

Assessing DITA as a foundation for XML implementation

WHITE PAPER



Sarah O'Keefe
President

The Darwin Information Typing Architecture (DITA) is being positioned as *the* solution for XML-based technical content. Is DITA right for you?

This white paper describes the potential business advantages of DITA, provides a high-level overview of DITA's most important features, and then discusses how you can decide whether to develop a DITA-based XML implementation.

What is DITA?

The Darwin Information Typing Architecture (DITA) is described by its creators as:

“...an architecture for creating topic-oriented, information-typed content that can be reused and single-sourced in a variety of ways. It is also an architecture for creating new information types and describing new information domains based on existing types and domains. This allows groups to create very specific, targeted document type definitions using a process called specialization, while still sharing common output transforms and design rules developed for more general types and domains.” (<http://xml.coverpages.org/dita.html#overview>)

DITA includes several components:

- ❖ Document type definitions (DTDs), which define the structure of DITA files.
- ❖ Open Toolkit, which provides a set of files to create output from DITA XML content. The Open Toolkit is made up mostly of Extensible Stylesheet Language (XSL) transformation files (to produce HTML output, for example) and XSL-FO (formatting objects) transformation files for generation of Portable Document Format (PDF) files.
- ❖ Documentation, which describes the element set and their relationships.

DITA is intended as a starting point or a foundation for XML authoring, and it includes several interesting features to support customization. (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita)

Originally developed by IBM, DITA is now an open source standard managed by the Organization for the Advancement of Structured Information Standards (OASIS) and is available for download from the [OASIS web site](#).



The basic information unit in DITA is the topic. Topics are assembled into deliverables, such as books or help systems, using map files. Each map file describes a sequence of topics and their hierarchical relationship. Transforming the map file results in the final deliverable format.

Business case for DITA

DITA is an XML vocabulary, so the rationale for XML implementation also applies to DITA. A publishing workflow based on XML can:

- ❖ Enable content exchange
- ❖ Support automated database content extraction
- ❖ Lead to reduced content duplication and increased reuse of information
- ❖ Extract information based on structure and metadata
- ❖ Improve formatting consistency
- ❖ Reduce author learning curve
- ❖ Reducing repetitive desktop publishing tasks in house and in translation
- ❖ Improve compliance with required document structures

For a detailed discussion of the rationale for XML implementation, please refer to our [Structured authoring and XML](#) white paper.

DITA may offer additional benefits, which are described in the following sections.

Reducing content modeling requirements

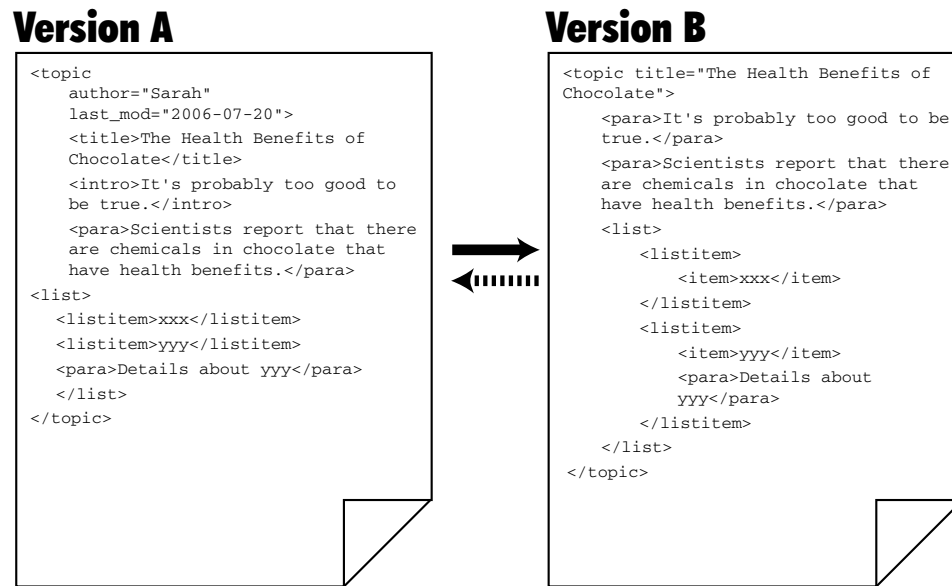
The transition to XML-based authoring typically includes a significant content analysis effort. The DITA architecture offers a modular, flexible architecture designed to support software documentation efforts. By assuming that DITA structure is a reasonable match, some organizations are bypassing the content modeling effort.

Eliminating content modeling allows for a much faster transition to structured authoring. The trade-off is that DITA, out of the box, probably is not an exact match for the organization's content. Without doing content analysis, though, it's hard to know ahead of time whether the mismatch will be significant. Using more general markup is a potentially serious problem for content in specialized industries. For example, the metadata required to support medical device documentation or financial services information probably goes beyond what DITA provides by default. If, however, your organization is creating basic documentation for consumer software, DITA structure might be a good fit.

Making content truly portable

The XML file format is application-neutral and easily read by many different applications. But when two organizations choose different XML structures, exchanging information can still be problematic. Consider the example shown in Figure 1 on page 3.

Figure 1: XML solves the technical interchange problem, but differing content organization can still result in incompatible content.



XSL transformation would allow you to convert the element structure shown in version A to the element structure shown in version B. For this example, the transformation would be simple as the two element trees are basically compatible. Notice, however, that Version A contains author and modification date information and Version B does not. That means that you when you transform from A to B, you will discard some content. If you author new content in B, transformation to A will result in a document that is missing some information.

One argument for implementing DITA XML is that the content will be compatible with any other organization using DITA. This can be especially important for companies that license their products to other organizations. IBM, for example, has standardized on the use of DITA for technical publications, so if you are required to deliver content to IBM, it's likely that they will request (or require) DITA-based content.

Supporting content reuse

For many organizations, reuse is a primary factor driving XML implementation. DITA supports reuse of topics and smaller bits of information. This is discussed in more detail in the “Architecture overview” on page 4. A custom XML implementation can also support content reuse, but the work has already been done in the DITA architecture.

Tools support

The Big Three XML authoring tools—Arbortext Editor, structured FrameMaker, and XMetaL—provide support for DITA authoring. In your evaluation, be sure to look at the following:

- ❖ Support for map files
- ❖ Integration with the Open Toolkit



- ❖ Creation and resolution of content references (conrefs)
- ❖ Support for specialized elements

Software vendors are some of the most ardent supporters of DITA. For a software developer, supporting all the possibilities of XML presents a formidable challenge. Focusing on DITA, a specific implementation of XML, makes it much easier to create user-friendly authoring applications.

In other words, if you choose to implement DITA, you will probably reduce the cost of configuring the authoring and publishing tools you choose. Keep in mind, however, that the up-front implementation costs are only a small component of the overall cost structures that you need to evaluate. Reducing integration cost is obviously a plus, but it probably won't make or break your business case analysis.

Generating output

The DITA Open Toolkit (DITA-OT) provides transformation files to create output for several common formats, including PDF, HTML, and HTML Help. Using the Open Toolkit as a foundation for your publishing efforts can reduce the cost of building output generators.

For a custom XML architecture, no default transformations are supplied, so you must build the output generators on your own.

The toolkit files provide basic output. If you want to customize this output, you need to modify the transformation files provided with the Open Toolkit. If your output requirements differ significantly from the default, or if you do extensive customization work on the default DITA structure, the effort required to customize the toolkit files may be just as great as the effort to build output generation files on your own.

Architecture overview

This section describes the features that make the DITA architecture especially interesting.

Topics as basic content units

The default content unit in DITA is the topic. (In DocBook, it's the book.) Topics are assembled to create deliverables, and you can reuse topics in as many deliverables as necessary.

By default, DITA ships with four topic types: task, reference, concept, and the basic topic. You can also add additional topic types. This emphasis on information typing results in a library of content that separates procedural, conceptual, and reference information.

The topic-oriented architecture requires that authors create modular, self-contained information. For content creators who are accustomed to working on cohesive books, this can be rather a difficult transition.

One topic (sorry!) of heated discussion is the issue of "glue text," the content that provides coherent transitions from one topic to another. Some argue that glue text is unnecessary and that transitions are overrated; at the other extreme is the opinion that modules without transitions are

unusable. If you belong to the latter group, keep in mind that implementing transitional text in DITA is quite difficult. Transition text that makes sense in one context might not be relevant in another.

Reuse with map files

DITA provides two major reuse mechanisms: maps and references. DITA maps provide a list of links, in a particular sequence and hierarchy, that describe the content of a deliverable, as shown in the following example:



```
<?xml version=" 1.0" ?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map title="Zoo Policies">
  <topicref href="Animal_nutrition.xml">
    <topicref href="Aardvark.xml"/>
    <topicref href="Baboon.xml"/>
    <topicref href="Crane.xml"/>
    <topicref href="Dingo.xml"/>
  </topicref>
  <topicref href="Visitor_behavior.xml">
    <topicref href="Adults.xml"/>
    <topicref href="Children.xml"/>
  </topicref>
</map>
```

Map files are similar to FrameMaker book files, but offer more flexibility—a map file can contain references to other map files, and topics can be nested more than one level.

Typically, you would create a map file to support each major deliverable. Thus, components of the *Zoo Policies* shown in the preceding example could be reused in an *Animal Care Guide*. The information about animal nutrition might be provided in both documents, but the discussion of visitor policies would appear only the policy manual. Multiple map files can reference the same topic as necessary.



Reuse with content references

For reuse inside a topic, DITA provides a content referencing mechanism. A bit of content that will be reused carries a unique ID. In this example, the <note> element is set up with an ID for reuse:

```
<topic>
  <title>Aardvark</title>
  <body>
    <p>Aardvarks eat mostly termites.</p>
    <note type="danger" id="nofeeding">Do not feed snacks, scraps, or people food to the animals.
    </note>
  </body>
</topic>
```

You then create a reference to the element you want to reuse by specifying the file name and the ID in a conref attribute:

```
<topic>
  <title>Baboon</title>
  <body>
    <p>Baboons eat mostly fruit.</p>
    <note type="danger"
      conref="aardvark.xml#nofeeding"/>
  </body>
</topic>
```

One difficulty with conrefs is that they are quite tedious to create and manage manually. If you plan to use conrefs, consider how well your potential authoring tools support content creation and management.

Conditional content

The DITA architecture provides support for attribute-based versioning. That is, if you have some content that is intended only for some deliverables, you label the content with the relevant attributes:

```
<topic audience = "internal">
  <title>Secret Settings in our Software</title>
</topic>
<topic>
  <title>Setting up Your Project</title>
</topic>
<topic audience="external">
  <title>Controlling Your Files</title>
</topic>
```

When you publish content, you use a settings file called ditaval to specify which information to exclude from the final output. In the preceding example, a deliverable for internal users does not require a ditaval file because internal users get all of the information (internal, external, and all audiences). For external users, you need to suppress the internal-only information, so you set up a ditaval file as follows:

```
<val>
  <prop att="audience" val="internal" action="exclude"/>
</val>
```

You can also set up more complex conditional processing with intersecting attributes. For instance, you could produce the Linux-specific version of a document for external users by excluding the internal and Windows-specific information, as shown here:

```
<val>
  <prop att="audience" val="internal" action="exclude"/>
  <prop att="platform" val="win" action="exclude"/>
</val>
```

You will need a ditaval file for each combination of conditions, or you can create some additional processing to create a custom ditaval file when you are ready to publish your content.

Customization

You can customize in three different ways:

- ❖ Subsetting
- ❖ Specialization
- ❖ Extension

Subsetting

Subsetting means that you remove extraneous elements. For instance, consider the standard `<body>` element definition in DITA:

```
(p or lq or note or dl or parml or ul or ol or sl or pre or codeblock or msgblock or screen or lines or fig or syntaxdiagram or imagemap or image or object or table or simpletable or required-cleanup or section or example) (any number)
```

After reviewing your content, you decide that you do not need the highlighted elements. You remove those elements and create a new, shorter definition for `body`:

```
(p or lq or note or dl or parml or ul or ol or sl or lines or fig or imagemap or image or object or table or simpletable or required-cleanup or section or example) (any number)
```



Extension

If you cannot build the structure you need with subsetting and specialization, you need to consider extending DITA. When you extend, you create a structure that no longer conforms to the standard specification (see Figure 3 on page 9). This means you have to extend the entire tool-chain to support your customizations.

The DITA community, in general, frowns on extension. They recommend using the specialization mechanism to maintain interoperability with other DITA content. However, starting with DITA and making customizations could be less time-consuming than building the structure from the ground up.

Generating output

XML is generally not appropriate as a delivery format, so content must be published to HTML, PDF, CHM, and the like. The effort required to configure publishing streams is significant and may involve multiple sets of tools and technologies. For example, creating plain HTML from XML requires someone who at a minimum understands Extensible Stylesheet Language Transformations (XSLT), HTML tagging, and Cascading Stylesheets (CSS). Output to PDF/print is even more complex, requiring knowledge of XSL-Formatting Objects (XSL-FO), structured FrameMaker configuration, knowledge of the E3 publishing engine from Arbortext, or a similarly complex toolset.

The DITA Open Toolkit provides a set of transformation files for several types of output. This gives DITA-based authors a starting point for creating their final output. Configuring and customizing the Open Toolkit files is not trivial, but it's probably easier than building the entire publishing workflow on your own.

Several commercial tools, including Arbortext Editor, XMetaL, and structured FrameMaker, provide helpful integration of DITA output based on the Open Toolkit.

Implementing DITA versus implementing custom XML architecture

With a basic understanding of DITA, it's time to tackle the \$64,000 question: Should you use DITA for your content? First, you need to determine whether XML in general makes sense for your content requirements. If you decide that XML is appropriate, take a look at DITA. The following table outlines a few possible scenarios.

Scenario	Recommendation
Content must conform to a specific standard, such as S1000D (manufacturing and aerospace), SPL (Structured Product Labeling, pharmaceuticals), or NewsML (newspaper articles).	Use the required standard.
DITA, out of the box, meets all requirements.	Use DITA.

Scenario	Recommendation
A customer or business partner requires you to deliver DITA content.	Use DITA.
Content contains lengthy narratives that cannot be broken into reasonable modular chunks.	DITA is probably not a good fit. Consider a different standard, perhaps DocBook, or build your own.
Single sourcing is a requirement. No existing content. Can be flexible with markup requirements in exchange for quicker implementation.	DITA is a good fit.
DITA is not an exact match; customization would be required.	Compare the cost of DITA customization to the cost of custom implementation.
Markup requirements are industry-specific, complex, and strict.	Look for an existing standard in your industry or build a custom structure.

For a detailed discussion of XML implementation, refer to our [Managing implementation of structured authoring](#) white paper.

Any XML implementation project should begin with an assessment of the business reasons for moving to XML and an analysis of current and future content requirements. Once you understand your requirements, determine how closely DITA matches your content requirements.

If DITA provides a reasonable match, you need to conduct additional research on specialization, the Open Toolkit, software support, content migration, modular writing, and the like. Before you decide to implement DITA, it's critical to understand all of the issues—DITA is much more than just a DTD.

Using DITA or another preconfigured solution out of the box is less expensive than creating a custom XML structure and the publishing workflow to support that XML. As your customization requirements increase, however, there comes a tipping point where the cost of massaging the standard structure becomes greater than the cost of a custom build.

You must also consider the quality issues: How close to your exact markup requirements can you get starting from a DITA base? How important is it to match your requirements? Can you compromise? Are there existing specializations that you can leverage instead of doing your own? What sort of documentation and training are required to ensure your content creators are successful in the new environment?

DITA is the result of thousands of hours of research and development. It provides a graceful stepping stone into XML-based authoring. It is not, however, the only way to implement XML.



Resources

Below are a few links to help you get started with additional DITA and XML research.

DITA OASIS Technical Committee

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita

DITA Open Toolkit

<http://sourceforge.net/projects/dita-ot/>

Norm Walsh (creator of DocBook) discusses DITA

<http://norman.walsh.name/2006/03/24/dita2006>

Scriptorium white papers

<http://www.scriptorium.com/papers.html>

Scriptorium Publishing Services, Inc.

PO Box 12761

Research Triangle Park, NC 27709-2761

USA

info@scriptorium.com

919-481-2701

www.scriptorium.com

