DITA, Content Management and XML Authoring

Implementing DITA in Content Management Systems in general and Documentum in particular

Paul **Prescod**

Abstract

DITA is quickly becoming the dominant XML schema for topic-oriented authoring. DITA's topic-oriented model raises new challenges for XML authoring and Content Management Systems. Many DITA adopters are tasked with the responsibility of implementing DITA by configuring a departmental CMS or an XML aware Enterprise Content Management platform such as Documentum. This paper describes how to implement key DITA concepts on an (Enterprise) Content Management platform with a particular focus on Documentum.

Table of Contents

1. Background

DITA is an up-and-coming standard for XML authoring. DITA is not just a set of pre-defined DTDs/Schemas for common topic and real-world document types. It is really a comprehensive framework for technical authoring. DITA allows organizations to define their own document types by customizing the base types using a sort of inheritance technique called specialization.

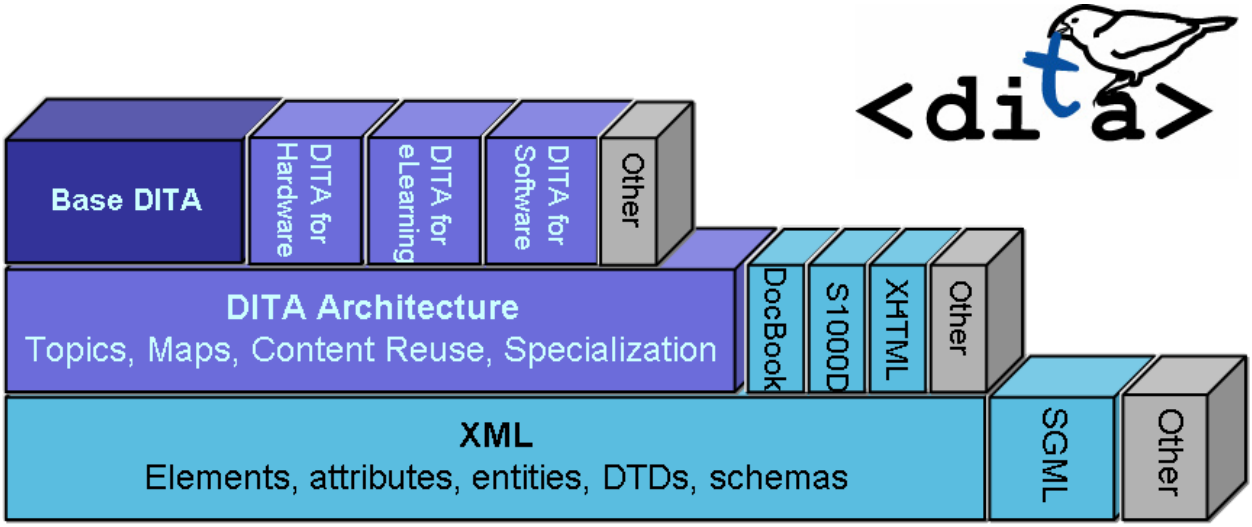

Figure 1. DITA as a Framework

In addition, DITA is a highly practical way of moving to XML authoring in general and granular content reuse in particular. DITA distinguishes itself from earlier standards by explicitly rejecting the book paradigm in favour of a topic-oriented model. A topic is a single continuous narrative that should be written to be independently usable and understandable. Topics tend to be medium-sized objects with independent titles and metadata: more analogous to web pages or chapters than paragraphs or lists. Topic orientation has advantages for the reader and also for content creators. Readers prefer topic-oriented information because it can be read in bits. Furthermore, readers can choose their own paths through the content. Content creators like topic-oriented authoring because it can drastically reduce the amount of information they need to write across a complex modern product line consisting of compound products (such as operating systems, product suites or automobiles) and subset products (e.g. "light versions"). Instead of rewriting or copying and posting content, authors can mix and match topics to meet the needs of a particular product or audience.

Because of the topic-oriented nature of DITA, many technical writing departments decide to move to DITA and Content Management at the same time. Topic oriented authoring can drastically increase the number of files that an author must keep track of. Furthermore, authors are expected to collaborate on content creation to a much greater extent. You cannot safely combine topics into deliverables if you have no management over who is changing them, when they are doing so or why. Content management systems make reuse manageable and scalable.

This paper is about using DITA with a departmental or enterprise content management system using the leading ECM, Documentum as an example. Because Documentum is often deployed company-wide, it is frequently the responsibility of each department to determine for itself how to make their content repository fit its needs. We hope that this implementation roadmap will help organizations understand the issues and options relating to the intersection of DITA and Content Management.

2. The DITA Model of Content

The two central units of authoring in DITA are the topic and the map. A topic is a single continuous narrative that should be written to be independently usable and understandable. A map combines multiple topics into a structure that is appropriate for a particular deliverable. A topic might be "Configuring the server." That topic might appear in an installation manual, a troubleshooting guide and a system maintenance handbook. Each book would typically be represented by a unique map (although maps can share structure). Maps link to topics through an element called `topicref`.

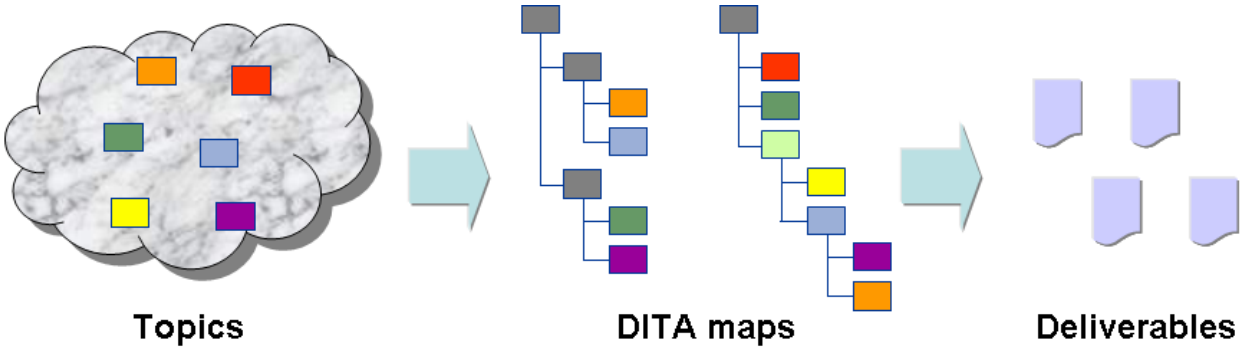

Figure 2. DITA Maps Combine Topics Into Deliverables

Maps are XML documents that consist primarily of links to topics and metadata. They do not have content themselves. Most often, they are hierarchical and define a Table of Contents for the deliverable. Portions of maps may also be table or list structured.

Example 1. Hierarchical Map Example

```
<?xml version="1.0"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map title="Welcome and Sample Documents">
<topicref href="Welcome_to_XMetaL_DITA_Edition.xml"/>
<topichead navtitle="Sample DITA Project">
    <topichead navtitle="Changing a Print Cartridge">
        <topicref href="cartridgeoverview.xml"/>
        <topicref href="prepareprinter.xml"/>
        <topicref href="removecartridge.xml"/>
        <topicref href="newcartridge.xml"/>
   </topichead>
    <topichead navtitle="Color Printers">
        <topicref href="colorprinters.xml"/>
        <topicref href="liquidinkjetprinters.xml"/>
        <topicref href="solidinkjetprinters.xml"/>
        <topicref href="dyesublimationprinters.xml"/>
        <topicref href="colorlaserprinters.xml"/>
    </topichead>
</topichead>
<topichead navtitle="Sample DITA Topics">
    <topicref href="ditaintro.xml" navtitle="Sample Generic Topic"/>
    <topicref href="Using_XFT_Forms_Layout.xml"
 navtitle="Sample Task Topic"/>
 </topichead>
 </map>
```

Maps enforce a hard distinction between "the document" and the topics that comprise it. Maps are also the home for various kinds of out-of-line linking in DITA. For example, a map can say that within the context of a particular deliverable, one task is a pre-requisite for another. The publishing toolkit (typically the DITA Open Toolkit) will then inject links into the output to make this relationship visible to readers.

DITA has another construct for enabling more granular reuse within the context of a topic. This is the content reference, or "conref". The conref feature is in some ways similar to the SGML feature of the same name. It allows an element to reuse content from elsewhere through a "conref" attribute. In the following example, a phrase element ("ph") references ("conref") another phrase ("otherphrase") using a DITA-syntax URL ("otherfile.xml#othertopic/otherphrase").

Example 2. Content Reference From One Place To Another

```
somefile.xml:
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="sometopic">
…
<p>…
<ph conref="otherfile.xml#othertopic/otherphrase">Ignored stuff here.</ph>
</p>
…
</topic>

otherfile.xml:
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="othertopic">
…
<ph id="otherphrase">Another phrase</ph>
…
</topic>
```

Note that it is theoretically possible to use this mechanism to create an unmaintainable web of reuse links between documents. `otherfile.xml` could have another element that referenced an element in `somefile.xml`. DITA best practices discourage this, and the user interface of XMetaL Author DITA Edition goes further by helping authors to carve reusable fragments into their own topics called "reusable components".

Reusable components are a specialized kind of topic. They start with a description of the component and some other metadata. Then they have a single element which can be any DITA element. That element is the reusable content. When you refer to it using standard DITA syntax, the component can be embedded in other DITA documents. You can think of them as similar to SGML/XML entities but with more modern syntax and semantics.

Example 3. Reusable Component

```
<?xml version="1.0"?>
<!DOCTYPE ditacomponent
  PUBLIC "-//Blast Radius//DTD DITA Component//EN" "ditacomponent.dtd">
<ditacomponent id="ditacomponent_95A2BE8C03B4405C8A42B9724953DE85">
<description>My reusable paragraph</description><compbody>
<reusable-content>
<p id="p_1"> The ink cartridge connects to the print head,
which contains hundreds of tiny nozzles, each thinner than
a hair. The resolution of the printer is determined by the
number and size of these nozzles. As the print head moves
across the paper, the printer turns the nozzles on and off
very quickly to send the ink onto the paper. Sometimes, piezo
crystals (which change shape when a voltage is applied to
them) are used to propel the ink onto the page. Each tiny
dot made on the paper can have more than one color nozzle
spray ink onto it. The more droplets that are applied,
the higher the quality of the image. Hello IBM.
</p></reusable-content></compbody></ditacomponent>
```

Topics and components are only reusable if you can find them when you need them. DITA has a rich set of constructs for embedding metadata in topics and maps and also for associating metadata with topics. These constructs can be reflected in a content management system to allow users to search for (and find!) components for reuse. DITA metadata can also be used to filter topics out of maps when the maps are being published in certain media, for certain products or for particular audiences.

As you can see, a deliverable in DITA is almost always a complex amalgamation of files connected by links: a compound document. This raises a host of interesting issues for content management deployments.

3. DITA Support in a Content Management System

As described earlier, the defining characteristic of DITA-based content reuse is that reuse relationships are just links between XML documents. This implies that it is more important for a DITA-implementing CMS to support rich linking capabilities than document composition and shredding features. If you are evaluating a CMS that will be used to implement DITA, you should consider a variety of characteristics relating to linking.

First you should determine whether the system can manage an arbitrary number of linking levels. In DITA, it is possible for a map to embed a map which embeds a map which embeds a topic which embeds a reusable component (via conref) which embeds a reusable component and so forth arbitrarily. You cannot expect to succeed with a system that can only support two or three levels of nested linking.

Second, you should consider whether the system has means for reporting on linking relationships. You will want to be able to ask it questions like:

- What objects (topics, maps, content references) reference this object?
- Conversely, what objects does this object reference?
- Which of these relationships are unbreakable dependency relationships and which are less vital? For example, a task pre-requisite may be unbreakable, whereas a "see also" link may not be.

Also: you will want to know whether the CMS extracts metadata information from DITA elements for searches later.

Next, you should consider issues of batch operations.

- Can the CMS do batch operations (check out, check in, move) on documents composed of links ("compound documents")?
- Can the CMS workflow system route complete compound documents through workflows? Does it have a well-thought-out permissions and locking model in this case?

In theory it would be nice for a content management system to have first-class support for DITA specialization. This would mean that if you created a specialized document type through inheritance from a generic one, the content management system would automatically infer appropriate link management behaviours for the specialized document type based upon the configuration it uses for the base document type. Because the concept of DITA specialization is so new, however, it is unrealistic to expect this in most content management systems. Furthermore, this feature is just a labour saving device for the implementation of specialized types. Given this is not an every-day task, the requirement to cut and paste rules from one configuration file to another seems manageable.

In summary, the key thing to evaluate when considering a content management system for DITA is its handling of links and its ability to treat links as reuse relationships.

4. The Documentum Model of XML Content

Documentum's model is certainly compatible with DITA's, but the features of Documentum that are most relevant to DITA implementation are the ones that are least familiar to many Documentum implementors. In particular, Documentum has fairly rich linking constructs even though most people focus on its "chunking" and "bursting" features. All of these features work together and are configured in similar ways.

Documentum is configured to support a new XML document type through a so-called "XML Application". The application consists of:

- an "XML Application configuration" file that defines the rules for processing the XML file
- A folder containing files needed by the client application. These will be downloaded to the client when checking out files:
  - DTD/Schema
  - Stylesheets
  - User interface configurations
  - etc.

The XML Application Configuration controls Documentum's XML import and export behaviour for the document type. It is associated with the document type through doctype, namespace or other similar mechanisms. It controls the following aspects of Documentum's behaviour:

Chunking Rules	Using chunking rules, Documentum can split an XML document into individual objects. For example, it might split a book so that each chapter becomes a repository object. Although this feature is quite powerful and useful in the right contexts, we do not recommend its use for topic-oriented authoring for reasons that will become clear.
Link Recognition and Classification	Using linking rules, Documentum can recognize links on a per-doctype basis and map them into Documentum linking constructs. Links come in two types, parent-child and peer-to-peer. Objects connected using parent-child links can be worked with together as "virtual documents" (Documentum's term for a compound document). Documentum constructs the virtual document by following the parent-child links. Peer-to-peer links are not connected in this manner.
Content Location Selection	The XML Application Configuration can define where chunked or referenced components are imported into the repository. For example, you might say that every image should end up in an "images" directory whereas referenced or chunked topic should end up in a "chunks" directory.
Validation Rules	The XML Application Configuration can require that Documentum validate XML documents on import and reject their import if they do not pass validation against a DTD or Schema.
Metadata extraction	The XML Application Configuration can direct Documentum to map the content of XML elements and attributes to Documentum metadata to enable queries.

5. The Question of Authoring Model

Ultimately, the question of how to implement Documentum revolves around your vision of how authoring and publishing proceeds. For clarity, we will label the two models "document oriented" and the other "fragment oriented".

In the document oriented model, authors check out a compound ("virtual") document. Documentum merges the fragments into a single XML document which authors work with as a unified whole. On checkin, Documentum automatically splits the document into its fragments. This model is ideal if you want to give authors a very seamless environment for authoring documents. In this model, documents are fragmented on the basis of business rules encoded in software. They are also re-assembled by the CMS for publishing.

In the fragment oriented model, authors work primarily on fragments (topics or reusable components). The CMS primarily manages the links between fragments and their containers. Authors determine how many topics to put into a file based upon business policies and pedagogic principles. In this model, it is the responsibility of authoring and publishing applications to give authors views of what entire deliverables will be like. For example, the XMetaL Author DITA Edition allows the user to visualize the hierarchy using the map editor and the DITA Open Toolkit can combine fragments into PDF documents for printing.

You can choose between these models based upon the following criteria:

- do you want fragmentation to be driven by business rules or by authors following conventions?
- do you want your authors to think primarily in terms of fragments or in terms of documents?
- do you have authoring and publishing tools that can work with fragmented information or are they only able to work with full documents?

Because DITA is specifically designed for topic-based authoring, we feel that a focus on fragments is appropriate. DITA is designed specifically to encourage authors to think in terms of topics (fragments) rather than books (complete compound documents). In summary, a fragment-oriented Documentum implementation model reflects and supports DITA's topic-oriented authoring model.

6. Configuring Documentum for DITA

You can use Documentum's XML Application Configuration mechanism to configure Documentum for a fragment-oriented model. First, we must configure parent-child links between maps and topics. For example, to recognize DITA `topicref` elements:

Example 4. Linking Maps to Topics through topicref elements

```
<link_rule ndata="false" peer_or_child="child">
            <!--rule_name:Apply to topicref-->
            <element_selection_pattern>
                <element>topicref</element>
            </element_selection_pattern>
            <link_target>
                <attr_name>href</attr_name>
            </link_target>
            <location>
                <path>topics/</path>
                <object_type/>
            </location>
            <object_type>dita_document</object_type>
</link_rule>
```

On the other hand, cross-references should be expressed using peer-to-peer links:

Example 5. Recognizing DITA cross references

```
<link_rule ndata="false" peer_or_child="peer">
            <!--rule_name:Apply to xref-->
            <element_selection_pattern>
                <element>xref</element>
                <attribute_test>
                    <attr_name>format</attr_name>
                    <value>dita</value>
                </attribute_test>
            </element_selection_pattern>
            <link_target>
                <attr_name>href</attr_name>
            </link_target>
            <permanent_link/>
 </link_rule>
```

Similar techniques can be used to handle images, related-links and so forth.

Content references are a bit different: the appropriate rule syntax will depend upon your organization's business rules about their management. We advise our customers to obey the rule that objects linked by content references form a strict hierarchy. These can be represented in the CMS as parent-child links. DITA itself does not enforce this rule. We also advise XMetaL customers to make content references only to "reusable component topics" rather than to arbitrary elements in generic topics. This improves manageability by allowing authors to know that changes that they make to ordinary elements will not accidentally appear in other documents.

DITA metadata can be reflected into Documentum metadata using a `dctmattr` element in the XML Application Configuration. The question is, what metadata should reside exclusively in the XML, which should reside exclusively in the CMS and which should be mapped back and forth? We suggest to our customers that they use the following rules.

1. If the content is solely for filtering then it should reside exclusively in the XML, because the CMS does not care about filtering.

2. If the metadata is primarily for categorization and searching, then it should live in both the XML and the CMS, to keep your metadata system CMS agnostic.

3. If the metadata is inherently specific to a CMS *or* system-generated and transient (e.g. "Last authored by", "approval status") then it should reside solely in the CMS and not be mapped back and forth.

7. Caveats

At this point it is worth reminding readers that no software is perfect. Even Documentum has limitations and bugs. The deeply nested, fragment-oriented patterns described in this paper are very advanced and not as widely deployed as flatter, document-oriented models. Certain older versions of Documentum have bugs that prevent management of DITA-syntax links. The 5.3 series can recognize the links but there are certain patterns of checkin and checkout that may cause problems. For example, you could run into problems if the same topic is in two maps that are both checked out by the same user at the same time.

Blast Radius is taking a multi-faceted approach to this issue. First, we are working closely with Documentum, communicating DITA-related requirements and issues. Second, we are architecting the XMetaL integration with Documentum to guide users around these issues by disallowing patterns of use that we know to be problematic on a per-version basis. Third, for our customers using Documentum through interfaces other than XMetaL, we document the issues that they should be aware of. If you do not have access to such a list or code to help work around the issues then we would strongly encourage you to do substantial system testing of your Documentum/DITA installation so that you can discover issues of relevance to your users. You should also pay attention to issues reported on EMC Documentum's customer issue tracking system.

Another important thing to keep in mind: a key reason that Documentum supports the "document-oriented" model of authoring and publishing is because it substantially reduces the amount of effort it takes to integrate Documentum with the document-oriented authoring and publishing systems that are still dominant in the world. If you plan to use Documentum in a fragment-oriented, topic-oriented fashion, then you should expect to also deploy authoring and publishing systems that can work directly with linked fragments rather than linear documents. You should ensure that your tools can support maps, topics, DITA-compliant conditional text and content references.

8. Summary

...

Although this paper is focused on Documentum, there are other DITA-compatible Content Management Systems. In fact, the XMetaL team is in close contact with many CMS vendors who have implemented or are thinking about implementing DITA support. Even CMS' without explicit DITA support can often be configured to do a good job with DITA as we've demonstrated with Documentum. We hope that this paper has given you enough background in DITA and Content Management to start the selection process or to validate your CMS selection.

A. Definitions

Check In	Submit updated content to the CMS.
Check Out	Retrieve content from the CMS for authoring.
Chunk	Documentum's term for a fragment of XML that it creates ("chunks") when you import or check in a document. The chunking is done according to rules defined in the XML Application Configuration.

Compound Document	Document made up of multiple parts, such as chapters in a book or topics in an online help system.
Document	... See also, XML Document
Fragment	A part of a compound document, such as a chapter in a book or a topic in an online help system.
Import	Submit new content for management by the CMS.
Object	A bit of content managed by the CMS: essentially a file with metadata attached to it.
Reusable Component	In Blast Radius' implementation of DITA, a fragment smaller than a topic, such as a reused step in a step list or a boilerplate paragraph.
Topic	In DITA, a reusable unit of text that is meaningful on its own.
Virtual Document	Documentum's implementation of compound document.
XML Application Configuration	A configuration file describing how Documentum interprets XML documents on import, checkin, checkout and export.
XML Document	A document as defined in the XML specification. Maps, topics and reusable components are each an independent document from the low-level XML point of view.

Biography

Paul **Prescod**
Group Program Manager
Blast Radius Inc. [http://www.blastradius.com]
XMetaL
1146 Homer Street
Vancouver
British Columbia
V6B 2X6
Canada

Paul Prescod is an implementor of XML-based systems, co-author of the XML Handbook, author of numerous articles on XML and contributor to open source XML tools.