

# Introduction to Structured Content Management with XML

by Kay Ethier and Scott Abel

Organizations of all sizes are beginning to realize how content and its reuse across the enterprise can improve productivity—and the bottom line. The need for change is driven by the desire to better manage information assets (documents, creative ideas, illustrations, charts, graphics, multimedia, etc.) and eliminate costly processes that fail to facilitate the effective and consistent re-use of content. At the heart of managing content for re-use however lies the job of exposing the underlying *structure* of that information.

This article is meant to serve as an introductory primer on how to define and use information structure when managing content.

## The Case for Structure

To be sure, managing structured content -- usually in an XML-based CMS -- comes with a price. The transition to structured content management systems can be quite challenging, especially for authors.

However, there are oftentimes great benefits to structure that make the extra investment worthwhile. Benefits include:

- Making content more retrievable and re-usable;
- Reducing costs and complexity of translation;
- Enforcing authoring, style, and branding guidelines;
- Improving information interchange.

So leveraging structure within documents can be quite valuable, but how exactly do you go about defining structure in the first place?

There are four basic tenets critical to structuring information:

- Defining information types;
- Identifying rules of content hierarchy;
- Creating modular content units;
- Applying standards consistently.

## Defining information types

When you begin to analyze your existing documentation and future requirements, think about your content according to its informational type rather than its format. Procedures, topics, facts, terms, definitions, prices, product numbers, and product descriptions are common information types.

As you continue to analyze the content you create, you will likely discover that many information types are reusable. For instance, you may discover that there is no reason that your product description should be any different regardless of where it is published. You may ask yourself, “Why are we recreating

content that could be reused?” If you don’t have a good business reason for recreating the content, you should consider reuse.

For an excellent primer on content reuse, see “Fundamental Concepts of Reuse” by Ann Rockley, Pamela Kostur, and Steve Manning.

<http://www.managingenterprisecontent.com/>

## **Identifying rules of content hierarchy**

The most significant way that structured documents differ from unstructured ones is that structured documents include “rules.” These rules formalize the order in which text, graphics, and tables may be entered into a document by an author. For example, in an unstructured document, a paragraph has specific formatting—font, size, and spacing. In a structured document, this same paragraph also has an exterior “wrapper” that governs the elements that are allowed to appear before and after it. The elements’ rules are defined in a document type definition (DTD) or Schema -- more about those later.

Structured content management implies moving away from formatting cues to signal such relationships and instead working with information rules. This is where the power of the information model comes from, but also the difficulties in change management, inasmuch as authors are used to working in presentation-based systems.

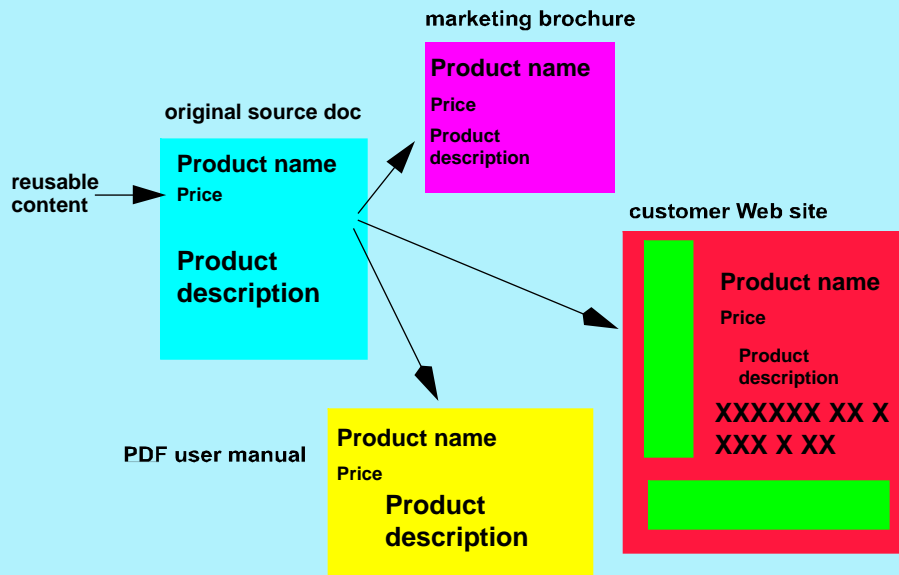
## **Creating modular content units**

Structured content management requires that you begin to look at the content you create as separate, identifiable chunks of information that can be reassembled differently depending on audience, purpose, or delivery method. Note that this represents an intentions-based analysis, and not an academic exercise. How, where, and when you intend to re-use that content should drive your modularity here.

These chunks of information, once identified and tagged, can be reassembled (reused/repurposed) in other information products. They can even be reused in a different order.

The following illustration demonstrates how modular content from a source document could be reused in a marketing brochure, user manual, and customer-facing web site.

## Example: Reusing Chunks of Content - Cell Phone Company



In this simple example, a cellular phone company reuses chunks of information from a source document in a marketing brochure, user manual, and a customer-facing web site. Three information types, product name, price, and product description, are reused from the source document instead of being recreated.

## Using standards consistently

At a subconscious level, you may understand the importance of following internal standards, branding guidelines, and formalized structure. But, it is human nature to continue to find reasons to override templates or alter the format “just this one time.”

Breaking the rules is not allowed when it comes to structured authoring. Reuse is only possible when your information is consistently structured. Imagine how useless a phone directory would be if the data entry clerks at the phone company were allowed to enter information in any order they choose. Some clerks use the first field for first name, others for last name. And instead of last name, first name, ordered alphabetically, what if some of the listings were first name, last name? Talk about a retrieval nightmare.

Of course, most enterprise content is not as highly structured as a phone book. But if your goal is to reuse content, it must be at least be structured *consistently*. If adhering to a particular document standard seems prohibitively painful, re-examine whether the content is really as structured as you think, or recalibrate your expectations about how much information can be re-used and easily shared. Document models can be made “looser” and more flexible, but with a cost in downstream utility of the information.

## XML Building Blocks

So you’ve identified your content types, chunked them into modular components intended for re-use, established the relationships among those chunks, and decided that you can live with them in a

componentized fashion to the extent that your team will follow that structure consistently. Whew! That can be a lot of work, but almost always worth the while.

Now let's delve into some technical details about how structure is implemented in XML -- a language explicitly designed to represent the very hierarchical models you just labored to create. In order to understand structured XML content management you should have a basic understanding of these concepts:

- Elements
- Attributes
- DTDs and Schemas

## Elements

The basic unit of information is called an element. Elements can be text, graphics, tables, or even containers for other elements. In short, everything is an element.

When you create an information model, you define a document hierarchy. A hierarchy specifies the order in which elements are allowed to be used in a particular information product.

For example, in a set of user documentation, a *chapter title* always begins a *chapter*, followed by a *synopsis* and a bulleted list of topics in the chapter. The XML markup example below illustrates the structured hierarchy of the chapter.

```
<chapter>
<chapter title>Using XML</chapter title>
<synopsis>XML is a meta-markup language appropriate for
use when creating structured content.</synopsis>
<ul>
<li>Why XML?</li>
<li>XML and multi-channel publishing</li>
<li>The importance of structure</li>
<li>DTDs</li></ul>
```

**Elements are powerful tools that allow you to create structured content appropriate for reuse—when attributes are added to the mix.**

## Attributes

XML elements can be extended to contain more information than just a label. Elements have *attributes*; additional information about each element. For example, a chapter element can have an optional attribute of *author* and the author's *university* affiliation. These attributes allow us to find all instances of a specific author or university. The XML markup example below illustrates the attributes of *author* and *university*.

```
<chapter author="Chris Smith" university="Duquesne">
<title>Analyzing Satellite Rotation</title>
.
. [rest of document here]
.
</chapter>
```

---

Because you can classify information based on attributes, you can create new information products from your source content that you would otherwise have to cobble together manually. Documentation authors have long benefited from adding attributes to the elements of content they create, allowing readers to traverse “help” applications and user guides more intelligently. Attributes can help indicate in which information products an element should appear, and in which languages. For example, some elements should be present on a web site, but may not be appropriate for a printed guide; others should appear in the Spanish version of a document, but not the Portuguese.

Ponder this for a moment. Attributes make content smart enough to know where to go. For example, elements and attributes can be harnessed to create dynamic content for web-based information products, based on the personal preferences of your users.

## DTDs and Schemas

You define the structure of an information product in a document type definition (DTD) or a schema. A schema, unlike a DTD, is an actual XML document, but both are used widely to define information models (DTDs a bit more in publishing, Schemas somewhat more in industry). Both provide considerable publishing power and can help facilitate content reuse and multi-channel publishing.

For purposes of explanation, we’ll look primarily at DTDs. Here’s an example of the chapter abstract element as defined in a DTD.

```
<!ELEMENT Chapter (Title?, Section, Section+) >
<!ATTLIST Chapter author      CDATA      #IMPLIED
                  university  CDATA      #IMPLIED &#x27E9; ELEMENT
```

Note again that XML content is, for the most part, human readable. In the above example, the first line of the DTD is used to declare an element called Chapter. A chapter title is optional (Title? informs the DTD that a Title is optional in a chapter; it may be used once or not at all), and is followed, if it is present, by one or more section elements (Section, Section+ informs the DTD that a Chapter is followed by two sections or more). Remember, you created the rules; the DTD is just how they got encapsulated.

DTDs and schemas—like your information models they are intended to represent—can be simple or extremely complex. If you’re interested in structured content management and information re-use, take time to learn about XML and DTD/schemas and how they work. You can learn a lot on your own from some of the Web resources listed below. But consider seeking the help of a consultant who specializes in creating structured content before tackling highly complex projects.

## More Reading

### Document Type Definition Tutorial - free online class from w3schools

<http://www.w3schools.com/dtd/default.asp>

### XML Schema Tutorial - free online class from w3schools

<http://www.w3schools.com/schema/default.asp>

### Fundamental Concepts of Reuse - Understanding content reuse (Ann Rockley, Pamela Kostur and Steve Manning)

<http://www.managingenterprisecontent.com/>

## **XML Weekend Crash Course**

<http://www.wiley.com/WileyCDA/WileyTitle/productCd-0764547593.html>

## **About the Authors**

Kay Ethier is an Adobe Certified Expert in FrameMaker 7.x and several prior versions. She instructs training classes, performs consulting, and provides support to clients in a variety of industries. Kay resides in the Research Triangle Park area of North Carolina. In 2001, Kay co-authored the book *XML Weekend Crash Course* (Wiley/HungryMinds). She has most recently been a contributing author on *Advanced FrameMaker* (TIPS Technical Publishing) and *XML and FrameMaker* (Apress).

Scott Abel is a technical writing specialist and content management strategist whose strengths lie in helping organizations improve the way they author, maintain, publish and archive their information assets. Scott is a presenter at industry and professional service seminars, a technical editor, an instructor at Community Learning Network, Indiana University Purdue University at Indianapolis, and immediate past president of the Society for Technical Communication (STC), Hoosier Chapter. Scott is also a member of the Drug Information Association (Document Management, e-CTD, and XML committees), AIIM (the Enterprise Content Management Association), The Asilomar Institute for Information Architecture (“AIfIA”), and a founding member of Content Management Professionals (CM Pros).

## **Contributors**

Ann-Marie Grissino, and George Luke of [Bright Path Solutions](#) contributed to this document.