

Portraying your passions

Job satisfaction needn't end with your retirement



Communicator

The Institute of Scientific and Technical Communicators
Summer 2010



**Facing ethical dilemmas:
tell us what you think**

**Hearing from users about
MadCap Flare**

**Reviewing SmartDocs from
ThirtySix Software**

**Designing and illustrating
content for localisation**

Laying the foundations for success

Nicky Bleiel discusses best practices that can help to save time and generate higher quality outputs when single sourcing.

Technical communicators have been single sourcing projects for many years—and we have tools that make this easy to do—but if the project isn't planned correctly, the outputs produced will not be as usable and logical as they could be. Proper planning also increases writing efficiency, so by starting with a good project foundation we can save time *and* produce higher quality outputs.

In this article, I will discuss best practices and methodologies for single-sourcing projects. Using best practices helps to reduce repetition, missing information, indecision and rework. You will create logical documentation for your users, no matter what the output. This article will focus on developing software documentation, but is applicable to any single-sourcing project.

Let's start at the beginning:

I'd like to start by providing context. In this discussion, the definition for single sourcing is.

Techniques used to create some combination of documentation for:

- *Multiple output formats (such as a printed manual and online help)*
- *Multiple audiences (such as versions for administrators and managers)*
- *Multiple deliverables (such as user guides and quick reference guides).*

A single project can deliver all of the above with proper planning and organisation (Figure 1).

Before any writing begins, determine the documentation deliverables. You will probably

be creating a help system and a PDF manual as a minimum (although if something isn't needed by users, don't produce it), but you should consider incorporating quick reference guides and tutorials into the project. You can use conditions (we will talk about those more later) to output information as separate deliverables. If these deliverables aren't originally planned for, they are often spun-off into separate projects; this adds extra work later.

Topics and paradigms

Structuring and organising topics are key to project success. Some of us write in a traditional book format (chapters, sections and so on), some of us write topics as separate chunks of content. Neither is more correct than the other; you can write either way and be successful. Ultimately, if your project is planned well, the end user will not know which paradigm you used; they will just find each output logical and the information findable.

Sorting out the content first, and creating and using topic architectures will keep you on track no matter which method you use.

Sorting content

To create topics that make sense in all outputs, we first need to determine what topics we need. After that, we can decide their structure and order. Start by deciding what information about your software application needs to be covered in each of the following categories.

Housekeeping

What mandatory topics must be included? Welcome topics, system requirements, installation, end user licence agreements, and support information all fall into this bucket.

Functionality

The core of your product is what it does to solve your users' problems. Note all of the major functionality of your product, using terms that users would use, not the internal names.

Reference

These topics provide information that is not topic-based in nature, such as glossaries and tutorials. Reference topics are not mandatory, but can make outputs much more useful. For example, if your product includes special terminology, creating a glossary and linking to definitions from the appropriate places in the

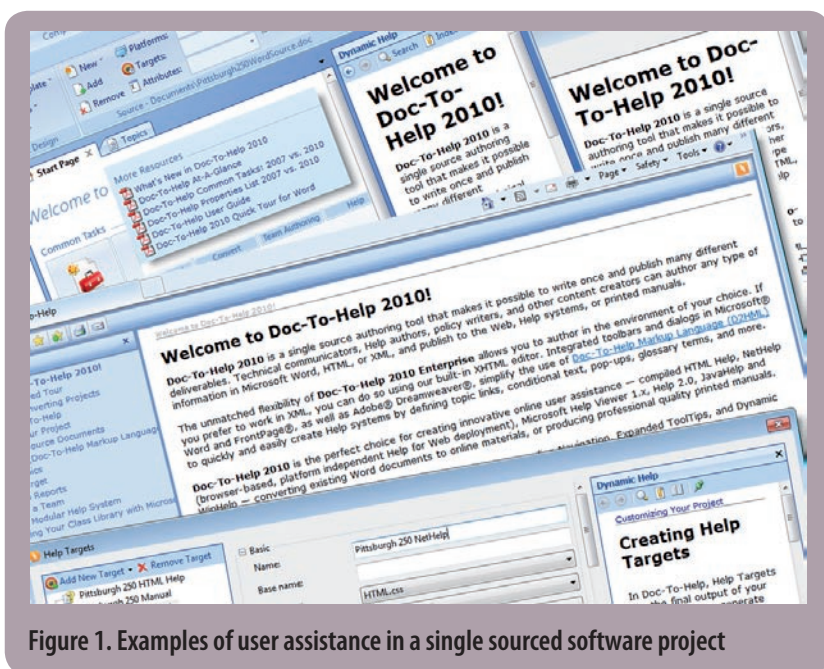


Figure 1. Examples of user assistance in a single sourced software project

online help can be very helpful to users, and it is easy to do.

User Interface (UI)

Software documentation has another sub-requirement: context-sensitive help within the user interface. For this, you need to analyse the user interface and note the dialog boxes and other interface items that will need specific help topics. Those topics will generally be dropped into the functionality bucket later, but they should start in this category so that they are not overlooked. Special requirements (such as embedded help) may require additional user interface topics not normally needed.

An existing project can be reorganised using this method. After you've sorted the content, you can more clearly see what is missing... and also what is redundant.

Creating topic architectures

Creating pre-defined topic architectures makes it easy to structure your topics so that you neither omit information nor repeat it in several places within your project. If there is an existing topic type theory you would like to use, take the time to analyse it and adapt it for the needs of your project.

To create topic architectures, first determine what you need. For example, my project included embedded help, so I needed user interface overview topics that explained windows, ribbons, panes and so on at a high level. I also needed an architecture for context-sensitive help for dialog boxes. Another architecture was needed for conceptual overviews of major functionality. Work out what architectures you will need and then decide what should go in them, keeping in mind the way the user will find them; for example, a dialog box topic should always explain how to open the dialog box, because the user may have found the topic through a search. You want users to know what they have to do *and* how to get there.

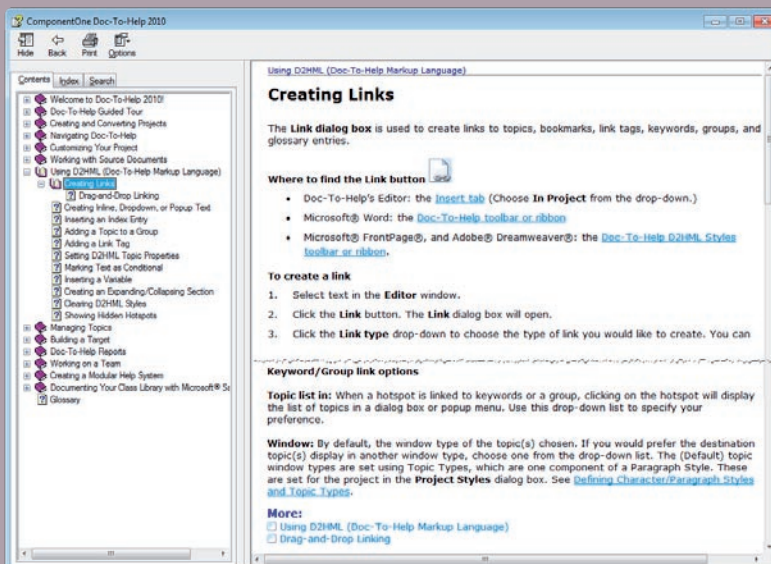
When you create your topic architectures, be sure to compare them and verify that requirements are not repeated among them, and that nothing is missing. You can create three or four to start with, and add more later if you find that you need them.

Conditions and variables

Tagging information for different uses (conditions) enables us to create a single project with many outputs. Creating chunks of reusable text (variables) makes our work much more efficient, so let's talk about them now.

Conditions are essentially 'markers' that allow you to flag specific information for different things. Using conditions, you can mark specific text or graphics to display only in specific

Example: topic architecture for information about a dialog box



Requirements:

- Breadcrumbs*
- Title
- 'Why do I need this dialog box?'
- Optional: Important Fact (for example, right-click information)
- 'How to get there' (include button graphic)
- Task(s)
- Dialog box field definitions
- Related Topics*

* Breadcrumbs and Related Topics are generated by the help authoring tool but additional Related Topics can be added manually.

instances: by output type (such as HTML Help, manual, JavaHelp), or by customised attributes (audience or deliverable), or by a combination of these. Attaching conditions to topics, entire documents or chunks of text gives you quite a bit of flexibility.

Single sourcing solely for multiple output formats will generally not require using conditions, unless you have specific information that should only appear in one output. If you write in an output-neutral way, that should not be necessary.

Using variable text, you can manage content in one place and reuse it across your project because the help authoring tool will insert the text at designated places in the final project.

If you are going to use conditions, determine what your conditions are and create them. Note what text is used repeatedly with your project, and create variables for that information. Variable text can be as short as your company or product name, or it can be longer pieces of text, such as descriptions of dialog box fields. It is common for software applications to use the same fields in several dialog boxes. Instead of keeping that description in several different topics, create a variable for it and insert it where needed. If it is repeated more than once in different contexts, it should be a variable.

Tips of all sorts*Naming conventions*

Be consistent — standardise on a consistent heading setup that makes sense in both online and printed outputs. Some prefer gerunds ('Adding, Renaming and Deleting Topics' or 'Inserting a Variable') but some don't. The important thing is to be consistent. Try to use terminology that the users would use, so that when they scan the table of contents or use the search facility they find what they need quickly. A bonus: if your online help is on the Web, your topic will surface in the search results.

Logical output

To help your documentation read logically in both printed and online outputs, avoid words that are specific to books or to help systems (such as chapter, page, topic, help system and manual). 'Section' is a good substitute. Don't fall into the trap of conditionalising this architecture of information.

A little minimalism

Only explain what your audience needs to have explained — don't give instructions on how to use common interface controls if the audience doesn't need them (for example, 'click the arrow keys up or down to set the line spacing'). That effort and those words are better used explaining what the ideal spacing is, or why you would change it. (An example from Word Help: Click 2.0, to double-space the selected paragraph. Click 1.0 to single-space with the spacing that is used in earlier versions of Word. Click 1.15 to single-space with the spacing that is used in Word 2007.)

Appearance of online and printed versions

Often, quite a bit of time is spent customising the way the online and printed outputs look. Formatting is a separate issue from single sourcing, but it is important to think about it before creating your topic types. You are going to want to consider what styles will be used in each topic architecture and this is a good time to make that decision.

This can be tackled by writing one book chapter or help topic, making sure that it includes all objects and styles, with several levels of headings, bullets, numbered tasks, tables and so on. Use greeked text as a placeholder if complete information isn't available (there are greeked text generators at www.duckisland.com/GreekMachine.asp and www.lipsum.com). Figure out what you need and don't need in your project, including the number of levels of headings and bulleted lists.

One other thing to think about: topics used for context-sensitive help on dialog boxes should use the same heading style (maybe two different styles at most). If the help for dialog boxes has a variety of different looks, it looks chaotic to the user.

Cross-references

You could make the choice to avoid cross-references altogether within your text and group them all at the bottom under a single heading such as 'More' or 'Additional Information'. It makes the copy cleaner, but since the link is not in the context of the topic, users may not see the significance as clearly.

Save time later

Create a variable for your product name. This will make it easy to do a quick swap if your product name changes, or if you need to create deliverables for multiple products. This creates a little extra work at the beginning, but can save hours later in the project cycle.

Make graphics more targeted

Some graphics appropriate for printed output are unnecessary or too detailed for online use. You can exclude graphics from online help using conditions; if you still want graphics in help, create unique, focused ones and make their condition 'online only'. If you are having your manual professionally printed, you will need higher resolution than for online help but this means larger output files. Produce one set of graphics at 300 dpi and conditionalise those only for print output. Use 96 dpi for online help.

Tables

Sometimes a table is the best way to make information findable and useful. When faced with explaining concepts that have a number of minute details, try sorting them into a table. For example, I created a table that lists documentation outputs, when they should be used, pros/cons, file output types and locations, and installation notes. This information could have been presented in a narrative, but putting it all together makes it easier for users to find the information they need. Point all related topics to it.

Putting it all together

Now that you know what topics you need to create and how they should be structured, you can create your project architecture. Put your topics (parents) and subtopics (children) in order. You can do this in an Excel spreadsheet, on a legal pad, in a wiki — whatever you prefer.

Of course, different outputs could have different project architectures, but you first want to get all of the topics from your four buckets into a logical order.

After you have done that, assign the appropriate topic architecture to each topic.

Following this methodology really is easy, and it does not have to be linear. You can work on different pieces of the puzzle at the same time. When working on the project architecture, you should put topics in priority order (with the least-used functionality lower down in the table of contents). If you need to cut topics because of time constraints, those will be the ones taken care of in the next release. This is much like

using the reverse pyramid in journalism.

Overall, one of the big takeaways is that you will find that you spend less time writing because you know exactly what information belongs in each topic, so nothing will be missed or repeated. **C**

Nicky Bleiel is the Lead Information Developer for Doc-To-Help. She has 16 years' experience in technical communication; writing and designing information for software products in the documentation, media, industrial automation, simulation and pharmaceutical industries. She is a Director at Large of the Society for Technical Communication.

E: nickyb@componentone.com

W: www.doctohelp.com

B: Technical Communication Camp

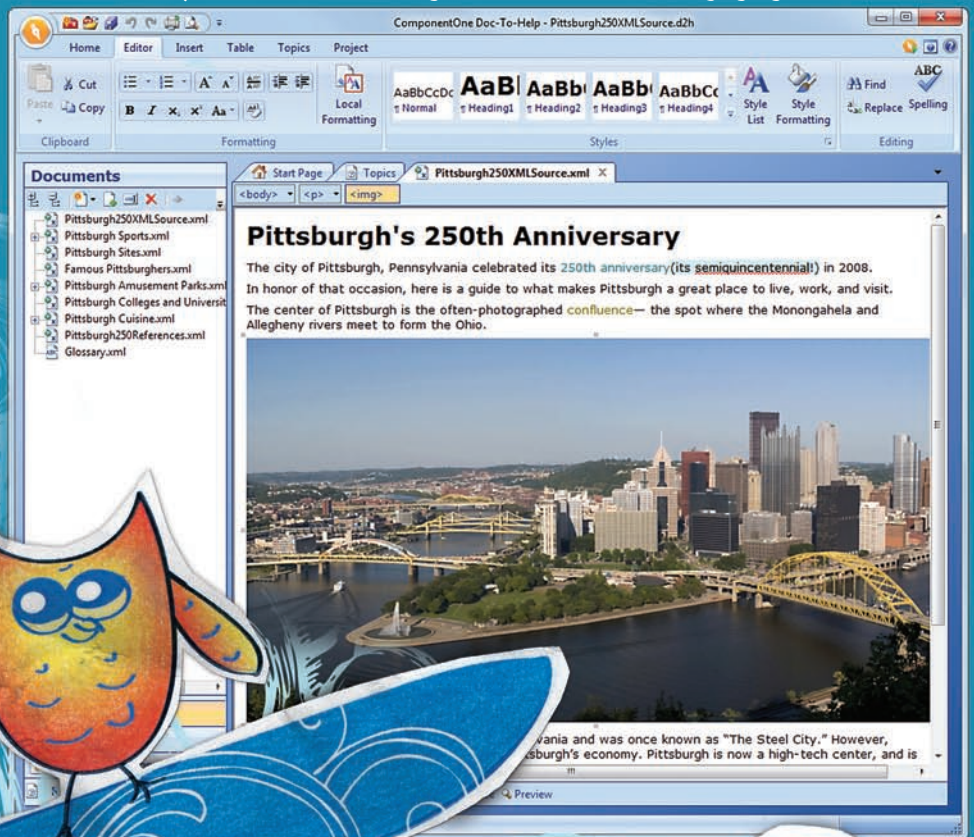
<http://blogs.componentone.com/CS/blogs/techcamp/default.aspx>

ALL - IN - ONE

AUTHORING & PUBLISHING

Doc-To-Help is the one solution for all your documentation projects. Publish as many outputs as you need with just one project. Use Doc-To-Help's editor, author in Microsoft Word, or create content in an HTML editor to produce online Help for desktop use, Web sites (NetHelp), and print-ready manuals. Automatic single-sourcing features ensure you can publish all this from one project without re-formatting.

Doc-To-Help does it all so you can relax & enjoy your summer!



AVAILABLE THROUGH ONE OF OUR VALUED PARTNERS

Grey Matter

www.greymatter.com/referral/componentone
+44 (0)1364 654 100

QBS Software Ltd

www.qbssoftware.com/componentone
+44 (0)8456 580 580

ComponentSource

www.componentsource.com/componentone
+44 (0)800 581 111