

Bringing Help to the Forefront:



By NICOLETTA A. BLEIEL, Senior Member

Strategies to Increase the Usability of Your Software User Assistance and Your Product

Online Help became a required component in software applications in the early 1990s, driven in part by Microsoft's "Designed for Windows" certification process. Help was generally integrated into the product from the Help menu, with context-sensitive Help buttons on dialog boxes and "What's this?" field-level help rounding out the list of ways users could access and interact with the information.

After learning how to develop and design Help, technical communicators have been developing and discussing ways to bring it to the forefront ever since—not because of vanity, but because we realized its potential to make users (and therefore our products) more successful.

Embedded Help (in this context, I am specifically referring to an embedded pane in the user interface, or UI) is widely acknowledged as one of the best ways to integrate Help within the interface, but it has the reputation of being difficult to develop and execute. Without the ability to completely control the information and the mapping of that information to the interface itself, technical communicators faced an implementation that was expensive and therefore hard to sell internally. In addition to implementation issues, an embedded Help pane could add another online deliverable to the documentation mix—increasing the product footprint—and that time could be spent on other user assistance offerings.

But let's put all that aside and take a look at one way to elegantly implement and map embedded help. With proper planning and design, one Help file can be developed to work as embedded assistance and as a stand-alone project.

The Case for an Embedded Help Pane

According to Galitz in *The Essential Guide to User Interface Design*, a typical user interaction with documentation involves three steps: "(users) need to find it, understand it, and apply that understanding to solve the task at hand."

Best of all users don't see embedded Help as Help.

Online Help is passive, with the user taking the initiative. Embedded Help is nonintrusive and does not require the user to take the initiative, says Lynne Hall. "Embedded Help sidesteps the user's reluctance to go to the help system, providing background and often invisible support to users."

Embedded Help also encourages learning, because, according to Mike Hughes, "users shift from a learning/exploration mode to a task execution mode ... they stop exploring and experimenting." Embedded Help can delay that shift.

An embedded, dynamic Help pane displays relevant information as the user navigates the interface, making the concepts and language of the interface obvious to the user. It's task-specific, context-specific, doesn't require users

to look for it, and doesn't require them to ask the right question. Users continue their workflow and don't have to split their focus between two different windows, unlike traditional online help. As Cheri Zubak, one of the pioneers of embedded Help, has noted, "It's help that is designed as software." Best of all, users don't see embedded Help as Help (see DeLoach).

Combine embedded Help with an information-packed Start or Welcome page and you will ease users into the interface. This solution brings Help to the forefront and, at the same time, promotes the entire user assistance set and encourages its use. This will increase the usability of your documentation set and your product.

OK, OK, You've Convinced Me— Embedded Help Is Cool, But ...

I have discussed a number of reasons why embedded Help is an excellent way to increase the usability of your software application (actually, I have quoted some very knowledgeable people), but I haven't addressed the BIG question—how is it implemented?

Figure 1. The product interface with the embedded Help pane on the right. Note the variety of UA (traditional Help, videos, tutorials, quick reference guides, etc.) available from the Start Page.



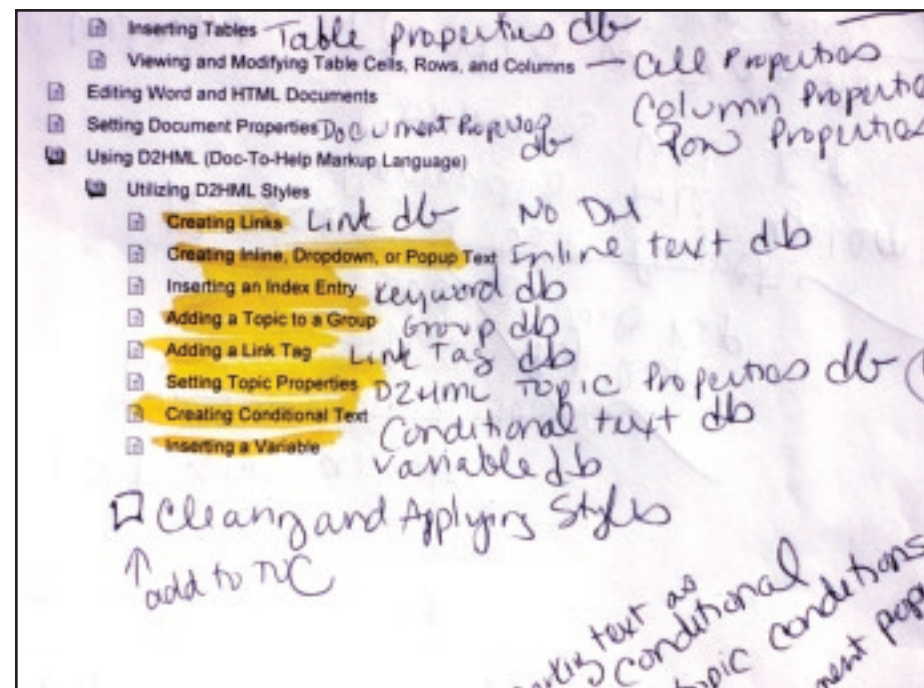
Traditionally, implementation has been challenging because the pane itself had to be coded, and the mapping of Help to the interface had to be coordinated between the software development and information development departments. A combination of the following file types and technologies could be used: Text and RTF files, HTML, DHTML, XML, style sheets, JavaScript, Jscript, ASP, JSP, PHP, VB, C++, and C#.

Once implemented, changes and updates could be time-consuming, and since traditional Help is still a necessary deliverable (and rightly so), two help files would still be needed.

My solution: I very nicely asked some extremely knowledgeable software developers at my company to build a control (a visual component that can be snapped into an interface and configured) to my specifications. One of my priorities was a mechanism for the information developers to control the mapping process. After an exchange of ideas, the control more than met my requirements, and it can be used in any application developed in Microsoft Visual Studio.NET. The Help file can be compiled HTML Help (a .chm) or web-based Help.

I mentioned above that traditional

Figure 2. A section of my draft TOC.



Online help is passive, with the user taking the initiative.

Help is still required. Why? Because the Table of Contents, Index, and Search are still necessary navigation aids and should be available if a user prefers to work that way. I created web-based Help for this project; web-based Help has a number of advantages over HTML Help, but one aesthetic advantage: I had greater control over the skin and could design it to match the product.

Designing User Assistance for Multiple Uses

You can use the same source files for embedded help, stand alone help, and a manual if you single-source it with a plan. It is efficient reuse and a time-saver, but it needs to be designed well.

First, choose the interface elements for which you want to provide embedded assistance. For example, if your application includes ribbons, you could choose

to provide a topic for each ribbon, break it up into one topic for each ribbon group, or even one topic for each button. You could even do all three. After you have made your decision, break down the topics you'll need to cover the entire UI. That is one set of topics. These topics should be conceptual overviews that link to more detailed task and reference information.

Another set of topics are the ones that cover the major functionality of the product. These topics will be the bones of your file and will most likely be the top-level "books" in your TOC.

Finally, you will need to list each dialog box that will require embedded help. You also have a decision point for dialogs—will you map the entire dialog box to a single topic or provide Help at the field level? Once you do that, you'll have another set of topics. Some of the dialog topics may double as "functionality" topics.

Remember, everything that you want to map to the interface must be a topic, so choosing to map to every toolbar button and dialog box field, check box, and radio button will break your Help file into very small pieces, and logic may suffer. Any information that can be logically grouped together should be.

Once all this information is gathered, you can put together a draft TOC. Every topic that will be mapped to the interface should be flagged.

This is a good time to sketch out the relationships between topics, which will be useful later for creating "see also" links. In addition to the "see also" links at the bottom of each topic, "breadcrumbs" are a great navigation aid for the top of each topic, since they indicate the user's position in the information hierarchy and, like the "see also" links they provide clues about where they should navigate next. The navigation tools on the embedded help pane provide another level of navigation, but they will only take the user back to topics they have already visited.

Mapping and Delivering Embedded Help

Mapping the Help to the interface was handled with a mapping tool that was

built into the control. All the mappings are saved to a single XML file that must be delivered to software development.

It was a five-step process:

1. Install the software application.
2. Drop the Help files in the folder specified by software development. I used uncompiled HTML web-based help, so I had multiple files; if I had created compiled HTML Help, I would have only needed to drop the .chm file.
3. Open the mapping interface using the key combination specified by software development.
4. Map the topics to the interface.
5. Deliver the Help files and the map-

Figure 3. The Help pane in three phases; with the mapping interface open and unmapped, mapped, then closed and ready for dynamic action.

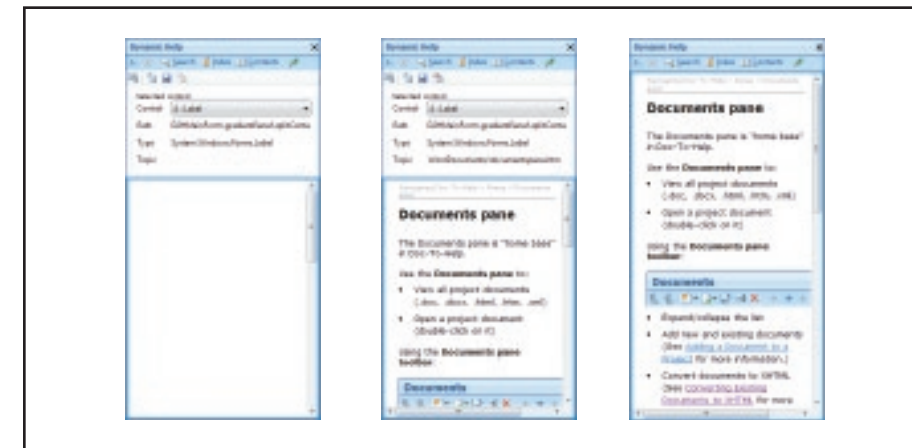
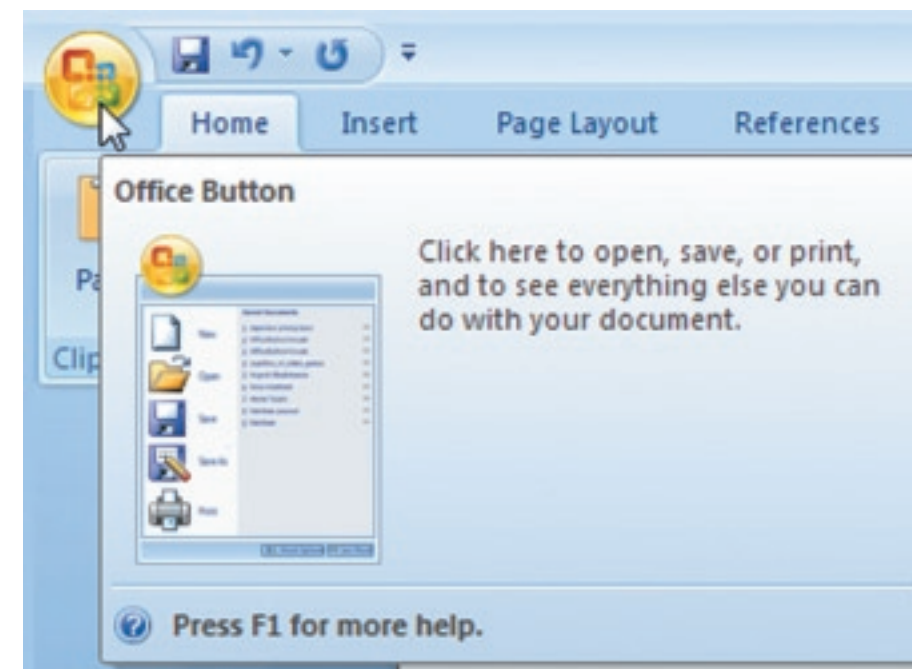


Figure 4. A Microsoft Word Expanded Tooltip.



ping XML file to software development for the install.

Integrating Other Deliverables, "Layering" User Assistance

Embedded, dynamic Help is just one piece of the deliverables set. Documentation needs to be layered to provide a user experience that serves all types of users and their learning styles.

In *Usability Engineering* (1993), Jakob Nielsen noted that because users have many uses for documentation, you might need three levels: short reference cards or job aids (i.e., quick reference guides), tutorials for learners, and traditional reference manuals for expert users. That is a good baseline, and now we can create many additional forms of user assistance that make it possible to engage visual, auditory, and kinesthetic learners, as well as different experience levels.

These deliverables include videos and podcasts, quick reference materials, and tutorials, as well as websites, company forums, blogs, wikis, Twitter feeds, and other Web 2.0 initiatives. With a little planning, these can be integrated into the interface—and not just into a Help menu (although if you have one, they should always be there, too). If your interface has a web-like design, incorporating them into the Start or Welcome page is an option. If it doesn't, perhaps streamline a bit by creating a webpage that incorporates all of these deliverables and link to it from a menu, or from the splash screen. Links to information about training courses, consulting options, upcoming conference exhibits, online newsletter subscriptions, and other useful materials aid users, even if they aren't documentation (and can increase revenue for the company). Links to all of these should also be included in the online Help.

One other embedded option to consider is Expanded (or "Super") Tooltips, which can be found in Microsoft Office 2007 and other products. As you hover over toolbar buttons and other parts of the UI, these tooltips display expanded information, (including graphics) and are formatted to be easier to read. If your product includes these, you should



Considerations for Embedded UA

- Understand single-source authoring and concepts. Take advantage of conditional text, variable text, and other options your help authoring tool (HAT) has to offer. For example, you'll need to optimize space in embedded topics, so incorporating expandable or collapsible text (information that displays only when a user clicks on it) is a great option.
- When designing your Help, choose colors, fonts, and styles that work with the user interface. This is always a good idea, but it is especially important for embedded UA.
- Work with software development to determine the best position in your application's UI for the embedded help pane. Galitz recommends the right side of the interface, in order to "minimize obscuring of screen content—in a window, position priorities are right, left, above, and below."
- Once users are comfortable with the software, they may want to turn embedded help off. Alan Cooper notes that the ability to configure an interface will make a difference between liking and disliking a program. Make sure users have the option to close the embedded help window in the main UI and dialog boxes.
- No topic should be a dead end. My recommendation: At the bare minimum, every topic should have a "see also" link to all of its subtopics, and every subtopic should have a link back to its parent.
- Avoid adding requirements just for Help, such as separate databases or plug-ins (see Mueller).

Once you have worked out the design, the writing goes very quickly, but the design process can sometimes be a lot like putting together a puzzle that is mostly sky.

be writing the text—if not, you should suggest expanded tooltips for your interface. They provide useful and timely information to users while they are exploring the interface.

Ever-evolving User Assistance

The constant evolution of user assistance deliverables is exciting because of the many options we have for tailoring the appropriate set for our product. And, with a little additional planning, that carefully crafted set can increase the usability of a software application. That is the ultimate goal, and it has been since online Help (and even manuals) was introduced. 📌

FURTHER READING

Cooper, Alan. *The Inmates Are Running the Asylum*. Indianapolis, IN: Sams Publishing, 2004.

Cooper, Alan. *About Face: The Essentials of User Interface Design*. Foster City, CA: IDG Books Worldwide, 1995.

DeLoach, Scott. *Best Practices for Embedded User Assistance*. WritersUA Conference, 2007, www.clickstart.net/presentations/embeddedua_writersua07.pdf.

Galitz, Wilbert. *The Essential Guide to User Interface Design*. New York, NY: John Wiley and Sons, 2002.

Hall, Lynne. "Performance Support: Online Help and Advisors." *International Encyclopedia of Ergonomics and Human Factors*. Boca Raton, FL, and London, UK: CRC Press, 2006.

Hughes, Michael. "Fattening the Long Tail Through Progressive User Adoption." *Cutter IT Journal* 20.4 (2007).

Mueller, Paul. *Exploring and Implementing Embedded Help*. STC Austin Meeting, 2007.

Nielsen, Jakob. *Usability Engineering*. San Diego, CA: Academic Press, 1993.

Zubak, Cheryl Lockett. "What Is Embedded Help?" *Intercom* (March 2000).

Nicoletta A. (Nicky) Bleiel (nickyb@componentone.com) is an STC Director-at-Large and a senior information developer at ComponentOne in Pittsburgh, PA. She has been a technical communicator for more than 14 years; with experience designing and writing for products in a variety of industries; including information development, media sales, industrial automation, simulation, and pharmacy. Nicky has delivered presentations at STC international, regional, and local conferences and meetings, as well as DocTrain, WritersUA, tcworld, and LavaCon. Topics have included embedded help, tools and technologies, user assistance design, single sourcing, wikis, Web 2.0, and convergence technical communication.