

Help Is Dead. Long Live Help!

Paul Sisler, Catherine Titta

Help: Noun.

That little online book “thing” we put alongside a software application. Dead.

Help: Verb.

To assist someone through a task or process to achieve goals. Alive and kicking!

As Help Authors, we often treat online help as a “thing,” not an activity. We’ve favored the noun over the verb! This preference is natural for writers, who enjoy producing books. If we hope to survive on a dynamic development team, we must train ourselves away from writing books, toward helping people. This shift means examining the bigger picture and adopting different ways of working.

Evolving user expectations and technologies have put online help (little online books) in jeopardy. By changing our concept of the deliverables we produce today from those little online books to a more active, embedded help model we can not only survive change, but add real value to our organizations. Designing and developing information-rich interfaces with supportive learning environments is the future direction of help authoring.

INTRODUCTION

New technologies for applications development such as the World Wide Web, Application Service Provider (ASP) models, and cross-platform development mean fundamental changes in the way Technical Communicators view our work. They also mean new opportunities for Help Authors and represent a steady move communicators’ work toward programming, and programmers’ work toward communication.

While this shift is good for computer users, it is frightening for many communicators. The fact is, however, we have a great deal to offer software development organizations, not the least of which is an understanding of users and their information needs. Changes in software development today are opening doors for Help Authors to become integrated, indispensable members of software development teams.

HOW WE LOOK AT ONLINE HELP

As Technical Communicators, we invest our careers in creating and structuring information to help people understand complex systems and use complicated

products. We expend a great deal of effort trying to understand our audiences and their needs for information, training, and support. We painstakingly form and sharpen the content we believe will best meet our audiences’ needs. And we spend many long days in the office creating indexes, links or cross-references, tables of contents, and other points of access to the catalogs of procedural, conceptual, and reference information we’ve determined people need.

But the sad fact, which always horrifies us to realize, is that few people take our manuals off the shelf, and perhaps fewer explore our online help systems.

Many computer users don’t even realize that online help exists for most software applications, and those who do are unlikely to be aware of advanced methods for accessing help. Do we really believe that everybody knows about pressing the F1 key in order to view context-sensitive, topical help? How many times do we ourselves struggle to find the information for which we’re looking, even though we are among the most advanced users of online help systems?

It has become common wisdom, especially in marketing circles, that software users expect, even demand, some kind of online help with the products they purchase, regardless of whether they actually use help or whether they have a favorable attitude toward it. Thus, nearly all software vendors and IT department development teams currently view online help as a must-have for their products. Some of the most enlightened organizations even include Help Authors on their development teams—though they’re often expected to play the role of advanced note taker. In spite of all this effort, most online help systems are regarded as terribly inadequate.

HELP IS DEAD

For some time, it’s been the joke in many documentation departments to refer to manuals, quick reference cards and other print materials as “dead-tree ware.” But let’s face it, most online help systems aren’t exactly vibrant either.

Some pronounced help dead several years ago with Microsoft’s announcement that they’d discontinue further development of WinHelp in favor of HTML Help, now known as Help 2.0. And, in spite of the fact that many software publishers still support their Windows products with WinHelp, it is true that there is a trend toward obsolescence of traditional forms of online help authoring and delivery.

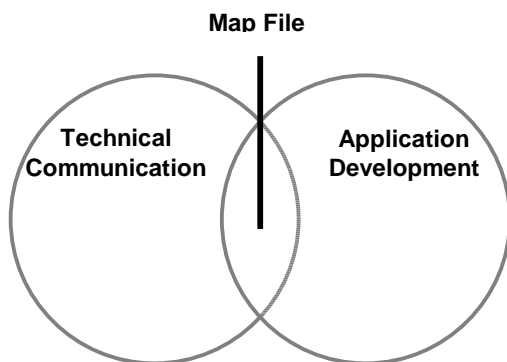
Technology and Obsolescence

A continuing problem for Help Authors has been that technologies for creating and delivering online help change faster than we're able to keep pace. If the shift for Windows Help developers from WinHelp 3 to WinHelp 4, from WinHelp to HTML Help weren't difficult enough, the demands of creating online help for web-based and cross-platform applications are certainly sufficient to frustrate even the greatest technophiles among us.

Users' Expectations. Fewer and fewer computer users require basic information covering such topics as the use of a mouse with each of our new releases. The Web has made computers so ubiquitous that most software users today are already familiar with the basics, leaving us free to concentrate on slightly more complex information in our help systems.

On the other hand, the Web presents us with a few problems: Users see flashy Web hypertext and they expect the same features out of online help. The problem is that the most well-established help technologies can't support the whiz-bang features for which users are often looking.

Organization Charts. When we sit in our cubicles corralled off from the development team, our power to make a difference for our users is diminished.



The more we “throw help over the wall” with a map file, the more we are trapped into the old patterns of work.

Authoring Tools. One of our greatest hurdles comes from the tools on which we've relied to create help. All of the help authoring tool vendors have made great strides in their product lines over the past few years, but at their core, most are primarily designed for use by Help Authors who create WinHelp, HTML Help, or systems with similar book-style features, such as JavaHelp.

For many of us, not only have authoring tools seemed to have had difficulty keeping pace with challenges of new

development requirements, but because they rely on out-moded ways of understanding help, we find them inadequate for our needs. This is not to say it's impossible to build online support for a web-based application with a familiar authoring tool, just that help produced with these tools often fails to integrate well with web site development tools, to meet cross-browser, cross-platform requirements, or to provide the just-in-time support contemporary users demand.

Many technological problems are easily surmounted. Recent versions of RoboHelp, for instance, include a JavaScript-based Table of Contents and Index for use with their web-based help solution, WebHelp. This change undoubtedly improves performance and compatibility and makes WebHelp a more attractive solution. Still, some authoring tool issues are far more related to a fundamental shift in the way we view help and a demand for a new type of online help that is better integrated with the user interface than they are related to technology.

A Help Paradigm Shift

While technological barriers are vexing, they are the least of the shortcomings of online help. When we say help is dead, we mean that our past and current conceptions of help lend themselves to a lifeless form of online help to which the great majority of computer users rarely turn for support.

Among the problems for help today are these issues:

- People must go out of their way to get help
- People don't get what they want when they do try to use help
- Help is often viewed as a means of fixing problems with poor product design
- Help is difficult to use
- Our task-oriented help is often mere “duh” help
- Help authors try to accommodate all users by being generic, rather than creating help that adapts itself to individual users
- Using help takes users away from reaching their goals

Where Task-Oriented and Context-Sensitivity Fall Short

An overarching deficiency of most online help is that its book-like nature interferes with the user's flow of work and interrupts his or her progress toward a goal. When we separate help authoring from application development, we create separate and distinct products: a tool and a manual.

If people have difficulty using our tools, they must put the tool down, grab a manual, find the information they

need, read and understand it. Then, they can return to work. (Rarely will anybody read our help systems except when he's in trouble.) When people get back to work after searching through our help systems, they must reorient themselves to the place they left in their workflow. Because this process is often fruitless and frequently disorienting, people rarely bother to use online help.

Task-based Help. Task-based help, the product of our analysis and authoring, is typically no more useful than a manual locked in a system administrator's vault where nobody but she can get it. One of the shortcomings of task-based help is that Help Authors cannot anticipate the step in a process at which a user may get lost, and our steps and procedures cover more territory than users are likely to need.

Another problem is more fundamental: Task-based help, while well intentioned, almost never amounts to more than a numbered list of interface features. When we write task-based help, and when we invite our team members to review our help, we often judge our product on the basis of whether we've covered everything. We end up with procedures containing steps that read something like, "1. Enter your name in the name field." Around the office, we call this "Duh" help, help that offers only the most superficial information. Even when we place this type of information in a procedure rather than setting it aside in a reference topic, it is still useless and irritating.

Context-sensitive Help. In spite of its promise to match the work users want to accomplish, task-based help usually fails to be helpful because it rarely does more than document the interface. Context-sensitivity offers its own promise—to put the right information in front of users just when they need it—but it too fails. Very few computer users actually use context-sensitive help. Partly, this is because the methods for accessing help are not immediately obvious. Even "What's This Help" buttons, which are an improvement over the F1 key, get little use.

When they can figure out how to get help, people find little consistency in what's presented to them. Sometimes its help with a procedure. Other times it's a description of valid entries for a field. Sometimes it's a dialog telling the user "No help is available..." This inconsistency discourages people from using online help. They are often further discouraged when they discover have taken valuable time to shift their attention from their goals to useless information and fruitless searches.

People Want to Reach their Goals

People don't want to complete tasks, they want to reach goals. You didn't have a phone installed in your home

because you wanted to use a phone: You wanted to talk to your family and friends. Help—even task-based and context-sensitive help—often focuses on tasks (on phone use), rather than what we actually want to do (connect with friends). For our help systems to be truly helpful, they must support, not interfere with, users' ability to reach goals.

We are moving in the right direction, but to push our efforts to the next level, we must push aside beloved book metaphors and refine approaches, such as task-orientation and context-sensitivity. To help people who use our software, we must create proactive, integrated support systems, and to do that we must become central players on software development and design teams.

LONG LIVE HELP!

When we envision a help industry, the goal of which is not to create online books but instead to build proactive support systems into software products, we see that help has a lot going for it. New technologies and means of doing business are opening doors to offer us unprecedented opportunities to build not just better software support systems, but better software.

What Does Help Have Going for It?

For many of us, used to thinking of ourselves as writers who simply happen to write about technical subjects, the thought of shifting our roles and becoming more involved in product development is more than a little intimidating. We're not engineers and we don't want to be. But the benefits of change are enormous and they're easily within our grasp.

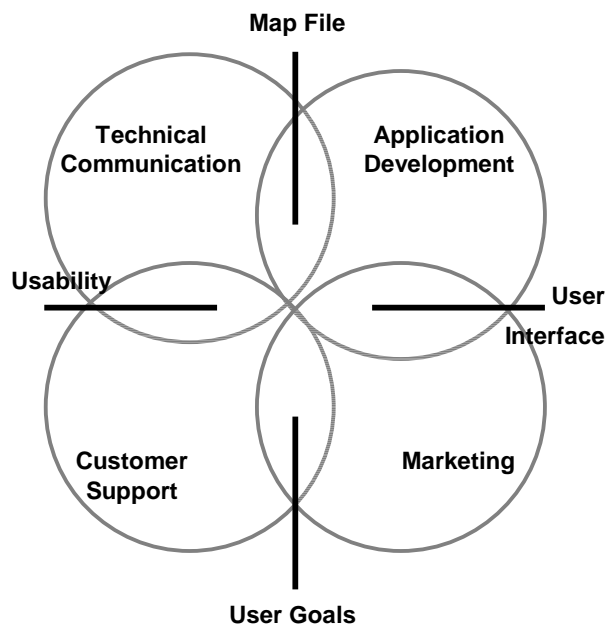
What We Already Know and Do. Consider web-based applications development. While Technical Communicators are often left out of design and development in traditional applications development, because the web was first devised as a system document delivery, we're the first people our clients and bosses turn to when it comes to web development.

Technical communicators are well suited to taking the lead in many of these development ventures, because many of us specialize in areas related to this work:

- Programming
- Database Design
- Information Architecture
- Performance Support
- Usability
- Graphic Design
- Interaction Design
- Instructional Design
- Process Design

New Ventures. A strong economy and rush to build new businesses on new media allow Technical Communicators to expand our responsibilities and further explore related design, development, and usability skills and knowledge. New e-commerce companies, and e-commerce ventures within larger organizations don't come with established development practices, giving Technical Communicators a chance to define their role. In these organizations Technical Communicators can find themselves driving new product development.

Organization Charts. Getting involved in related organizations—moving beyond the traditional organization chart—is the mark of a Help Author who will thrive in the next generation of user assistance.



Extending our reach into Customer Support and Marketing ensures our position as user advocates.

Technology. In addition to organizational shifts and opportunities presented by emerging technologies, new development platforms and methods of distributing software and information provide us with better methods for supporting users:

- Web-based applications free us from the constraints of traditional, proprietary applications development and help authoring tools and take us beyond the book metaphor
- New technologies for applications delivery allow us to develop dynamic, conversational business relationships with customers for rapid feedback about their needs and use patterns
- Help for web-based applications and component-based applications can be stored as HTML or in a

database and easily updated along with other components of an application

- New technologies merge the application with immediate customer support
- Web technologies provide an opportunity to merge context-sensitive help with the interface, allowing us to provide more support than we would with a tooltip more easily than we can with typical embedded help

Models. A new, supportive application model that includes support information built into the user interface is not so hard to imagine.

Many applications vendors are creating interview-based interfaces, including rich domain content for consumer markets. Popular personal tax software, such as Tax Cut and Turbo Tax, as well as personal photo-editing packages, such as PictureIt and PhotoDeluxe are designed with an interview-style interface that walks users through a process and offers help along the way. These approaches offer examples on which to base development of more complex business applications.

How Do We Revive Help?

The help authoring industry is at the apex of fundamental change. By recognizing these shifts and taking advantage of new opportunities we can not only improve our standing within our organizations, but more importantly, help the organizations with which we work to deliver better products to users. There are many things we can do to revive help and to provide users the support they've been clamoring for.

Really Embedded Help. A hot term among Help Authors and developers in recent years has been "embedded help". Often "embedded" means little more than adding a tab containing the same book-style help we've become used to seeing to the application interface. Like context-sensitivity, embedded help, even in its most rudimentary form, is a step in the right direction—it gets information closer to the user.

For online help to be truly effective, however, we must push the boundaries of embedded help. A significant shift both for Help Authors and for development teams is to think of help as a core component of application development rather than as an accessory to it.

Aspects of product design typically the exclusive domain of programmers are these pieces of the user interface:

- Error messages
- Menus
- Labels
- Status bar messages
- Tooltips

Each of these interface elements provides users with information about using the application. A truly helpful application design must include an approach that integrates these elements with traditionally more detailed comprehensive information that appears in help files.

More than this, an improved approach to both help authoring and application development is to push support information out to the user interface, by including just enough information as the user is likely to need on the screen where it is immediately and always available—or by taking the more radical step of flipping development on its head and allowing help to drive the application design rather than the other way around.

Interview-based interfaces and wizards are examples of this type of approach. Learning to design tools that operate with this approach is the first step in building new skills for the next generation Help Author.

Beyond “Duh” Help. Many times when we document procedures or tasks we do little more than document the interface. What people often need instead of information about the tool is information about the domain in which they’re operating. When a new small business owner who knows little or nothing about principles purchases an accounting software package, she’s likely to need at least as much information about accounting (the domain) as she is about software (the tool).

An additional step in our professional development is to either become intelligent about the domains in which our products operate or to become well versed at gathering domain specific information, information that goes beyond a collection of menus and field labels to rationale and principles.

Not Covering It All. Less is more: one of the ways in which we can dramatically improve help and applications development is to limit the focus of the information we provide. By covering every menu option and giving each option more-or-less equal weight in our online help files, we overwhelm and confuse users.

We are better off narrowing the scope of the support we provide to these types of information:

- Information people are likely to need most frequently
- Information about the trickiest or most complex aspects of the tool
- Information people are likely to need in their first contact with the application

First impressions are as significant for software products as they are for interpersonal relationships. Early success builds confident users. These users are more likely to

explore other application features as they gain experience.

Don’t bother to document the obvious. If most users find it easy to use, leave it alone. Focus your attention instead on aspects of the application that deal with complex ideas. As you sort out what’s complex from what’s simple keep the following thought in mind: sometimes the process but the concepts are not. Help is often considered a solution for these problems, but the real issue is poor application design. In these cases redesign the user interface.

In addition, when you encounter information that’s required to keep a user from some disastrous outcome, make sure the information he or she needs is immediately and obviously available in the user interface, not hidden in a help topic.

Remember too, that the information people need most frequently isn’t necessarily information regarding the tasks they complete most often. In fact, they’re more often likely to require information about those tasks they complete the least.

Information-rich Interfaces. A key issue with the way help is perceived by software development organizations is that project managers, developers, and others often believe help can explain needlessly complex application designs. Help is not technological duct tape.

When an application or portion of application is confusing, it’s better to revisit the design of the application than it is to tack on help that users are unlikely to read. This conundrum presents a good argument for allowing help drive the application and for allowing user experience and user support to drive design and development rather than the other way around.

Adding help to solve design problems late in the game is pointless. Application redesign is costly and pushes back rollout dates. By focusing application design on users and their goals rather than on technology and product features software development organizations can minimize redesign setbacks. One of the products of user-centered design is an assessment of the users’ information needs.

A good way to address these needs involves a blending of support information with the user interface. Web-based applications, for example, are often spaces that blend tools with information. Help authors, who are already familiar with online media, are well suited to design these applications.

Active Help. Help conceived as a book is, like a book, passive. But when we shift our view of help to a perspective that considers help an activity (supporting users’ work) we begin to see new opportunities for

improving software. Users often don't use help because they must go out of their way to use it. They must seek, read, and understand help before they can apply it.

We can improve help by providing users with active assistance, help that senses what part of an application a user is working with, recognizes when there's trouble and offers unintrusive tips for success.

Learning-supportive Environments. Nobody likes to have to ask for help, and our applications shouldn't make anybody ask. A new view of help includes learning structures built into applications that users don't recognize as lessons.

Keeping Pace with Development. The technological shift to e-commerce and web-based application delivery means that programmers are learning new skills and developing new ways to deliver tools to users. The support we deliver for these tools must keep pace with these technological shifts.

We simply can't expect the same tools we've used in the past for developing help to be appropriate for completely new environments. This fact means that Help Authors must learn about new technologies (HTML, XML, JavaScript, Java) and the tools used to create them if we hope to provide adequate support for tools developed using these technologies.

This isn't just a technology issue or a help design issue. It's also an organizational issue. It means becoming a participating member of a development team, not a secretary, not a manager, but a designer.

SUMMARY

When we begin to view help as an activity rather than a thing, we're better prepared to offer software users the support they're looking for. However, this change in perspective means a change in our job. Many of just want to write and find facing these new challenges more than a little frightening.

On the other hand, the rewards of this career shift include the satisfaction of building better products and developing higher status within the organizations we serve.

Continued professional growth as a Help Author includes a title change to User Experience Designer, User Interface Specialist, Application Developer, or Builder of Better Products.

REFERENCES

- (1) Beyer, H., Holtzblatt, K. 1997. *Contextual Design: A Customer-Centered Approach to Systems Designs*. Morgan Kaufmann Publishers.
- (2) Cooper, A. 1999. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity*. Sams.
- (3) Hackos, J.T., Redish, J.C. 1998. *User and Task Analysis for Interface Design*. John Wiley & Sons.
- (4) Nielsen, J. 1999. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing.
- (5) Norman, D.A. 1999. *Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex and Information Appliances Are the Solution*. MIT Press.

Paul Sisler
Senior Interaction Designer
ArborComm, Inc.
220 N. Fifth Ave.
Ann Arbor, MI 48104
(734) 996-9006, ext. 14
psisler@arbor-comm.com

Paul Sisler is an active member of the southern Michigan web-based development environment and currently serves as chapter President for the Society for Technical Communication's Southeastern Michigan chapter. Sisler has worked on web development since the early days of Mosaic, and has focused on online help authoring for the past five years. He has worked as a RoboHelp trainer and has provided training services in HTML and HTML-based help development. He frequently presents on subjects including Cascading Style Sheets, integrating help in web-based applications, and hypertext theory and design.

Catherine Titta
President
ArborComm, Inc.
220 N. Fifth Ave.
Ann Arbor, MI 48104
(734) 996-9006, ext. 11
ctitta@arbor-comm.com

Catherine Titta has been solving business problems by identifying and creating information to support technology for fourteen years. She has authored online help in UNIX and DOS, before the days of RoboHelp 1.0, and has integrated online help, web content, and managed user interface design projects for a variety of information technology and Internet startup companies, such as JustTalk, Fullscope.com, www.i-gift.com, Crystallize Inc., Bell & Howell, and Veridian ERIM International. Named Washtenaw County's Small Business Person of the Year for 2000, Titta co-founded the Digital Design Institute of Michigan, directs a local new media networking group, and is on the board of the Society for Technical Communication, Southeastern Michigan chapter.