

Thinking Outside the Book: Wikis for Writing and Delivering Documentation

By: Scott Nesbitt



Documentation. It's a necessary evil in a software or hardware project. In some ways, peoples' attitude towards documentation can be distinctly bipolar. There are people who read and use documentation. There are people who don't believe it's necessary. There are those who never (or claim to never) read it. And there are those, like me, who write it for a living. What's funny is that all four camps complain when there's no documentation, or the documentation is sub par.

Broken glass. That's how I look at the ways in which documentation gets delivered. Some companies still put out printed, dead trees manuals. Most companies, though, issue PDFs or HTML-based documents. There's also online help (you know, what you get when you press F1). Word documents. Text files. UNIX man pages. Forums. Knowledge base articles. Files on hard drives. In emails and in places like Sharepoint repositories and Lotus Notes databases.

For the most part, they're good formats. But they're mired in a sometimes unnecessary level of complexity. On top of that, updates to the documentation often can't be made and deployed quickly.

Each shard of broken glass represents documentation in a different format. It's a form of information chaos.

Enter the wiki

In many cases, a wiki can bring order (or something resembling it) to that chaos. A wiki gives technical writer a way to centralize documentation in a single location. A wiki gives subject matter experts access to it. A wiki offers a very flexible environment for creating and delivering documentation, and for enabling users to contribute content.

For the next 50 minutes or so, I'll be discussing how people in the wacky world of technical writing are using wikis. I'll look at the tools, the techniques, the advantages, and the drawbacks of using a wiki for creating and delivering documentation.

What's interesting is that what I'm going to touch on in this presentation applies to *any* content that you put on to a wiki. You don't need to be a technical writer to take advantage of what I'm going to talk about.

Why a wiki for documentation?

When technical writers get together, one of the things that they invariably discuss (and argue about) is tools. Last year, I [talked to Adam Hyde](#) who founded the [FLOSS Manuals](#)

project. More on FLOSS Manuals in a little while. Adam mentioned that he noticed there is a lot of *tool fetishism* in the technical writing world. And he's correct.

I could probably spend 10 or 20 minutes talking about all the tools of the tech writing trade, and just as long about the various formats in which documentation is delivered. That kind of talk is only interesting to technical writers themselves. But you know what? A wiki can be an effective substitute for many of those tools.

So, what are the advantages of using a wiki for documentation?

- You can quickly set up a wiki
- It's easy to use and edit
- A wiki can promote and enable rapid authoring, updating, and publishing
- It can enable collaboration, both with people in a development shop and outside it
- A wiki has built-in version control and change tracking – this can be essential when you need to recover content
- It acts as a central repository for documentation (and

other information) inside and outside an enterprise. You don't need to implement a costly and complex content management system

- With a wiki, you can take advantage of any number of so-called Web 2.0 features like RSS, tagging, and searching using both the built-in search engine (OK, not so Web 2.0) or with a tag cloud

Wikis are very popular in shops that use the [Agile development](#) methodology. Because of the short iterations (one to two weeks) used by Agile teams, it's easier to post documentation on a wiki for review rather than passing paper or electronic copies (along with the attendant problems that causes) to subject matter experts. All reviewing and publishing (maybe even the writing, too) is done on the wiki. The documentation keep up with development, instead of having to scramble to catch up at the end of a project.

Here's an interesting statistic: according to wiki consultant [Stewart Mader](#), 39% of all wiki content in a company consists of documentation. Thirty nine percent. That's a lot. Why not leverage that even further?

Who's using wikis for documentation?

It might be easier to ask who *isn't* using them ... Here's a sampling of who is:

- A number of Open Source projects (small and large) like [Apache](#), [Ubuntu](#), the [Fedora Project](#), and [OpenOffice.org](#)
- Sun Microsystems
- Motorola
- Splunk
- Adobe Labs
- WebWorks
- Atlassian
- United States Army
- Various Internet Service Providers

That's just a few customer-facing wikis. What would be interesting to find out is the number of firms and projects that are using a wiki *behind* a firewall. I looked around, but couldn't find any figures about that. I'm sure, though, that number will be quite high.

How are they using wikis?

It runs the gamut of documentation. Everything from user guides to operational and troubleshooting documentation. From API references to user interface specifications. Some companies, like [Atlassian](#) (developers of the popular [Confluence](#) wiki), use wikis to deliver online help.

Atlassian publishes all of its documentation on a public wiki, powered by Confluence of course. But Atlassian goes one step further with their developer tools called [FishEye](#) and [Crucible](#). The documentation team has included icons in the user interfaces of those applications. Click an icon, and you're taken to the appropriate documentation page on the Atlassian wiki. And it's not just a single icon in the UI, either. The help is relatively context sensitive, with help icons beside elements on a screen.

Minimalistic documentation and wikis

I'm a believer in minimalism in documentation. By that, I mean the documentation should explain how to do something; all background and reference material should be elsewhere. If you're using a wiki, you can include that information elsewhere – out of the main flow of the documentation.

If you're not using a wiki as the primary delivery method for your documentation, you can still take advantage of one by shunting all of the overview and background information to the wiki. If readers want it, they can get access to it with a click or two.

Structure

Structure is essential when putting documentation on a wiki. Just dumping documentation – or any other content – on a wiki doesn't do the content or the reader any favours. It's

difficult to navigate and to find information. And who wants to scroll through page after page after page reading long blocks of text?

In the context of a wiki, I look at structured content as being broken into manageable chunks. Effective documentation is a structured collection of those chunks, and not just a big slab of information that's dropped in front of a reader.

For documentation, you can create that structure using the principles of *topic-based writing*. With topic-based writing, each portion of a document – reference material, a procedure – is an individual bundle of information. Those bundles are called *topics*. A topic stands alone; it doesn't rely on any of the content that comes before or after it.

This approach results in what seems to be disjointed documentation. There's little, if any, continuity between topics. Then again, who reads a manual from end to end?

The benefits of this approach

You might be able to see where I'm going with this. Each wiki page in a manual can be a topic. You can add continuity by linking between topics. Readers can find what they need, either through cross references or direct linking.

This opens the door to have several possibilities. Like what? Here are a few examples:

Online help. Each topic can be a a help screen for an application. You might recall that I discussed how Atlassian does this a few paragraphs ago.

Reusing content. That's another reason technical writers like topic-based writing. Often in a manual or set of manuals there's content that's duplicated – warnings, tables that contain descriptions of log files or directories, that sort of thing. Imagine you have something like that across four documents. Imagine having to change that four times with each new release of the documents? By using an *include* you only need to do that once and the change is spread to all the affected points. An include is like a macro. It points to content elsewhere on a wiki and pulls that content into a particular page.

Custom documentation. Not everyone needs the same information. Why should they have to spend time searching for what they need when they (or the technical writer) can build a custom document? That can be done in a wiki, and I'll be giving an example later.

Or, if the wiki supports the downloading of topics as individual files, readers can collect the topics that are relevant to them in a format like [ODT](#). Then, they can stitch those files together in a word processor or other application, and print the resulting file or generate a PDF.

The namespace is your friend

A [namespace](#) is like a directory on a PC or a server. It's used to group wiki pages that contain similar content. If you're dealing with multiple products or even just multiple documents for a single product, then the namespace is your

friend.

With a namespace, you can have individual silos on the wiki for each project and never their twains shall meet. Unless, of course, you do some linking.

The namespace enables you to silo documentation into different categories – like user guides, references, or troubleshooting. This makes the documentation easier to manage.

Getting documentation into a wiki

How you'll do this will depend on whether you're starting fresh or moving your existing documentation to a wiki. I'm assuming that you want to, as a [friend of mine](#) says, *wikify* your documentation. I won't go into the reasons why you wouldn't want to do that.

The most obvious way is to use the wiki itself. Every wiki has at least two editors – one in which you use wiki markup, and a WYSIWYG editor. WYSIWYG editors are a mixed batch – some are decent, and some are really buggy.

You could start off by writing documentation in a word processor, like [OpenOffice.org](#) Writer or Microsoft Word, and then convert it to wiki markup. You could do the same thing with HTML. There are tools to convert between word processor formats and wiki markup – like an add-on for OpenOffice.org Writer, or the [Word2Twiki](#) macro. There are even a few [HTML to wiki converters](#). Going this route,

though, involves a lot of copying and pasting. On top of that, there might not be a converter available for the wiki you're using.

There are also bits of software called *connectors*, which literally connect a word processor directly to a wiki. You write your documentation, click a button, and it's published to the wiki. A couple of example of these are the [Sun Wiki Publisher](#) for OpenOffice.org Writer, and the [connectors](#) between Confluence and Writer or Word.

To be honest, these aren't the greatest or even the most flexible ways of getting the job done. They work, but many technical writers (and I'm talking about the tool fetishists I mentioned earlier) would blanch at doing this.

There are ways in which to use familiar tools to write and maintain documentation, and then publish the content to a wiki.

WebWorks Publisher

I have something of a like/dislike relationship with [WebWorks Publisher](#). It's an application that takes files created in Microsoft Word or [Adobe FrameMaker](#) and converts them to online help. Recently, the application gained the ability to publish directly to a wiki. Right now, the wikis that are supported are [MoinMoin](#), [MediaWiki](#), and Confluence.

I haven't worked too extensively with WebWorks' wiki publishing function. I don't work with any of the supported wikis, and a license for the software is expensive. From

reports that I've been reading, and from a few people I've talked to, the results range from pretty good to ... well, not so much.

That said, I have to give WebWorks credit. The company eats its own dog food – they've wikified their documentation set, which is available at docs.webworks.com.

RoboHelp2Wiki

[RoboHelp](#) is a venerable application for creating online help. Again, it's one of those applications that I have a like/dislike relationship with. A number of other technical writers are of the same mind.

Still, RoboHelp is popular and thanks to [RoboHelp2Wiki](#), you can publish a RoboHelp project to MediaWiki. The process isn't as well automated as WebWorks' solution; there's still some manual fiddling that you have to do. But it doesn't do too bad a job from what I've seen and heard.

DITA

[DITA](#) is short of the Darwin Information Typing Architecture. DITA is an XML-based method for writing and delivering information in a variety of forms. Remember when I was talking about topic-based writing earlier? Well, DITA is designed around those principles. With DITA, you build topics and, from there, can build custom output by combining those topics using something called a *topic map* (which is like a table of contents).

Because of that, DITA is gaining a lot of traction in the technical writing world. And since it's topic based, DITA and a wiki are a good match.

One way to go from DITA to a wiki is with [DITA2Wiki](#). It's an Open Source tool from Lombardi Software that publishes a DITA project to Confluence. Not only do you get your content, but also working navigation. I saw a demonstration of DITA2Wiki at a conference in 2008 and not only did it work quite well, it was fast.

Is it that easy?

No. It's not enough to say *We've got documentation. Let's dump it all on a wiki*. You need to consider a number of factors:

- What are the relationships between the various bits of documentation that you have?
- How do they fit together?
- How can you link them?
- What kind of feedback mechanisms will you have in place?
- Will you have comment forms on each topic page? RSS feeds for new comments and updates to the documentation? Email feedback?

You might also need to restructure your documentation to make it more suitable for a wiki. Simply shoveling what you have on to a wiki might not work. Restructuring it could mean

breaking the documentation down into more manageable chunks, rewriting it to make it shorter and more friendly for the Web.

Getting documentation out of a wiki

I've met a few technical writers who view a wiki as a black hole. They say it's easy to get content into a wiki but tough to get it out. If not as tough as facing a black hole, then like pulling out a stubborn nail. Usually, I have to slap them down when they say that in my presence. It's not entirely true.

My view on this is don't worry about it. Why? *The wiki is the delivery method*. If you're putting your documentation on a wiki, you're doing it for a reason. And that reason is to easily write, manage, maintain, and deliver documentation. It's where customers will read and perhaps comment on the documentation. It's where they'll be sent when they press F1 or click the Help link in an application. It might even be where they find supplementary information like knowledge base articles and introductory material.

That said, there are still a number of people who can't separate themselves from the idea of a PDF or a pile of paper. For them, *that's* what a document is; not a page on a wiki somewhere. Yes, the paperless office is still a myth and probably will be for a while.

Going back to the black hole comment, most wikis have the ability to (either built-in or via add-ons) export to formats like:

-
- PDF
 - HTML
 - LaTeX
 - DITA
 - DocBook
 - Some other form of XML
 - LaTeX
 - ODT

They can generate files in those formats either from a single wiki page or from the entire document. It's not always the prettiest output, but it works.

FLOSS Manuals

One very interesting wiki-based documentation project is [FLOSS Manuals](#). FLOSS is short for Free/Libre Open Source Software. The goal of the FLOSS Manuals project is simple: create quality free documentation for free software.

The project uses a heavily-customized installation of [Twiki](#). In fact, it's so heavily customized that you don't know you're working in a wiki unless you take a really close look at page titles and the navigation between topics.

On the writing side, there's no wiki markup involved — FLOSS Manuals uses a WYSIWYG editor. Remember what I

said about topic-based writing? That's pretty much how the contributors approach the sections of a book. Each topic can stand alone. Which can make one of the ways of getting content out of the FLOSS Manuals wiki really useful for the reader.

FLOSS Manuals makes getting content out of the wiki very easy. You can download an entire manual as a PDF, as a set of HTML files, or you can get a widget that you can embed on a Web page or blog.

FLOSS Manuals also holds regular *book sprints*. These are events held over two to five days in which a group of contributors get together, both in one location and virtually. During the course of a sprint, the contributors literally write a book. Pretty much from scratch. At the end of it all, they generate a PDF and upload it to a print-on-demand service called [Lulu.com](#). You can download a PDF of the published book for free, or buy a dead-trees version.

But there's one more, very interesting way that you can pull content out of FLOSS Manuals. It's called [Remix](#). Earlier, I talked about giving readers the ability to pick, choose, and publish what they want. Remix lets you do just that. I've put together a [short screencast](#) that shows how it works.

While you might not get as many output formats as you might with one of the so-called standard documentation tools, the FLOSS Manuals wiki does a lot of what software costing hundreds, if not thousands of dollars, can.

Working with the crowd

This can be a tricky subject. There are a number of technical writers who are, to put it bluntly, control freaks. They don't like other people touching *their* documentation. When I run into technical writers who think like that, I have to remind them that the documentation isn't ours. It's for the customers. We're only maintaining it. We have no rights of possession.

When we consider the crowd, we think about a specific mass of people who use the software and hardware that we document every day. They push it, bend it, and try to break it. They use it in ways that we never imagine in our little development silos. And those people often have great tips and techniques that we can incorporate into our documentation. So why not let the users, whose insights into using an application or device can be as valid as ours, have a say too?

A wiki has the potential to exploit those insights and that experience by getting users to contribute new documentation and to add to what's there already. But this isn't easy to do. If you build and open the wiki, they won't necessarily come.

Going back to Atlassian, earlier this year the company opened its wiki-based documentation to customers. I talked with one of the writers there, and she told me that only about a dozen or so people created accounts and started editing. That number has probably increased a bit since then, but definitely not exponentially.

That more or less meshes with the figure I keep hearing bandied about: less than 10% (obviously far less) of users will contribute, and even fewer will continually contribute.

How to get people to contribute is *the* social media question. There's no definitive answer. If you want more information and ideas, you should definitely check out the following books:

- *The Economies of Online Collaboration*, by Peter Kollack
- [*Conversation and Community: The Social Web for Documentation*](#), by Anne Gentle
- [*The Art of Community*](#), by Jono Bacon

Assuming that you do get people to contribute, you need to 1) have a wiki style guide, and 2) ensure that contributors follow it. If they don't, your structure will break down and the wiki will grow uncontrollably.

A shift for technical writers

It's not just a matter of adopting a new tool. It's a shift in thinking and in doing as well. To effectively use a wiki, technical writers need to:

- Keep structure first and foremost in their minds
- Adopt topic-based writing
- Re-evaluate how documentation is delivered

-
- Seriously consider working with the crowd
 - Give up any ingrained notions of control or proprietary interest in the documentation

It's not going to be an easy shift.

Keys to success

There are several of them. First off, once again, is the need for structure. To create a structure and to maintain it. That can be tough, and will probably require one or more [wiki gardeners](#) to do the job.

Next, you'll need a way to *baseline* the documentation for each release. Documentation changes, and if your employer follows ISO or ISO-like processes, then during a quality audit you may need to correlate the information that's on a wiki with a particular release of a product. Or, you might be subject to some regulatory requirements and need to maintain the digital equivalent of a paper trail. There are several ways to do that, like:

- Putting the baseline version in its own namespace and locking it down
- Taking a snapshot of the baseline version
- Exporting the baseline version to a different format like PDF

You'll also have to make the wiki attractive to users. It shouldn't look like a wiki. It should have something akin to

the look and feel of your corporate Web site or the online documentation that the company has produced in the past. A good example of this, though not from the documentation world, is the [official wiki](#) for the TV series *Glee*. That wiki looks, more or less, like the rest of the Fox Web site and not like a wiki.

But the real key to creating a successful wiki for documentation is *content*. Giving people the information that they want in the way they need it. The documentation should focus on how to do things. If the content doesn't follow what I call the 5 Cs:

- Clear
- Concise
- Consistent
- Complete
- Correct

then it's worthless, no matter how you write or deliver it.

Is a wiki for everything?

Definitely not. I don't mean to end this on a downer, but wikis aren't suitable for every documentation task. Like what? Here are a few examples:

When readers are behind a very restrictive firewall, then having your documentation on a wiki has the potential to

shut them out. It all depends on how restrictive the firewall is, and how willing management is to ease those restrictions (at least as far as the documentation goes).

Documentation for mobile devices like smartphones poses an interesting problem. You have a small screen, and even the most minimalist skin for a wiki can be a bit cluttered. You might be able to make the wiki mobile friendly, but if readers need to scroll through more than a couple of pages, or need to jump around a lot, they might not find the extra effort worth it.

Technicians who work in the field might not have network or wireless access to the Internet, and will be cut off from any wiki-based documentation. On top of that, they might not have room to put down a laptop or a tablet.

Larger pieces of hardware, like copiers or medical equipment, also pose special problems. It's not acceptable for people to have to run back to their workstations in order to consult the documentation.

Your company's business objectives might not warrant the use of a wiki – whether it's open to external users for editing or not. In larger enterprises, it can be difficult to persuade the powers-that-sign-your-cheques to try something new. On top of that, some departments in larger firms like to keep their magic or their kung fu to themselves.

Summing up

Wikis are very flexible and powerful tools for delivering any kind of information. They're especially good for the creation and delivery of documentation of all kinds. Wikis can match many of the features found in the software commonly used by technical writers, software that can cost hundreds to thousands of dollars.

On top of that, the role of the technical writer is [rapidly changing](#). There's more and more documentation being created by the people using the products that technical writers document. By collaborating with users, and even having them write a portion of the documentation, technical writers can produce documentation that is truly focused on the needs of the user. And a wiki is the perfect vehicle with which to do that.

Contact Us



Web site:

<http://www.dmncommunications.com>

Email:

info@dmncommunications.com

Blog:

<http://www.dmncommunications.com/weblog>

Podcast:

<http://dmn.podbean.com>

Twitter feed:

<http://twitter.com/dmnguys>