Developing Online Application Help

The primary goal for an application help system is to provide the capability for the end user to get useful help information and get back on task as quickly and successfully as possible.

by Dex Smith

Nearly 2000 online help topics are shipped with the HP MPower product. Throughout the HP VUE 3.0 and HP MPower projects, we've learned a lot about online help and its role with application software.

This paper describes online help and covers many of the issues encountered by application developers and authors. First, we outline the purpose of online help and suggest a use model. Second, two common information models are described, including how each relates to the domain of online help. Next, we discuss what is perhaps the most challenging and controversial topic in online information—navigation. Although this paper won't go into the details of these debates, it does reference some of the concerns and issues. Finally, we examine the roles of developers in a typical project, and we look at how these roles and online help systems may change in the future.

Although much of this material was developed during the design of the HP Help System and includes examples from that product, most concepts and ideas can be applied to any help system. The details of the implementation of the HP Help System are described in the article on page 79.

The Role of Application Help

To understand the purpose of online help, we begin with a prioritized list of goals, called a goal hierarchy.[1] Our goal hierarchy for online application help consists of the following:

1. The user leaves the help system.

2. The user is satisfied with the help information.

3. The application's response to the user's request for help is appropriate.

4. The help information is appropriate, even if the system response is not.

5. The user can pursue additional information.

From the user's perspective, the first two goals are clearly most important. The user wants to get out of the help system and back to work. Obviously, we want the user to return to work with useful information.

To application developers, goal 3 is an important reminder that help is part of the application. A well-designed application will use everything at its disposal (current modes, recent user actions and errors, and other contexts) to determine the best possible response to a user's request for help.

Goal 3 is also important to designers of a help system. It emphasizes that the design must provide the flexibility and power for application developers to integrate online help that is capable of responding as the user expects.

The fourth goal represents the help author's obligation to design and deliver the most appropriate information. It further acknowledges that there may be limitations in the help system, or in the application's use of system features, that the author may need to work around in designing the online information.

Finally, with the fifth goal, if the user's need is not immediately met, the system must allow the user to seek additional information.

All of these goals can be summarized in a single prime directive for an application help system—get the user back on task as quickly and successfully as possible.

The Application Help Use Model

Since the overriding principle for providing online help is to get the user back on task, online help must be easily accessible within whatever applications the user has chosen to accomplish the task. To that end, online help is cast in a supporting role. That is, the help system's job is to function as part of the application with the purpose of making the user successful with the rest of the application.

Keeping the help system tightly coupled with the application improves the user's *perceived proximity* (i.e., keeping the user feeling close to the application). If deviations to get help don't take users far from their current context within the application, they are more likely to stay on task and be productive. Avoiding unnecessary context switches is a fundamental human factors guideline.

The software architecture used to implement the help system doesn't necessarily have to be part of the application. The HP Help System's toolkit allows software developers to embed the help system within the application or to create a standalone help server.

If the help server model is used, special attention must be given to window manipulation and user interface design, so that the user still perceives the help as part of the application.

Handling Help Requests

In its simplest form, the interaction of a help system is a request/response transaction model. A request is made that

represents a need for information, and the system responds by providing information that meets the need.

For HP Help, we have categorized help information retrievals using the terms automatic, semiautomatic, and manual as follows:

- Automatic help. The application determines what help to present and when to present it. For example, an error message is automatic help because the application automatically provides information to the user. (This is sometimes called system-initiated help.)
- Semiautomatic help. The application determines what help to present based on the current context in the application. The user determines when information is presented by making a request for help. The de facto standard gesture for this kind of request is pressing the **F1** key.
- Manual help. The user requests a particular type of help. In this case, the user determines what kind of help and when to deliver it. Most manual help requests are made by choosing a help command from the application's Help menu.

Entry Points into Help

When a help request is made (either by the user or automatically by the application), the system responds by displaying a help topic. Each topic that can be displayed directly as the result of a help request is called an *entry point*. That is, if there is at least one way to get directly from the application to a help topic, then that help topic is an entry point into help.

Any single topic may be an entry point from more than one context within the application. The redundancy of overloading entry points in this way can often be advantageous to the user.[2] For example, a topic on entering numeric data may be appropriate help for many places within a database application.

Styles of Entry Points. Our goal hierarchy and use model has led us to classify two styles of entry points for help topics:

- Quick help. This style presents a focused topic for the current context, intended to meet the user's need and then to be immediately dismissed.
- General help. This style presents a help topic for the given request with the understanding that the user is likely to want additional information.

Comparing the two styles is like contrasting an information booth with a consulting office. At an information booth, contexts are well-defined, questions are generally short, and answers come quickly enough that there's no need to sit down. In seeking the services of a consultant, however, the expectations are different. You want to make good use of your time to learn, see examples, and explore related subjects so you don't have to come back.

An entry point into help is the first thing a user sees when a help request is made. The presentation, user interface, and information content set the user's expectations about how specific the information is to the current context and how quickly it can be dismissed.

For most applications of even moderate complexity, a combination of both styles of entry points is appropriate. The next two sections describe each style in more detail.

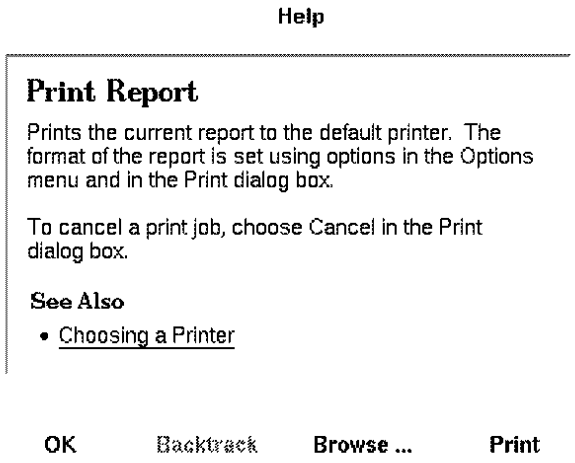

Fig. 1. A sample quick help dialog from the HP Help System.

Topics that are not entry points can be reached only by navigating from other topics or by other means within the help system. Therefore, the organization and relationships between help topics are important to designing appropriate entry points. The impact of the organization of help topics on navigation is discussed later.

Quick Help. Quick help entry points are optimized by writing style, presentation, and user interface to get the user back on task with minimal interruption. Quick help should have a very simple user interface and present easily consumable portions of information.

The quick help concept addresses the first goal in our goal hierarchy: User leaves the system. The second goal—User is satisfied with the help information—is up to the designers. That is, the correct entry point must be displayed and it must contain correct and useful information.

Quick help topics typically have a high level of context sensitivity and contain very specific information. By design, the content and presentation of quick help should not encourage the user to explore. Hypertext links (hyperlinks) should be used sparingly, and restricted to closely related topics.

Notice the Browse... button in Fig. 1. This is an example of how goal 5 in our goal hierarchy can be addressed for quick help. That is, the Browse... button provides a path for the user to seek additional information.

Quick help entry points are recommended for automatic and semiautomatic help requests. Some manual help requests may also use quick help, depending on what type of information is being requested. For instance, users expect minimal interruption when requesting "help on version" to get copyright and version information.

Typical uses of quick help include the following:

- Error messages, progress information, and feedback on user actions
- Context-sensitive help (invoked with the **F1** key or Help button in a dialog)
- Spot help (help on a particular item or area on the display)
- Application copyright and version
- Simple help on help.

General Help. General help typically covers the remaining entry points into online help. General help may have a more feature-rich interface for searching and browsing. Typical uses of general help include:

- Application overview
- A follow-up to quick help when quick help isn't enough (This use may follow one or more steps in a progressive disclosure sequence (described below).)
- Tutorials (Some tutorials may want to be more restrictive on navigation and prefer to use the quick help model.)
- Online documentation. (For browsing online information written and organized like a traditional manual.)

A sample general help dialog is shown in Fig. 2.

Progressive Disclosure

Even the best online help implementations are unlikely to provide entry points into help topics that meet every user's need for each possible situation.

So, what if a quick help entry point doesn't fill the user's need? Progressive disclosure sequences are designed to maintain a sense of proximity, while exploring a constrained path of topics that incrementally provide more information for the given context. (This concept is sometimes called progressive revelation or progressively more help.)

Often, a progressive disclosure sequence leads the user from initial "how to" context sensitive help through topics that are more motivational ("why") and eventually to detailed reference information. Generally, these sequences use a simple quick help interface for the first two or three progressions. At the point in the progression when the subject matter will likely lead the user to want to browse, the interface should provide additional browsing capability.

Fig. 3 shows how a quick-help dialog can provide a More button for walking through a progressive disclosure sequence. In each step in the sequence the dialog is reused, until the last step when the More button is renamed Browse and leads to a general help dialog.

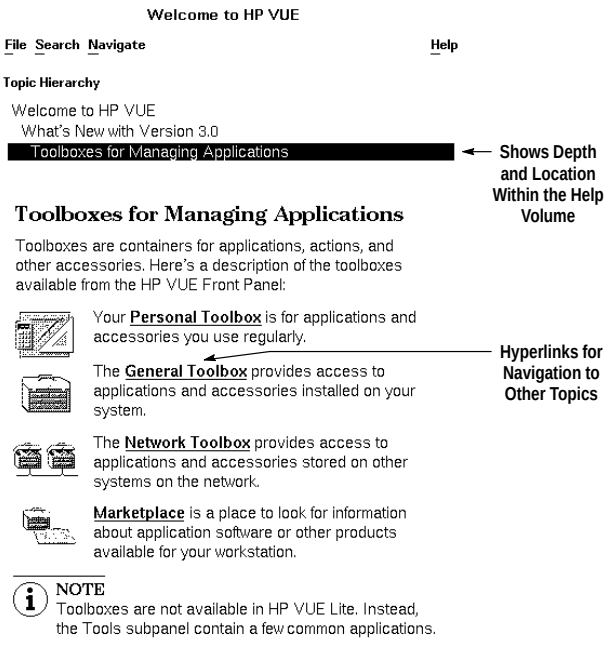

Fig. 2. A sample general help dialog from the HP Help System.

Information Models for Online Help

In designing online help for an application, it is important to organize the information in a way that best fits the application. A tight coupling between the application and its help information typically makes the information more useful when it is needed.

The help topics associated with an application are collectively called a *help volume.* Essentially, a help volume is like a book or document for the application. (However, we avoid the terms "book" and "document" because they tend to skew expectations. Further, some help volumes may have very little in common with their paper counterparts, such as a table of contents, chapters, and so on.)

Typically, there's a single help volume for each application. For very complex applications, or applications with distinct user groups, multiple volumes may be appropriate (Fig. 4).

Standalone Help Volumes. Not all online help is directly associated with an application. Help for the operating system, for example, isn't associated with any single command, program, or utility. Therefore, the information models for online help must allow for standalone help volumes.

Since standalone help volumes are not associated with an application, there must be a way to access them. One method is to provide a help manager. A help manager is simply an application that treats one or more volumes as its own and provides navigation into each volume. A help manager may provide entry points into help volumes that are also accessible from applications as shown in Fig. 5.

Implicit Relationships. Within a help volume, there are two significant organizational models: hierarchical and nonhierarchical. Most technical information is organized hierarchically like a traditional manual, in which chapters contain sections, sections contain subsections, and so on. Relationships between nodes of information are created implicitly based on the hierarchy created by the author. Fig. 6a shows a typical topic hierarchy.

Hierarchical navigation tends to give users more confidence in knowing where they are. There is a sense of depth (number of levels from the top) and movement (up, down, lateral). Hypertext links can be added to connect nonadjacent, but related topics (Fig. 6b). In this case, hypertext is perceived as a bonus, because it provides jumps that span the hierarchical structure.

Explicit Relationships. The free-form nature of a nonhierarchical information web allows authors to organize their information into hierarchies, circles, trails, webs, and grids. Fig. 6c shows an information web. Relationships are created explicitly with hyperlinks. (In some systems, end users can create their own links to connect related topics.)

When navigation is based entirely on hypertext, authors have much more flexibility to create any type of structure. This level of freedom requires a higher level of skill on the author's part. Links to and from each node must be designed and tested carefully.

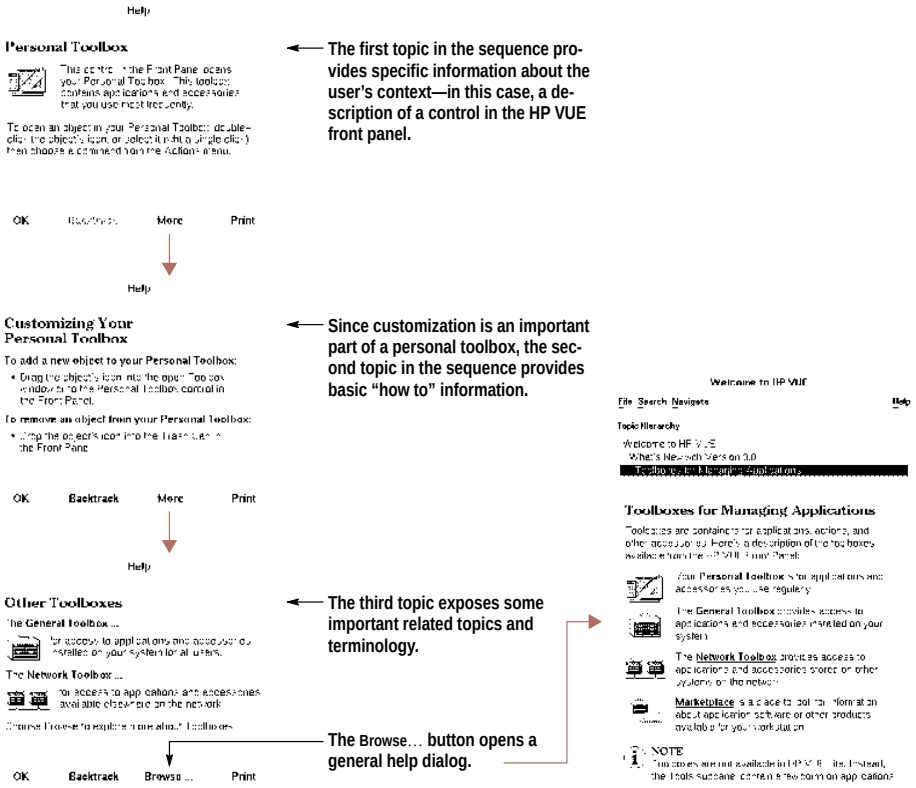

Fig. 3. A progressive disclosure sequence.

Hypertext Links. The lightning speed of hypertext links, and their ability to jump a user to a completely different area of the online information, leaves most users feeling lost after only a few jumps.

System designers and authors must recognize that each hypertext link is an invitation to explore, and therefore, also an invitation to get lost. Most research in this area indicates that hypertext is best used in combination with other navigation aids—"Hypertext is a spice, not a main course."[3]

Usability testing on the HP Help System indicates that when hyperlinks are used to navigate down the topic hierarchy, other links that span the hierarchy should be distinguished in some way. In HP MPower help, most help volumes use a convention of labeling lists of "Subtopics" and "See Also"s to make that distinction.

The HP Help System provides the following features to help users avoid getting lost in hyperspace:

- Backtrack command for undoing a hyperlink and going back to the previous topic
- Topic hierarchy display (in the general help dialog) to show "you are here" information within the current help volume
- History dialog that lists the titles of the topics recently visited
- A "home topic" at the top of each hierarchy to give users a known place to return to if they get lost.

Fig. 4. The association of help volumes in applications. Most help volumes are associated with a simple application.

The Developers' Tasks

There are many facets to developing an application with online help. Depending on organizational structure and the size of the project, these tasks may be performed by a variety of team members.

Planning and Design. Planning and designing a help system for an application requires close collaboration between designers, authors, and programmers. First, the design team must understand the user's need for the application and the tasks that users will perform. From this information a use model can be developed that describes the user's interaction with the application.

As a user interface is designed based on the use model, every effort should be made to keep things as obvious as possible, limiting the need for online help. Horton defines obvious as "... designing products that users already know how to operate ... or can learn by trying things out."[3]

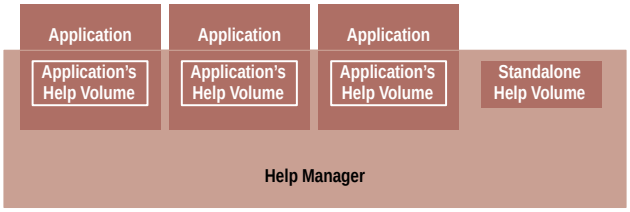

Fig. 5. A help manager provides access to all help volumes.

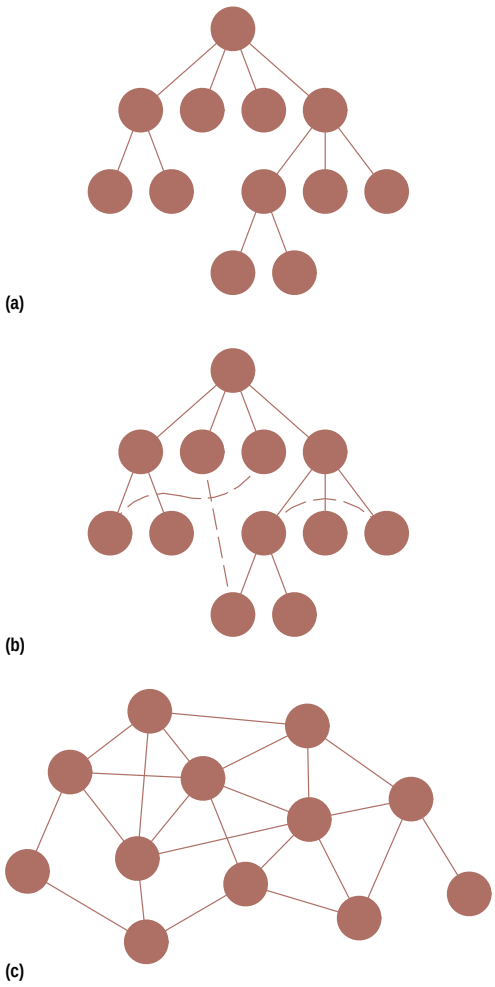

Fig. 6. Help volume organizational models. (a) A typical topic hierarchy. (b) Hypertext links provide shortcuts to span the topic hierarchy. (c) An information web.

For every context within the application that is not completely obvious, identify it as an entry point into help. Assign a unique identifier for connecting the application context to a help topic. Most help systems have an identifier naming scheme. In the HP Help System an identifier can be any string up to 64 characters including letters, digits, and hyphens.

Authoring. The author's job includes writing all of the help topics, assigning the corresponding identifiers along the way. For each help topic, the author must consider the many ways the topic may be read. For example, a topic that is typically seen as an entry point (displayed directly when the user requests help) may also be part of the browsable topic hierarchy. The topic must be authored to work well in both contexts.

The breadth and depth of exposure that an author has to the product during the writing process provides an opportunity to discover design flaws that may decrease usability. It continues to be the author's responsibility to reduce the need for online help by exposing these kinds of problems.

Programming. The software developer is responsible for providing the hooks from the software into online help. This requires using the agreed upon topic identifiers for displaying topics when the user requests help.

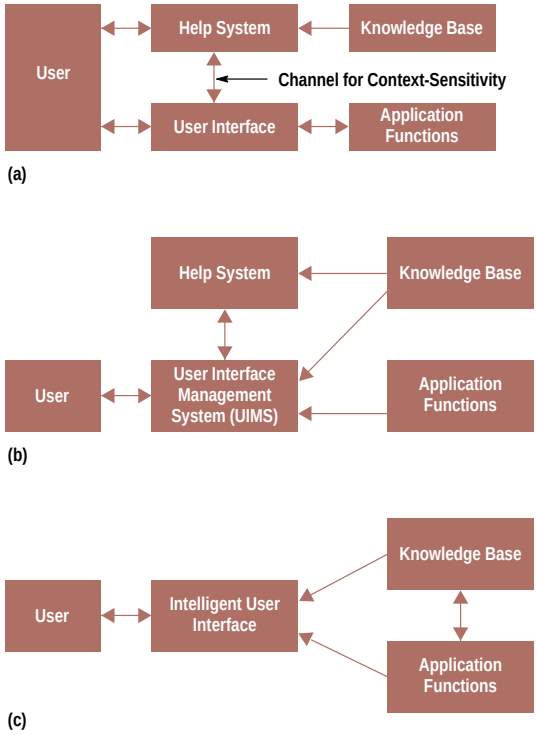

Fig. 7. Application help architectures. (a) The traditional application help architecture. (b) A possible migration of the traditional application help architecture. (c) Another possible migration of the traditional application help architecture.

Depending on the help system being used, there may be some additional programming necessary to achieve the desired behavior. For instance, with the HP Help System, progressive disclosure sequences like the one shown in Fig. 3 require the application to provide the correct behavior for the More button that walks through the sequence.

The Future of Online Help

Much of this paper has assumed a traditional approach to online help development. That is, one in which the help system and its supporting information are aligned with the application's user interface and supporting functions. Context sensitivity is enabled by direct connections between the help system and user interface components and states within the application. Fig. 7a shows a traditional application help architecture.

Reference 4 describes a help system whose design is driven by the same knowledge base that drives the user interface (via a user interface management system, or UIMS). In this model, the flow of information and functionality is all handled through the UIMS which presents the user interface for the application and the help system (see Fig. 7b).

Online information systems of the future may evolve from architectures like the one suggested by Foley and colleagues.[4] However, I expect help systems to meld further into the application, to the point that the help system itself isn't identified as a discrete component.

As this takes place, the engineering process for the information and knowledge portions of products will be as important as the traditional hardware and software engineering tasks. In

fact, the information architecture will become increasingly difficult to distinguish from the application architecture.

An intelligent user interface will draw upon an information base and the application functions to present user interface components (see Fig. 7c). The user interface may still present what the user perceives as online help, but it is generated from a combination of knowledge and application functions, along with the user's current context. Further, there will be a strong relationship between the knowledge base and the application functions (a channel missing in Foley's model).

Conclusion

Providing online help has grown in recent years from a "nice to have" feature to an expected component of all significant application software. For some products, online help reduces support costs because it puts information where the user is more likely to find it (meeting the goals in our goal hierarchy). For other products, providing online help also reduces the cost of printing hardcopy manuals.

As applications continue to grow in complexity, so does the need to simplify them for users. For the foreseeable future, online help plays an important role in helping with that simplification. Today, online help systems are discrete components that aid application developers in packaging easier-to-use products. In the future, online help will become a by-product of an information engineering process that addresses all information aspects of a product.

Acknowledgments

Gathering and refining of the information presented in this paper wouldn't have been possible without the contributions of many people throughout HP during the HP Help System project.

Richard Artz, Anna Ellendman, and Jodi Peterson were among the dozens of talented learning products engineers that contributed valuable design review and direction for online learning products. Jay Lundell and Jan Ryles offered insightful human factors review and testing.

Of course, the engineers who designed and built the HP Help System contributed the most important part—a great product. Thanks to Lori Cook, Brian Cripe, Axel Deininger, Steve Hiebert, and Mike Wilson. Also, thanks to Bill Kaster, loaned to us from Corporate Learning Products, for codeveloping the HP HelpTag language and compiler. Project managers Bob Merritt, Bob Miller, and Rick McKay also helped to foster a creative environment by welcoming cross-functional and cross-organizational collaboration.

References

1. N. J. Belkin, and P.G. Marchetti, "Determining the Functionality and Features of an Intelligent Interface to an Information Retrieval System," *Proceedings of the 13th International Conference on Research and Development in Information Retrieval* (SIGIR '90), September 1990, pp. 151-178.

2. J. Price, "Creating a Style for Inline Help," *ConText and HyperText—Writing With and For the Computer,* MIT Press, pp. 311-323.

3. W. Horton, "Let's Do Away With Manuals Before They Do Away With Us," *Journal of the Society of Technical Communications,* Vol. 40, no. 1, January 1993, pp. 26-34.

4. J. Foley, W. Kim, S. Kovacevic, and K. Murray, "UIDE – An Intelligent User Interface Design Environment," *Intelligent User Interfaces,* Addison-Wesley Publishing Company.