

TECHNICAL  
COMMUNICATION:  
THE NEXT  
WAVE

BY NEIL PERLIN  
*Senior Member, Boston Chapter*



New technologies seem to sweep over the field of technical communication in waves about ten years apart. In the early 1980s, word processors revolutionized content creation and began to shift our focus from content to tools and technologies. Starting in the early 1990s, *WinHelp* and the Web revolutionized content distribution and design and further shifted our focus toward tools and technologies. It's now the early 2000s. What's the next wave?

Predicting the future is fun, but it's easy to be wrong. Remember the glowing predictions for "server push" (the delivery of information initiated by the server rather than the user) a few years ago? But there are some technologies, designs, and issues that are likely to affect us as technical communicators in the very near future. This article introduces them, describes their likely effects, and tries to project what we'll need to know or do to use them.

### Confusion

One of the most significant issues we face is confusion over technologies, tools, and formats.

Some of the confusion stems from help technologies with similar names. For example, do the terms "*WebHelp*" and "Web help" refer to two different things, or is one a typo? Is "HTML help" a format or simply a shorthand way of saying, "help in HTML format"? This confusion can lead to selecting the wrong format, buying the wrong authoring tool (or buying the right one but using it the wrong way), or hiring the wrong people.

More confusion stems from the fact that tools like eHelp's *RoboHELP* and Forefront's *ForeHelp* are so widely used as to have become synonymous with the help formats. It's not unusual to hear references to "creating the *RoboHelp* file" or "distributing the *ForeHelp* project." This confusion often leads people to mistakenly think that the output of one tool must be completely reformatted to be opened in another tool.

Still more confusion results from the avalanche of other new technologies that *may* affect us. Is it important to know that WAP-compliant telephony standards use WML but DoCoMo iMode uses Compact HTML? Or that CBL uses SOX documents rather than true DTDs? Or that XSL and XLL are subsidiary standards

under XML? It's easy to get lost in this sort of alphabet soup.

You'll need to do two things to deal with this confusion:

- Continually train yourself. Unlike the early *WinHelp* days, it's no longer possible to be an expert on every new format or technology. But it *is* possible, and useful, to have a basic familiarity with the trends in our business in order to avoid being surprised by new developments.
- Take nothing for granted in project meetings; ask lots of questions. I've seen cases in which a client says "HTML help" and half the team hears "Web page" while the other half hears "that Microsoft format that replaced *WinHelp*." In a few cases, documentation groups have spent months creating the wrong format because of such misunderstandings. (Disaster stories are a lot more entertaining when they happen to someone else.)

### XML

XML, or eXtensible Markup Language, is probably the most significant new technology in years. XML surfaced in 1997, as HTML's limits were becoming apparent, and it offers several major benefits that are fueling its growth.

XML's best-known benefit is its support for custom tags. If you're building a Web site for a fast food chain, for example, it might be useful to have a <CHEESEBURGER> tag, conceptually the same as creating a twenty-seventh letter of the alphabet. Custom tags are impossible under HTML but integral to XML. A second benefit is XML's requirement for proper syntax. When you *know* that all your material is properly formatted, it's a lot easier to manipulate that material and have confidence in the results.

What's likely to affect most of us isn't XML itself but the Document Type Defi-

nitions (DTDs) and schemas that are being created under it. Think of DTDs and schemas as sets of grammatical rules, with different sets of rules for documents specific to different companies or industries. For example, e-commerce, the travel industry, and publishing each use one or more DTDs, with new ones appearing weekly. These DTDs let us create XML-compliant content without having to see the actual XML codes.

Technical communicators will probably fall into one of two groups when it comes to working with XML. A small group will consist of people who can work at the code level to create DTDs and schemas—effectively acting like programmers. A larger group will consist of people who use GUI-based tools to create content that adheres to those DTDs or schemas but who never see any actual codes. If this seems far-fetched, remember that only a few years ago, HTML was considered to be on the cutting edge of complexity.

What do you need to know about XML? If you'll be working at the code level, you'll find dozens of books and seminars on the subject. Even if you won't be working at the code level, it's a good idea to at least learn about the concepts through a seminar, book, or conference session so you'll understand what people are talking about. (You can find information about new XML tools in Alan Houser's article, "Trends in XML Software," on page 16.)

### Help

From 1990 to 1997, there was effectively one choice for online help formats—Microsoft's *WinHelp*. This changed in 1997 when Microsoft officially introduced *HTML Help*. The subsequent appearance of Sun Microsystems' *JavaHelp* and Oracle's *Oracle Help for Java* have only made the situation more complex.

*WinHelp* and *HTML Help* are both Microsoft-centric; the former requires

Windows and the latter requires *Windows* and *Internet Explorer*. (The U.S. Justice Department's suit over Microsoft's bundling of *Internet Explorer* with *Windows* slowed the spread of *HTML Help* because you could no longer be sure that a client had *Internet Explorer* and thus might not be able to run the help file.) *JavaHelp* and *Oracle Help for Java* are more platform- and browser-independent, but they have their own technical problems and come from an environment—Java—whose technology and concepts are unfamiliar to many *Windows*-based help developers and technical writers.

One new development has been the emergence of Web-based help. Often referred to generically as “Web help,” Web-based help is platform- and browser-independent and uses a variety of designs ranging from the *WinHelp* style (which offers the familiar navigation tools—a table of contents, index, and full-text search) to the totally new (eHelp's “WebHelp” format is a specific type of Web-based help). You can see a good example of *WinHelp*-style Web-based help at [www.octopus.com](http://www.octopus.com) (Figure 1). For a variant of this look, see [www.bigcharts.com](http://www.bigcharts.com), which has a table of contents and index, but labels and designs them very differently than *WinHelp* (Figure 2). Finally,

Figure 1. [www.octopus.com](http://www.octopus.com)



[www.centranow.com](http://www.centranow.com) is a good example of a completely unique style (Figure 3). For more analysis of Web-based help, as well as a closer look at [www.bigcharts.com](http://www.bigcharts.com), see Cheryl Lockett Zubak's article, “Developing Help for the Web,” on page 9.

What do you really need to know about Web-based help? You'll have to know how to use the authoring tools, but you'll also have to know how to conduct user analyses and needs assessments in order to come up with the design and select the *appropriate* authoring tool. This will probably mean choosing either a *WinHelp*-style format like eHelp's *WebHelp* or Forehelp's *InterHelp*, or a Webbish look like that of [www.centranow.com](http://www.centranow.com), which you can create using any Web authoring tool. In short, you'll have to know how to assess the environment and pick the appropriate tools.

A second development has been the search for new ways to present application help. “What's This?” help was supposed to have made it easier for users to get field-level help, but it has had mixed results. One new approach is “embedding”—designing the help as a part of the application interface rather than just opening it in a window plopped on top of the application. Microsoft's *Money* and *Picture It!* offer good, familiar examples of embed-

ded help. The problem with embedding is that it can be defined in many ways. Field descriptions and error window tips can also be considered embedded help, so you have to define “embedded help” before you begin to create it.

What do you need to know about embedding? First, you'll have to be able to work with the programmers at the planning stage of a new project to decide whether embedding makes sense. If it does, you'll have to be able to decide what it looks like, with very few precedents for guidance. (You can see one example, from UserFirst Services, by going to [www.userfirst.net](http://www.userfirst.net) and clicking “Demos.”) Finally, embedding is so new that there are no tools yet. You're going to have to know how to work at the code level.

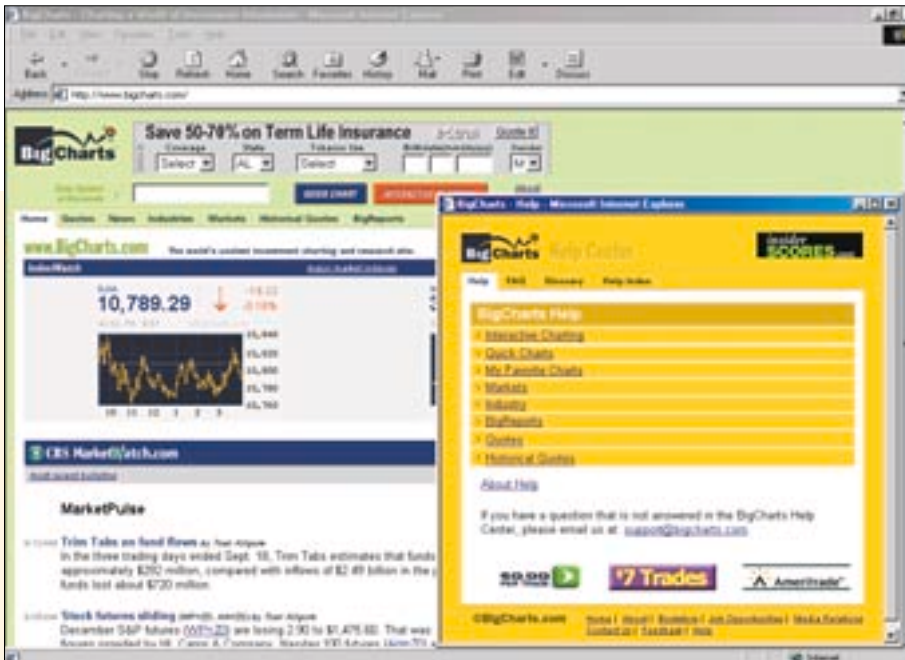
### Portable Information

Portable information has existed for years, starting with online books that people read on laptops in the 1980s. But several new developments in the past year have spurred portability.

One is the spread of personal digital assistants (PDAs)—devices like Palm Pilots and Pocket PCs. Though not created as platforms for online information, they support it well. Palm and Pocket PC-based online publishing tools have been around for several years, and Microsoft recently introduced a new format called *Reader* ([www.microsoft.com/reader](http://www.microsoft.com/reader)) that improves text resolution on tiny screens. And with Compact Flash cards that hold hundreds of megabytes of information, it's perfectly feasible to carry entire online libraries in your shirt pocket.

The other, explosive development has been the surfacing of wireless Web technology. Almost unknown in late 1999, it was a cover story in *USA Today* by early 2000. Officially, the term “wireless Web” refers to any form of wireless Web access, such as a laptop connected to the Web using a Web-enabled cell phone. Yet the recent excitement has focused on direct access using pocket-sized devices like cell phones and alphanumeric pagers.

These two technologies—PDAs and telephony-based devices like cell phones—are beginning to converge. For example, you can buy tiny keyboards to add onto several Ericsson cell phones

Figure 2. [www.bigcharts.com](http://www.bigcharts.com)

(see [www.igo.com](http://www.igo.com)) to type short messages without the agony of typing on a cell phone keypad. You can also buy clip-on modems for PDAs to access e-mail and the Web. New devices forecast to appear in the United States within a few years are supposed to integrate all these technologies into a single, unified package with greater speed and new features, such as full-motion video.

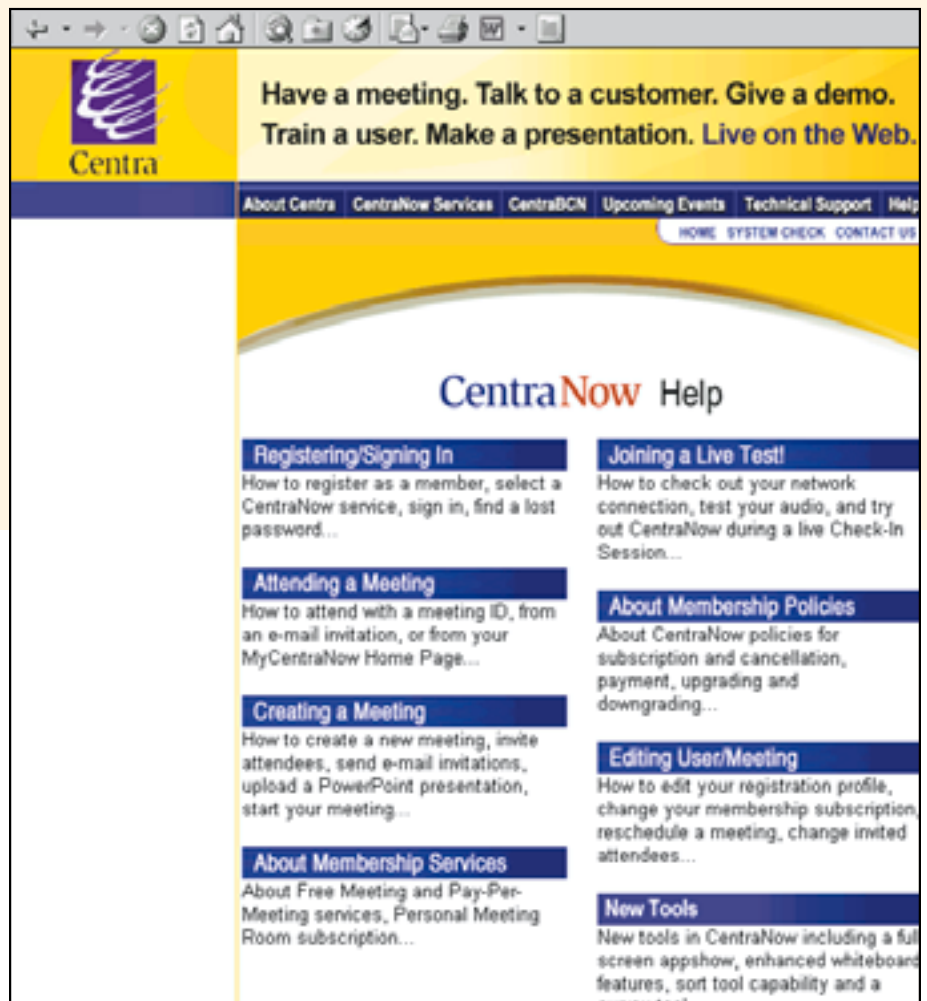
What do you need to know about portable information? Behind the hype are technologies at a very early stage of development. You'll have to know how to deal with evolving standards, configuration problems, and distribution problems...almost a repeat of what we went through with the Web in the early 1990s. You'll have to learn about new issues associated with the technology of telephony, such as different telephony standards, access in remote areas, and even battery life. Finally, you'll have to figure out how to create content that works on a four-line cell phone screen and, more challenging yet, content that works on both a four-line cell phone screen *and* a desktop monitor. Of all today's new technologies, portability presents some of the biggest challenges but also some of the most interesting.

### ASPs

ASP stands for application service provider. (Watch out for acronym confu-

sion here. ASP also stands for active server page, so make sure everyone saying "ASP" means the same thing.) ASPs are an extension of the ISP (Internet service provider) concept. While ISPs provide access to the Web, ASPs provide access to *applications* on the Web. These are typically applications that are too rarely used or too expensive for small companies to buy, such as tax software and B2B (business-to-business) exchanges, or applications that are more effective if all users can easily share data, such as calendaring software. The ASP concept has become widespread enough to earn its own nickname—"technology on tap."

ASPs face serious problems with access control and data security, but the concept has enough potential that it may overcome those problems. Many vendors, including Microsoft, are modularizing their applications for the ASP market. If this market takes off, many help systems

Figure 3. [www.centranow.com](http://www.centranow.com)

will have to be broken into modules and revised to match the application modules.

What do you need to know about ASPs? From a mechanical viewpoint, ASP-oriented help may not require new skills on our part, since help is help, whether it consists of one huge module or a hundred small ones. But you may wind up writing things other than traditional help, such as installation and deployment notes for the actual ASPs, which might require you to learn about Web servers, non-*Windows* platforms, Java servlets and applets, Java application deployment, and so on. And since B2B vendors are developing ties with ASPs as a way to reach small businesses, you may have to learn about B2B-style documentation. You may also have to get involved with the programmers at a much deeper level than usual in order to define the modules, which leads us to...

### Knowledge Management

The definition of “knowledge management” often depends on who’s doing the defining, but we can define it broadly as the process of finding, organizing, and packaging information for a variety of uses and users, with modularity as a by-product. In an online help project, for example, you manage knowledge by evaluating the material, deciding how to break it into modules (topics or pages), creating an outline of the modules, and assigning attributes to the modules. The irony is that we’ve been doing this for years; it just wasn’t considered glamorous because the documentation group did it.

What’s changed is the growing involvement of programmers, systems analysts, and consulting companies as knowledge management extends beyond documentation and into *content reuse*. In documentation groups, reuse basically means “single-sourcing”—using the same material for online and hard copy. But this idea has broadened in the past few years, first to “repurposing” and, starting in 1999, “dynamic content management.” Repurposing and dynamic content management both extend the concept of single-sourcing to include segmenting information into modules that can be manipulated for presentation on any platform or format.

What do you need to know about knowledge management? Any technical communicator with a few years’ experience or with classes in modular design or Information Mapping® is familiar with the subject, even if it’s under another name. The challenge now is twofold: First, to develop technical skills needed to deliver content on a wide range of platforms; second, to convince the project teams that we’re fully capable of performing knowledge management and advancing into dynamic content management and repurposing. How difficult this will be largely depends on how technical communicators are perceived within a company. In my experience, the solution has been to get as technical as possible in order to be taken seriously by the programmers, and to get involved as early as possible in any project.

### And on the Horizon...

Two interesting new technologies that *probably* won’t affect us for years are worth watching. One is database publishing—writing content in the form of modules that can be stored in databases and assembled into various forms through custom scripts. Often viewed as the technology that will put technical communicators out of business, database publishing is more likely to affect only leading-edge object-oriented help developers and writers of online documentation for B2B applications. The other new technology is digital paper. Basically a screen that looks like a sheet of Mylar film, it has been out in prototype form since early 1999. It’s too soon to predict how digital paper might affect us, but imagine what you could do with a distribution medium that blended elements of a screen and a sheet of paper. The August 2000 issue of *Wired* has a good article on the subject.

### Side Effects

What may come of these trends and changes?

We’re likely to see a reemergence of separate tracks for writers and techies in documentation groups. In the old hand-coding days of *WinHelp*, writers wrote content and passed it to techies for coding and compilation. The emergence of

tools like *RoboHelp* made this approach obsolete, since writers could do the writing and the technical steps. A few years ago, however, tasks like scripting sent the technical level back up, and the single-developer model began splitting back into separate tracks. Those tracks will undoubtedly come together as authoring tools catch up with new technologies, then diverge again. (For another look at the future of technical communication, see Lori Fisher’s article, “Technical Communicators: Designing the User Experience,” on page 21.)

Our increasingly complex environment will increase the need for continual learning. Continual learning has always been important, and it will become even more so as more technical communicators turn to contracting work. The more you know, the more flexible you are and the more employable you’ll be. However, you also have to realize that you can’t learn everything and still have a life. You’ll have to focus on areas that will help your career in the near future, whatever “the near future” means to you. This, in turn, means that you’re going to have to actively manage your career.

I’ll summarize very simply... If you like a job that provides intellectual stimulation, the next few years are going to be a blast.

### Acknowledgments

I’d like to thank Karen Bachmann of eTrango, Allen Beebe of Inforonics, Inc., Wendy Beren of Beren Associates, and Connie for reading this article and providing a reality check. ❶

---

*Neil Perlin has twenty-two years’ experience in technical writing, with sixteen in training, consulting, and development for various types of online documentation including WinHelp, HTML Help, JavaHelp, CE Help, and some now known only in legend. Neil writes columns and articles about online documentation, runs the “Beyond the Bleeding Edge” sessions for STC’s annual conference, and is a popular speaker for many professional groups. He provides training, consulting, and development for various forms of online documentation through Hyper/Word Services of Tewksbury, Massachusetts.*