick selections, plus three fields in which the user needed to enter text, we would provide explanations of these five items in the dialog-box help. This was of course essential, but our test had shown that test subjects accessed the dialog-box help for four other distinct reasons:

- *Definitions:* What is this dialog used for? What do these terms mean?
- *How-to information:* Can I use this dialog to do task X? If so, how do I do it?
- *Preconditions:* What preconditions do I need to meet, or what state or mode do I need to be in, to use this dialog?
- *Examples:* I want to see examples that use the functionality of this dialog.

That users approach an interface with a variety of different kinds of questions in mind, each of which needs an appropriate response, was also noted in a study by Sellen and Nichol (1990).

We had spent over a year carefully designing and writing something that our users didn't want to read.



Box Properties Dialog (Data/Label)

Use this dialog to:

- change the field that is connected to a box
- change how text or data is displayed in the box
- change the box label

Requirements to use this dialog

Click any of the following items:

Label
Data
Next/Previous

For complete information on editing boxes, click here.

Hyperlinks to the three how-to procedures, with definitions
States requirements to use the dialog
Usual explanations for the three controls in this particular dialog.
Links to complete, context-specific information for editing boxes: background material, complete list of related procedures, and examples

Figure 1. Help screen for the Box Properties dialog box.

As the mock-up of the dialog-box help for the "Box Properties" dialog box shows (see Figure 1), we incorporated the above information (underscore denotes a hyperlink).

For trial and error users, the redesigned dialogs now provided one context-specific, alternative route into the help that avoided the Help menu entirely.

Although we did not have sufficient time to formally test the new design, we did receive positive feedback from in-house users, particularly from the Customer Support representatives who used the online help to learn about the application.

The Customer Support Department uses the help extensively when responding to phone calls from users. Except when dealing with configuration, installation, printer and other issues not covered in the online help, Customer Support often plays the role of an online-help reading service, reading help to customers who never thought to read it themselves.

FUTURE DIRECTIONS

Our test subjects avoided the help menu but *did* like dialog-box help and pop-up help for tools. The latter two types of help share two characteristics that are not found on the Help menu:

- ◆ The help is *context-specific:* You see *only* the small part of the total information that is directly relevant to the command, tool, or part of the user interface on which you are focused.
- ◆ The help is *easily available:* It is just one, simple mouse click away. There is no need to rummage through huge help files and their (usually) incomplete indexes to find information.

These concepts are also identified by Bowie (1996), who notes that accessibility (that is, easy availability of help information without searching) is a major criterion in reducing the information burden on the user, and by Spool (1996), who states that the observed relative user acceptance of "Hint" buttons over the Help menu in usability testing results from their "context" and the low "cost of finding information."

For trial and error users, the redesigned dialogs now provided one context-specific, alternative route into the help that avoided the Help menu entirely.

Thus, we want to find methods of accessing help information that share these characteristics. Our hypothesis is that users would respond well to any access method into the help information that is both context-specific and easily available.

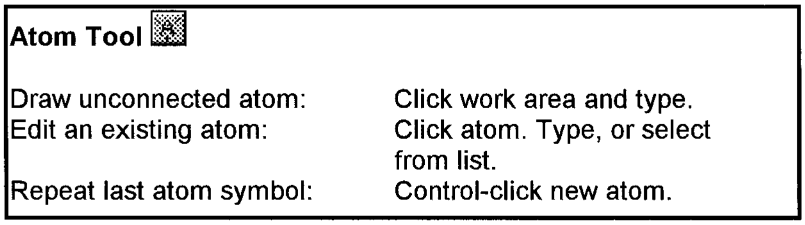

Atom Tool	
Draw unconnected atom:	Click work area and type.
Edit an existing atom:	Click atom. Type, or select from list.
Repeat last atom symbol:	Control-click new atom.

Figure 2. Current pop-up help for the Atom tool.

Providing pop-up help for tools/icons

The first access method into the help information that we want to improve is the pop-up help for tools. The drawing component of our database application uses many drawing tools, which are shown as a series of icons. Two characteristics noted by Zetie (1995) for what he calls "inline help" (pop-up, balloon, and status-line help) are:

- ◆ The help is *obvious to invoke:* If it is provided only on demand, the mechanism for invoking it must be obvious to the user.
- ◆ The help is *useful:* It must provide useful information. We extended this characteristic to include the requirement that the help must provide *all* the information relevant to the specific context. (Compare this to the "context-specific" characteristic noted above, which requires that the help provide *only* the information relevant to the specific context.)

Our usability testing revealed that we had failed to incorporate both of these characteristics.

Currently, the pop-up is invoked when the user presses *and holds* the mouse on a tool icon. For example, if the user presses and holds on the atom tool, the pop-up shown in Figure 2 is displayed.

Only 2 of the 10 test subjects actually found this help. The mechanism for invoking it (pressing and holding the mouse button over the tool) was not obvious and was unconventional.

The two test subjects who did stumble on the pop-up help (along with other test subjects from several other, more informal tests) liked it but found it too rudimentary to be useful. They needed much more information.

To correct these two problems, we are considering two changes. The first is to invoke the help simply by moving the mouse over an icon and hesitating briefly there without pressing any key. Because new users often hesitate, they should discover this help quickly. On the other hand, it is important to set the time delay such that, under normal circumstances, the help is *not* invoked and that the user must consciously hesitate over the icon to invoke it. Help that invokes itself, without the volition of the user, is potentially intrusive and irritating (Sellen and Nichol 1990). A well-known example of help that invokes itself without user volition is Macintosh balloon help in which a small text panel (shaped like a speech "balloon" in a comic strip) appears whenever the user moves the mouse over or near an interface object such as a tool or icon. The overall effect is that numerous balloons pop up, uninvited, as the user moves the mouse about the interface in the process of completing some task. A survey found this approach to be too intrusive and irritating (Rosenberg and Friedland 1994).

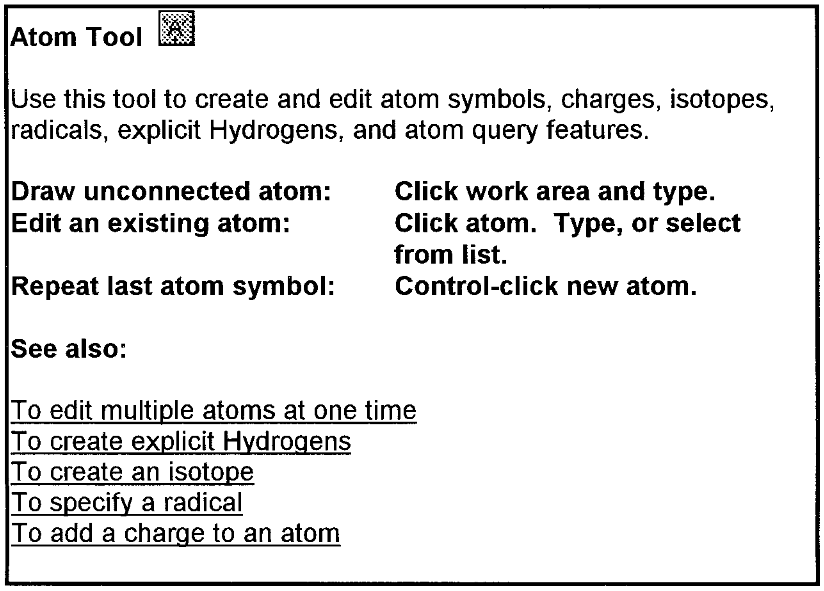
If online help is perceived to be intrusive and irritating, we can surmise that users will be less inclined to use it and may well even look for ways to disable it. This is clearly counter-productive. We can thus identify another characteristic of online help that is necessary to maintain users' positive (or at least neutral) feelings:

- ◆ The help is *non-intrusive:* It is invoked only when the user requests it. It does not distract users' attention from their work before being invoked.

If online help is perceived to be intrusive and irritating, we can surmise that users will be less inclined to use it and may well even look for ways to disable it.

The second change to our pop-up help for tools that we are considering is to provide all the information relevant to the tool. For example, the brief pop-up help window shown in Figure 2 could be expanded to provide this context-specific information, via hyperlinks into the help file (see Figure 3).

This "pop-up help with progressive disclosure of all the necessary context-specific information," at the user's request, and at the point on the screen of the user's focus of attention (in this case, a drawing tool) should be useful. It has the two characteristics that were attractive to our test subjects (context-specific; easily available); it also incorporates the characteristics that Zetie (1995) noted (obvious to invoke; useful); and it is non-intrusive.



Atom Tool

Use this tool to create and edit atom symbols, charges, isotopes, radicals, explicit Hydrogens, and atom query features.

Draw unconnected atom:	**Click work area and type.**
Edit an existing atom:	**Click atom. Type, or select from list.**
Repeat last atom symbol:	**Control-click new atom.**

See also:

To edit multiple atoms at one time
To create explicit Hydrogens
To create an isotope
To specify a radical
To add a charge to an atom

Figure 3. Proposed pop-up help for the Atom tool.

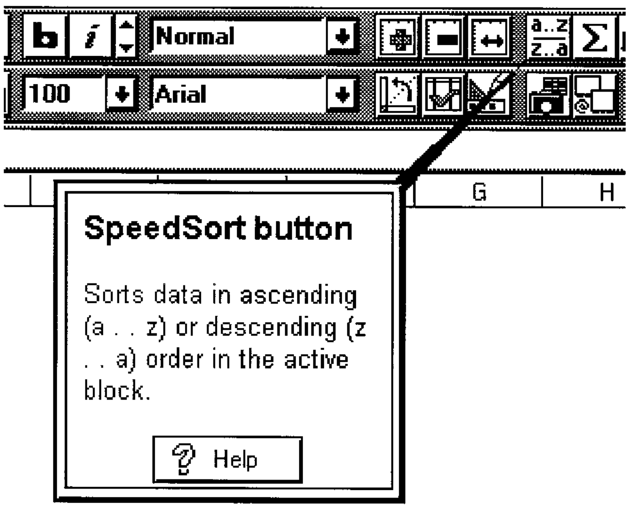
We can summarize the key characteristics for our proposed pop-up as follows:

- ◆ Context-specific—Yes
- ◆ Easily available—Yes
- ◆ Obvious to invoke—Yes
- ◆ Useful—Yes
- ◆ Non-intrusive—Yes

Another form of pop-up help for tools with progressive disclosure, called ObjectHelp, appears in the spreadsheet application *Quattro Pro 5.0*. An example is shown in Figure 4.

Here, users CTRL-click a tool to see an initial brief description of the tool (in this example, the "SpeedSort" button). They then click the help button within the pop-up to see a help screen that contains additional context-specific information.

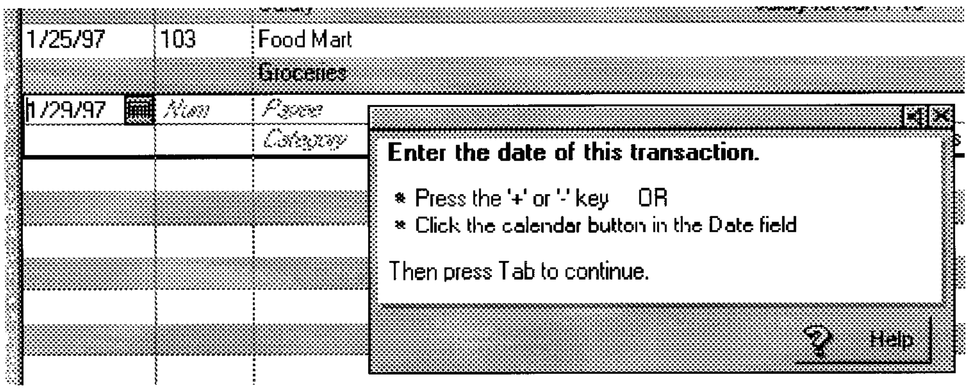
The design was usability tested with 10 real end users, who "loved the link to how-to help topics associated with



SpeedSort button

Sorts data in ascending (a . . z) or descending (z . . a) order in the active block.

Figure 4. ObjectHelp for the SpeedSort button in *Quattro Pro 5.0.*


1/25/97 103 Food Mart
Groceries
1/29/97 Num Payee
Category

Enter the date of this transaction.

* Press the '+' or '-' key OR
* Click the calendar button in the Date field

Then press Tab to continue.

Figure 5. Pop-up help with progressive disclosure in the *Quicken* check register.

the button" according to Robert Hipps, one of the designers and testers of ObjectHelp. ObjectHelp is clearly context-specific and easily available. It is also useful: The help screen accessed via the help button in the example contains the procedure for using SpeedSort, plus links to reference material and practical tips on sorting data.

However, the *QuatroPro* developers chose to invoke ObjectHelp via CTRL-click because they knew from their survey that help that is automatically invoked (as in Macintosh balloon help) was too intrusive (Rosenberg and Friedland 1994). Because it is an unusual and non-standard operation, CTRL-click would not seem to be obvious to invoke. The designers attempted to overcome this problem by displaying a message

SpeedSort button. (Ctrl + right-click for ObjectHelp.)

on the status bar whenever the cursor is moved over the associated button. In usability testing, the designers first had the test subjects work through a computer-based tutorial, which included information on how to invoke the ObjectHelp. As a result of this preliminary coaching, it is not entirely clear whether this help is truly obvious to invoke.

This struggle to make ObjectHelp "obvious to invoke" and yet at the same time "non-intrusive" highlights the fact that these two characteristics can sometimes conflict and must somehow be balanced. The difficulty is increased when we observe that, of the five key characteristics, these two retain a degree of subjectivity. For example, what is obvious to invoke to one user may not be obvious to another. The advantage of objective characteristics is that you can either measure or predict them: Ease of availability can be simply measured; and for context-specificity and usefulness, the information requirements of a particular situation can be closely estimated by a skilled technical communicator. However, to have confidence in conforming with the two more subjective key characteristics (obvious to invoke; non-intrusive), usability testing may be required.

We can summarize the key characteristics for ObjectHelp as follows:

- ◆ Context-specific—Yes
- ◆ Easily available—Yes
- ◆ Obvious to invoke—Unknown
- ◆ Useful—Yes
- ◆ Non-intrusive—Yes

Quicken, the extremely popular personal-finance application from Intuit, Inc., provides a similar form of pop-up help with progressive disclosure of context-specific information for fields in their Check Register. Brief, context-specific help appears in a small window for each check-register field. The window is invoked automatically when the cursor is placed in the field. A "Help" button in the window provides access to additional, context-specific information, as shown in Figure 5.

It is context-specific and easily available, and it incorporates Zetie's (1995) requirements (obvious to invoke and useful). Whether users find this window intrusive or not is unknown. We can summarize the key characteristics for *Quicken* pop-up help as follows:

- ◆ Context-specific—Yes
- ◆ Easily available—Yes
- ◆ Obvious to invoke—Yes
- ◆ Useful—Yes
- ◆ Non-intrusive—Unknown

Adding help topics to other menus

Another possible method for accessing the help information might be to add commands to menus that go directly to help topic screens. Because users spend a lot of time browsing through menus looking for assistance with their tasks (excluding the Help menu, of course), we can take advantage of this behavior and put help information where they will see it. For

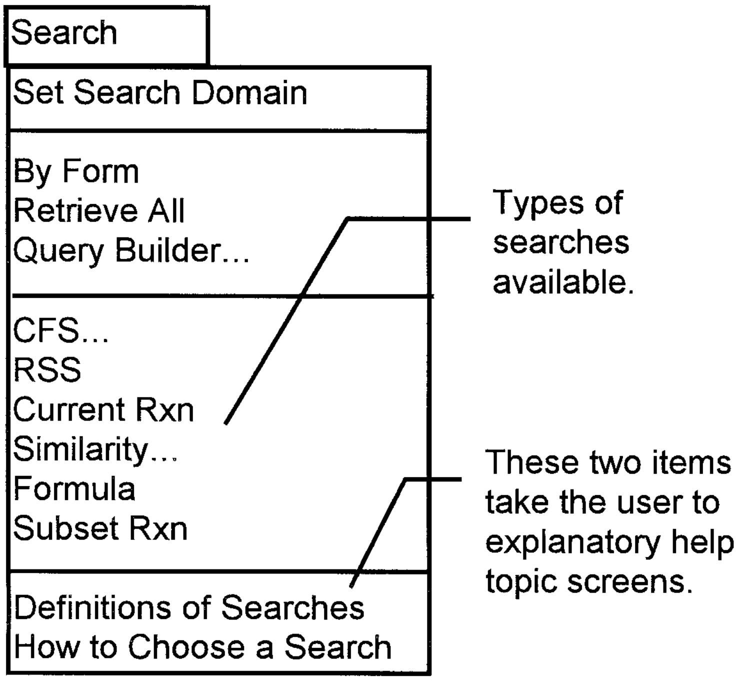
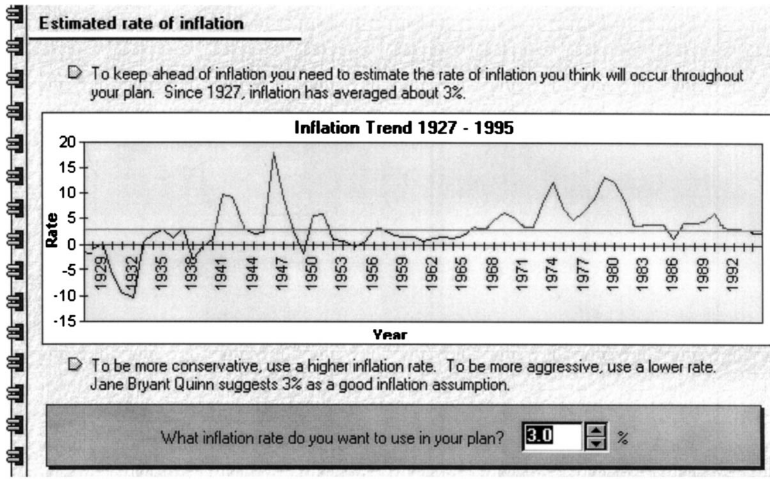


Figure 6. Context-sensitive help for menus.

example, our application has a Search menu that lists a large number of search types that the user can try. Each search appears on the Search menu in cryptic jargon with no visible definitions or explanations. We can easily add some help here, as the prototype in Figure 6 shows.

In this example, we add "Definitions of Searches" and "How to Choose a Search" to the Search menu. The first item would open a help topic containing all the search definitions, as well as links to procedures and all other search-related information. The second item would open a


Estimated rate of inflation

To keep ahead of inflation you need to estimate the rate of inflation you think will occur throughout your plan. Since 1927, inflation has averaged about 3%.

To be more conservative, use a higher inflation rate. To be more aggressive, use a lower rate. Jane Bryant Quinn suggests 3% as a good inflation assumption.

What inflation rate do you want to use in your plan? 3.0 %

Figure 7. Context-sensitive help in *Financial Planner*.

series of topics that help users to determine which type of search is best for their particular chemistry question. Like pop-up help, this method has the advantage of providing information at the focus of the user's attention. We can summarize the key characteristics of context-sensitive help for menus as follows:

- ◆ Context-specific—Yes
- ◆ Easily available—Yes
- ◆ Obvious to invoke—Yes
- ◆ Useful—Yes
- ◆ Non-intrusive—Yes

Using wizards and interviewers

Other alternative methods that we are considering for accessing help information are wizards and interviewers.

A *wizard* is an interface device (usually a series of dialog boxes) that leads the user, step by step, through a task that the user has identified, usually by choosing the task from a menu or index. A wizard thus assumes that users know the specific task that they need to perform.

An *interviewer* is an interface device that asks a series of questions of the user and then guides the user through a task.

Interviewers differ from wizards in that the user may *not* be sure of the exact task to be performed. For example, users working with tax software may know vaguely that they have to "do their taxes," but they may *not* know that they have to file Form 8606 because they are contributing to an IRA, even though the contribution itself is non-deductible. An interviewer helps to identify the *task* as well as the solution.

Figure 7 is taken from the interviewer in *Financial Planner* from Intuit, Inc. Here, the interviewer asks users to provide an estimate of the average rate of inflation for the duration of their financial plan.

This estimate has major implications for a financial plan. At this critical decision point for users, they are provided with help information that beautifully illustrates the key characteristics mentioned above. The help is obviously context-specific. It is provided directly on the screen and could not be more easily available. It does not even need to be invoked by the user. The help information is certainly useful: It is provided both as a mathematical average and as a graph; and it suggests how users can be less or more aggressive in their estimates. It is also non-intrusive in that, by virtue of using an interviewer, the user is requesting some form of help. We can summarize the key characteristics for wizards and interviewers as follows:

- ◆ Context-specific—Yes
- ◆ Easily available—Yes
- ◆ Obvious to invoke—Yes
- ◆ Useful—Yes
- ◆ Non-intrusive—Yes

CONCLUSION

Although we would always like to see more data, usability testing indicates that help information for Windows-style applications will frequently be ignored if it relies on the Help menu as the primary access method. Through their observed behavior, test subjects displayed a highly negative attitude toward help information provided in this manner. On the other hand, test subjects did like and use dialog-box help and pop-up help for tools. The hypothesis that we draw from these two strikingly different behaviors is that users will respond to help information that is simultaneously context-specific, easily available, obvious to invoke, useful, and non-intrusive. When designed carefully, dialog-box help, pop-up help for tools, help topics on menus, and help information in wizards and interviewers can all exhibit these key characteristics.

As technical communicators, we thus need to assist each other by developing, testing, and publicizing these—and other—alternative access methods that avoid the Help menu in bringing help information to our end users. TC

ACKNOWLEDGMENTS

Many thanks to current team members Nicki Davis, Frank Ripp, and Kimberly Steele, and former team member Cate Larsen for the hard work and enthusiasm they poured into the usability testing on which this article is based. Many thanks also to the anonymous reviewers and review coordinator whose behind-the-scenes work provided many insightful comments and suggestions that improved the article immeasurably.

TRADEMARKS AND SERVICEMARKS

REFERENCES

Bowie, J. 1996. "Information engineering: Using information to drive design." *Intercom* 43 (May):8.

Carroll, J. M., and M. B. Rosson. 1987. "The paradox of the active user." In *Interfacing thought: Cognitive aspects of human-computer interaction*, edited by J. M. Carroll. Cambridge, MA: MIT Press/Bradford Books, pp. 80–111.

Carroll, J. M. 1990. *The Nurnberg funnel: Designing minimalist instruction for practical computer skill*. Cambridge, MA: MIT Press.

Ness, K. B., and S. A. Butler. 1997. "Testing the usability of software documentation." *Intercom* 44 (October):19.

Norman, D. A. 1988. *The design of everyday things.* New York, NY: Doubleday.

Rettig, M. 1991. “Nobody reads documentation.” *Communications of the ACM* 34, no. 7:19–24.

Rosenberg, D., and L. Friedland. 1994. “Usability at Borland: Building best of breed products.” In *Usability in practice*, edited by M. E. Wiklund. San Diego, CA: Academic Press, pp. 261–292.

Sellen, A., and A. Nichol. 1990. “Building user-centered on-line help.” In *The art of human-computer interface design*, edited by B. Laurel. Menlo Park, CA: Addison-Wesley, pp. 143–153.

Spool, J. M. 1996. “Tips and Hints.” *Eye for design* 3 (May/June):5.

Zetie, C. 1995. *Practical user interface design.* London: McGraw-Hill International (UK). p. 230.