

## Choosing an XML Schema

If after analyzing your situation and considering the pros and cons you have decided to move to XML, you will need to decide which XML schema to use. For many organizations, there is not much choice. If you are working with a print publisher that uses the DocBook schema, you will end up using DocBook. If you share a Content Management System (CMS) with another organization, you will probably end up using the same schema they use. And, if you are working in certain industries, for example, aerospace or defense, you may end up using a specialized schema like S1000D.

If you do not have these constraints, then your choice will boil down to *DocBook* or *DITA*. These two are by far the most popular and best supported schemas for technical communication. Which one you choose will depend on the following considerations:

- ▶ What is your content like? Do you write narrative documentation (books, articles, etc.), modular documentation (reference pages, help pages, cook-books, etc.), or both?
- ▶ What are your deliverables? Do you deliver printed documentation, web pages, help systems, or all of the above? And, how important is each type relative to the others?
- ▶ How specialized is your content? For example, do you need to distinguish several different types of product numbers that are unique to your industry, or are you willing to mark up every product number the same way?
- ▶ How much content do you need to manage and how many writers do you have working on that content?
- ▶ How much can you afford to spend on tools and support?

Let's take a look at each of these in turn.

### **Content**

The type of content you author is the standard differentiator between DocBook and DITA. Common wisdom is that if you develop narrative material, like books or articles, you should use DocBook, and that if you develop modular documentation like help systems or if you want to build many different deliverables from the same content, you should use DITA.

While this is an important differentiator, you should not use it exclusively. It is not that difficult to write a book using DITA, and it is not that difficult to write

## 2 Choosing an XML Schema

modular content using DocBook. Both schemas are flexible enough to handle a wide variety of content.

However, there are important differences in the way the two schemas represent content, which means that while you can represent any kind of content with either schema, you will probably find that one or the other will be a more natural fit for your content.

There are two considerations to think about in finding the best fit:

- ▶ **Structural markup:** This is the way your writers think about and markup up the overall structure of their content. At one time, most documentation was structured like a book, with front matter, followed by chapters, followed by back matter (appendices, glossary, index, etc). DocBook was first developed to support this type of structure, though it has evolved over the years to support a much wider variety of structures.

The introduction of help systems integrated with the user interface of a product has led to topic-based methodologies. In a topic-based methodology, writers create individual modules of various types and knit them together into deliverables. DITA was designed specifically to facilitate topic-based authoring, and provides explicit support for this mode of development.

While this is an important distinction, it is also important to note that DocBook will support a topic-based methodology and DITA will support a book-based methodology.

- ▶ **Inline markup:** This is the detailed markup of content inside your structure. For example, if you are marking up a play, you need to identify dialog and stage directions. If you are marking up a software manual, you would want to identify commands, functions, variables, and so forth.

For software and hardware documentation, DocBook and DITA are similar. In keeping with its philosophy, DITA has fewer inline elements and expects users to create what they need as specializations. DocBook has more choices, and is less likely to need specialization. That said, markup for most common software and hardware components is available in both schemas. You may find one or the other more or less to your liking, but the choice will be primarily aesthetic rather than functional.

If you use a topic-based, modular methodology, you will probably find DITA more to your liking. It has native structures that support and “enforce” this style of writing. DITA was developed specifically to support a topic-oriented architecture and its structures steer writers towards that architecture. It is also possible to write a book or an article using DITA, but unless you customize the schema,

you will find yourself using elements with names like `<topic>` instead of `<chapter>`.

DocBook does not enforce a particular style, but because of its heritage, it leans towards a more traditional book or article approach to writing. However, it has supported modular methodologies for many years, albeit without the latest terminology. So, it is also possible to write topic-based content using DocBook, but unless you customize the schema, the topics will be marked up using elements like `<article>` or `<section>` rather than `<topic>`.

If you are willing to customize your schema, then you may want to let other considerations have greater weight. If not, then if your team writes traditional documentation, like books and articles, and is not planning to move to topic-based content, you will probably be happiest with DocBook. If on the other hand your team writes topic-based content using a methodology like *Information Mapping*, you will probably be happiest with DITA.

However, many teams need to write both kinds of documentation. If that is your situation, I would lean towards DocBook. I think DocBook does a better job handling modularity than DITA does handling books.

## ***Deliverables***

Both DocBook and DITA have standard stylesheets that let you create a range of deliverables. As I am writing this, both the standard DocBook stylesheets and the DITA Open Toolkit will produce: print (using XSL-FO), HTML, XHTML, HTML Help (HTML that can be compiled into Microsoft HTML help), JavaHelp, and Eclipse help. In addition, DocBook has stylesheets to generate WordML and plain text, and DITA has stylesheets to generate Microsoft's Rich Text Format (RTF) and *troff*. Finally, there are stylesheets available to convert DocBook to DITA and DITA to DocBook.

Since both DITA and DocBook support essentially the same set of output formats, the important differentiator here is how well the stylesheets work for your deliverables. The DocBook stylesheets are more mature, and are well documented in Bob Stayton's *DocBook XSL: The Complete Guide*. The DITA stylesheets are newer, and have less well developed documentation, but there is a growing community around DITA

No matter which you choose, the odds are overwhelming that you will need to customize the stylesheets. I have never seen an organization that did not need something different from the standard look and feel. If you have XSL-knowledgeable staff or contractors to customize your stylesheets, then either DocBook or DITA will serve you well. If you are running on a shoestring, or if you have less

experienced staff, you will probably find that DocBook is the better choice. It is better documented and the standard stylesheets provide a much wider range of parameters and options that do not require any programming.

### **Customization**

Both DocBook and DITA can be used “out of the box” to author useful content, and both can be customized relatively easily.

DITA is designed with customization, or in its terminology “specialization,” in mind. It uses an object-oriented model that allows you to create new elements that are specializations of existing elements. For example, if you were writing an application for an airline, you might want to markup flight numbers. Using DITA, the way to do this would be to create a new element called `<flightnum>` that is a specialization of an existing DITA element (for example, the `<programnum>` element).

DocBook starts out with a larger set of elements, so for many applications you can use standard DocBook where you would need to specialize DITA. However, you can customize DocBook in several ways. DocBook supports a user-defined attribute called `role` that can be used to differentiate variant types of an element. The `role` attribute can be used without changing the schema at all, which means that many “customizations,” including the flight number example above, can be handled within the standard. If you need need to do more, the latest version of DocBook – as of this writing that is DocBook 5.0 – provides support for extensive customizations, including specialization.

The “normative” or standard version of DITA is distributed as a collection of DTDs. The normative version of DocBook is distributed as a RelaxNG schema. While DTDs can be very flexible, RelaxNG provides much more precise control over the schema. This means that even though DITA was designed for customization, and shines in that category, DocBook is also easy to customize.

If you plan to do extensive customization and are working in a modular environment, you will probably be happier with DITA. If you are looking for a schema that will cover a wider variety of content “out of the box,” you will probably be happier with DocBook.

### **Scale**

XML may not be your best choice for small, isolated projects. The overhead may simply be greater than any benefit. But, if your project is big enough to consider using XML, but still small, it is likely that DocBook will be more cost-effective.

The are several reasons for this:

- ▶ DocBook is less likely to require customization.
- ▶ The DocBook stylesheets are, at this writing, more comprehensive, better documented, easier to modify, and more likely to be useful with minimal modification.
- ▶ DITA and DocBook can both be used without Content Management software, but as you scale up in size, DITA will need a CMS sooner than DocBook. For a project of any given size, DITA will almost always require more files and more links between files than DocBook. Therefore, the point at which managing a project's files manually gets out of hand will come sooner for DITA than for DocBook.

### **Cost**

Except at the low end, it is unlikely that choosing DocBook or DITA will make a significant difference in the cost of moving to XML. The cost of analysing your content, designing and implementing a solution, training your team, and maintaining your solution will overwhelm any likely differential between DocBook and DITA.

There is one potential exception to this. If you need to reuse content extensively, have a large documentation set, and have many writers, DITA *may* enable you to design in efficiencies that would be harder to achieve with DocBook. The only way you will know is by doing detailed analysis of your existing content and your proposed solution. It is not clear to me that DITA provides enough extra support for modular methodologies to make a significant difference over DocBook in this dimension, but I am willing to be proven wrong.

### **Buzz**

There is one other factor you should at least think about; popularity. As I write this, more people use DocBook than DITA, and it is clearly more stable and mature. However, the “buzz” clearly favors DITA. If you go to a technical communication conference, you will find many sessions on DITA and few (or none) on DocBook. There is more and more being written about DITA, and the best known consultants are devoting most of their time to it. Whether this foreshadows a mass movement to DITA and the decline of DocBook is not 100% clear, but there is at least the possibility that DocBook has seen the peak in its popularity. If that is the case, then over time it will become less well supported.

I would not use this as a deciding factor; in fact, I hesitated to mention it at all. However, if DITA becomes predominant, it will begin to get better tool support and it will become easier to find writers and tools people who are familiar with it. The good news is that if you are using DocBook, there is still a vibrant com-

munity that uses it, supports it, and will continue to do so for a long time. And, if you ever need to move from one schema to the other (or to some future schema), it will be much easier to do so with either DocBook or DITA than it would be with proprietary markup.

### ***Putting it together***

To make a decision, you will need to weigh the relative importance of each factor for your situation. Usually, the most important factors will be content and deliverables, in that order. Your writers will spend more time with the schema than anyone else; if you pick one that is not a good match in these dimensions, they will be less productive.

The importance of the other factors will vary. If your content would require neither schema to be specialized, that factor will be less important. Or, if you are blessed with an adequate budget, you may not have to stress as much over fine distinctions in cost.

One thing to guard against is letting a secondary factor dominate your decision for the wrong reasons. For example, if you happen to have an engineer who is a DITA expert, that is a plus for DITA, but if everything else points to DocBook, do not let that one factor overwhelm your decision.

A skillful vendor presentation can also have a disproportionate effect. You are best off avoiding sales presentations until you have analyzed your situation and made some basic choices. Then use the presentation as a way to find out how well the vendor's product matches your requirements, rather than letting the vendor steer you towards a solution that matches their product.

In the end, you need to analyze your current situation and your proposed future situation to make sure you understand the core reasons for making a change. Remember that the schema is only one of many decisions you will need to make as you move to an XML-based environment. Look at all of these decisions together, and if you need to, work with an outside consultant to help you better analyze your choices. Be aware, however, that most consultants have their pet solutions, and many have ties to particular vendors. Make sure you pick one who can give you an independent evaluation.