Leveraging Structured Authoring/DITA Techniques When All You Have are Unstructured Tools

Monique Semp
March 20, 2017
Introduction

- About me
- Presentation summary
  - What are the primary benefits of structured/DITA authoring?
  - The “experts” expectations aren’t required to get the benefits.
  - Examples for FrameMaker, Word, and Flare.
  - References.
Overview

- It’s not...
- Rather, it’s a “things to consider, for a variety of benefits” talk.
- Lots of possibilities and examples.

Outline:

- What is structured authoring, its features, and its toolset features?
- Why isn’t everyone using DITA or DocBook?
- Unstructured, Non-DITA? Adopt Its Features Anyway!
- Unstructured Tools’ Support for Structure
- Examples
- Summary, Concluding Thoughts, References
What Is Structured Authoring?

- **Consistency in structure**—the order of information for a given type of content
- **Consistency in writing.**
- Tools can enforce structure.
- Standards-based frameworks:
  - DITA [http://dita.xml.org/]
  - S1000D [http://public.s1000d.org]
  - DocBook [http://www.docbook.org/].
  - Problems because the specs are gigantic.
  - So your goal must shift.
Structured Authoring Features

Structured enables:

- Easy content reuse
- Implicit focus on topic-based writing
- Semantic styling
Typical tools provide:

- Multi-output publishing
- Portable content
- Metadata support

<Title>, <Author>, <Date>, <Application>...
Why Isn’t Everyone Using DITA or DocBook?

DITA can be a solution that’s looking for a problem.
Unstructured, Non-DITA? Adopt Its Features!

- Writing is writing
  - Topics
  - Short descriptions

- Leverage your authoring tools’ features:
  - Reuse content
  - Style semantically
  - Standardize navigation
  - Add metadata
  - Publish to multi-channels
Unstructured Tools’ Support for Structure

A document’s structure is (somewhat) dependent on the authoring tool:

Adobe FrameMaker

Microsoft Word

MadCap Flare

Dev-Doc tools
Caveats and Notes

- Structure isn’t all or nothing
- Plan for a new toolset/ecosystem and migrating existing content
- Some of my examples are short-term solutions/workarounds
- Not all features are in all a tool’s versions
Examples

1. Create structure—Write topics (not books or chapters); design and adhere to consistency across topics.

2. Write a short description.

3. Reuse content—Whole topics, fragments, and slightly-different content.

4. Style content semantically.

5. Standardize navigation aids.

6. Add metadata.

7. Publish to multiple channels—PDF and non-PDF.
1a. Write Topics: “Doc” Structure

In DITA, structure and topic-based authoring is designed in. But non-structured tools can support structure and topics.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
</table>
| Word (2010) | A couple of approaches:  
  - Create separate files for every topic, and include them in a master doc. Historically, master docs have had problems, but they do seem workable.  
  - DitaExchange integrates with Word, letting you create DITA topics and maps without knowledge of DITA markup. See References page. |
| Flare (11)  | • Based on XHTML, whose specification defines a well-formed document—a document that adheres to the syntax rules specified by the XML 1.0 specification. So structure is inherent in the content. See References page.  
  • Fully and easily supports separate files for topics, which can be included in any number of docs (books, help systems, etc.). |
1b. Write Topics: Consistently Styled

- DITA mechanisms foster consistency.
- You can achieve consistency in non-structured tools, using their features, and applying diligence and editorial review.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Use file naming conventions.</em></td>
<td><em>Use file naming conventions.</em></td>
<td><em>Create topic templates that contain predefined CSS classes, in a given order.</em></td>
</tr>
<tr>
<td><em>Use model <em>template</em> .fm files.</em></td>
<td><em>Use model <em>template</em> .dotx/.dotm files.</em></td>
<td><em>Use a metatag to define the topic type.</em></td>
</tr>
<tr>
<td><em>Configure <em>Next Pgf Tag</em> for all paragraph styles.</em></td>
<td><em>Configure <em>Next Pgf Tag</em> for all paragraph styles.</em></td>
<td><em>Some Flare styles contain logic for the next line’s style; all styles are editable via CSS.</em></td>
</tr>
</tbody>
</table>
Create Topic Template: Flare

1. (Not illustrated) Use the Flare XML Editor to create a new topic file, structure it as desired, and save it to the Flare Template Library folder.

2. From the ribbon’s Tools tab, click Manage Templates to open the Templates Manager for project template linking.

3. When you create a new topic, select the desired topic-template as the New from template source.

Flare Master Projects ensure that a team of authors always has the current templates.
2. Write a Short Description, `<shortdesc>`

- DITA has the `<shortdesc>` element.
  - “Guidelines for Writing Effective Short Descriptions”, in the *DITA Best Practices* book.

- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a <code>&lt;shortdesc&gt;</code> paragraph style, and assign this style as the Next Para for all heading styles.</td>
<td>Create a <code>&lt;shortdesc&gt;</code> paragraph style, and assign this style as the Next Para for all heading styles.</td>
<td>Create a CSS class that identifies a specific variant of an XHTML element.</td>
</tr>
</tbody>
</table>
To create a `<shortdesc>` “element” and configure FrameMaker to automatically “use the structure”:

1. Using the Paragraph Designer, create a `shortdesc` paragraph tag, and set its **Next Pgf Tag** to the “regular” paragraph, such as `body`.

2. For all heading styles, set their **Next Pgf Tag** to the `shortdesc` paragraph style.

When you press **Enter** after typing a heading, the next paragraph is the `shortdesc`, followed by `body`. 
The `<shortdesc>` Element: Word

To create a `<shortdesc>` “element” and configure Word to “add it to the doc”:

1. Create a new paragraph style, `shortdesc`, base it on the “regular” paragraph, such as `body`, and assign its **Style for following paragraph** as `body`.

2. For all heading styles, set their **Style for following paragraph** to the `shortdesc` paragraph style.

When you press **Enter** after typing a heading, the next paragraph is the `shortdesc`, followed by `body`. 
Neither XHMTL nor CSS define a shortdesc element; so create a CSS class:

1. Type a semantic element’s prose into a <P> element.

2. Create a custom CSS class (right-click the class in the structure bar, select Style Class > Create Style Class), p.shortdesc, and apply it to the prose.

Requires manual application; CSS does not support structured behaviors such as next paragraph or prev paragraph.
3a. Reuse Content: Whole Topics

- In DITA.
- With non-structured tools.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Write the topic in a standalone .fm file, and import it as a text inset.</td>
<td>• Master documents.</td>
<td>• Topics can be included in any number of projects.</td>
</tr>
<tr>
<td></td>
<td>• Recent thread from the [Framers] list: <em>Using Frame as a little CMS,</em> <a href="http://www.mail-archive.com/framers%40lists.frameusers.com/msg65018.html">http://www.mail-archive.com/framers%40lists.frameusers.com/msg65018.html</a></td>
<td>• SmartDocs is designed with content reuse in mind; see <a href="http://www.thirtysix.net/">http://www.thirtysix.net/</a></td>
<td>• Included snippets are automatically updated if you update the source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RiverFloe is a &quot;document generation machine for Microsoft Word.&quot; See <a href="http://www.riverfloe.com/">http://www.riverfloe.com/</a></td>
<td></td>
</tr>
</tbody>
</table>
3b. Reuse Content: Fragments

- In DITA.
- With non-structured tools.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Customize system variables.</td>
<td>• Use AutoText to define content fragments. Use the AutoText field code to enable easy doc updates for modified AutoText entries.</td>
<td>• Variables can be used in topics, snippets, and print page layouts.</td>
</tr>
<tr>
<td>• Add user variables.</td>
<td>• Custom variables (in Advanced Properties) let you create true variables.</td>
<td>• You can define variable sets.</td>
</tr>
<tr>
<td>• Easily manage variables via tools such as Leximation BookVars, <a href="http://leximation.com/tools/info/bookvars.php">http://leximation.com/tools/info/bookvars.php</a></td>
<td>• Building blocks provide even more control.</td>
<td>• You can define a set of doc-specific variables (title, version, etc.), and assign them as UPDATE-IN-TARGET. For each target, specify the variable values.</td>
</tr>
</tbody>
</table>
### 3c. Reuse Content: But Change it *Just a Little*

Easy in DITA; possible with non-structured tools, with varying degrees of ease.

<table>
<thead>
<tr>
<th><strong>Frame (2015)</strong></th>
<th><strong>Word (2010)</strong></th>
<th><strong>Flare (11)</strong></th>
</tr>
</thead>
</table>
| • Full-featured conditional text management provided.  
• Best practice: Create a per-project .fm template that has only conditions. Import its **Conditional Text Settings** into all a doc’s .fm files. | • No dedicated function, but by using hidden text, you can fake it; see References.  
• Use SmartDocs’ conditional functionality, [http://www.thirtysix.net/smartdocs/features](http://www.thirtysix.net/smartdocs/features).  
• Use **Document Automation** in Clio, a cloud-based management software system for the legal industry; see References. | Conditional text feature lets you tag content at the:  
• Topic-file level  
• ToC level  
• Target level  
• In-topic content level  
• In snippets |

---

Leveraging Structured Authoring/DITA ... With Unstructured Tools

March 20, 2017 21
4. Style Content Semantically

- Fostered by DITA.
- Typically easy in non-structured tools.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create character and paragraph styles; for example:</td>
<td>Create character and paragraph styles; for example:</td>
<td>Create a CSS class that identifies a specific variant of an XHTML element. For example: a div element, which in turn can specify unique formatting to its child elements.</td>
</tr>
<tr>
<td>• cite</td>
<td>• &lt;cite&gt;</td>
<td></td>
</tr>
<tr>
<td>• uicontrol</td>
<td>• &lt;uicontrol&gt;</td>
<td></td>
</tr>
<tr>
<td>• code</td>
<td>• &lt;code&gt;</td>
<td></td>
</tr>
<tr>
<td>• filepath</td>
<td>• filepath</td>
<td></td>
</tr>
</tbody>
</table>
Create *semantic* char styles instead of using *formatting*. For example, for doc titles, use a *cite* char style instead of applying *italic* formatting:

1. Using the Character Designer, create a *cite* character tag, assign the **Angle** as **Italic**, and assign all other attributes **As Is**.

2. After you type the doc title, select its text, press F8 (the shortcut for **Apply Character Tag**), and type/select *cite*.

Now it’s easy to revise the format of all citations, or to search for all of them.
Create *semantic* char styles instead of using *formatting*. For example, for doc titles, use a `<cite>` char style instead of applying *italic* formatting:

1. Create a new character style, `<cite>`, assign its **Style based on** as *(underlying properties)*, and select italic.

2. After you type the doc title, select its text, press ctrl-shift-s (the shortcut for **Apply Styles**), and type/select `<cite>`.

Now it’s easy to revise the format of all citations, or to search for all of them.
Semantic Tagging: Flare

Neither XHTML nor CSS define semantic tagging except coincidentally (such as cite and code).

1. To mimic semantic tagging, apply a CSS class to a design element, DIV, or SPAN.

Now it’s easy to revise the format of all instances of a given class, or to search for all instances.

Note: Flare does not have a mechanism for enforcing a given sequence of CSS classes. Editorial oversight is required.
5. Standardize Navigation Aids

- Typical in DITA setups; easy to use.
- Not difficult to set up with non-structured tools.
- Implementation-By-Tool:

|--------------|-------------|------------|
| Create paragraph styles such as `prereq` and `postreq`. In the Paragraph Designer:  
  • Use the Autonumber Format to specify the lead-in text, such as “Prerequisites” and “What to do next”.  
  • Assign the Next Pgf Tag to a bullet style paragraph. | Create paragraph styles such as `prereq` and `postreq`. Use **outline numbering** to specify the lead-in text, such as “Prerequisites” and “What to do next”.  
  (For help creating numbered styles that display lead-in text instead of numbers, see the References page.) | Flare has built-in features for creating browse sequences, related topics, and more. |
6. Add Metadata

- In DITA.

- With unstructured, “real” authoring tools (vs. simple text editors).

- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Supports Extensible Metadata Platform (XMP).</td>
<td>• Add the <strong>Prepare Document</strong> function to the ribbon (see References), and configure the (Document) <strong>Properties</strong>.</td>
<td>Includes a variety of built-in metadata controls:</td>
</tr>
<tr>
<td>• Use the <strong>File Info</strong> feature for book and/or file metadata.</td>
<td>• To add additional metadata (or for variable use), access Document Properties &gt; Advanced Properties &gt; Custom tab.</td>
<td>• Topic types</td>
</tr>
<tr>
<td>• For historical info, see <a href="http://www.iptc.org/std/Iptc4xmpCore/1.0/documentation/Iptc4xmpCore_1.0-doc-CpanelsUserGuide_13.pdf">http://www.iptc.org/std/Iptc4xmpCore/1.0/documentation/Iptc4xmpCore_1.0-doc-CpanelsUserGuide_13.pdf</a></td>
<td></td>
<td>• Custom file tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Any XHTML-compliant tagging structure that you want to create.</td>
</tr>
</tbody>
</table>
FrameMaker supports the Extensible Metadata Platform (XMP).

To set the metadata at the book level:

1. In the book panel, right-click the book file and select **File Info**.

2. Enter the information for the metadata fields and click **Set**.

Generally when you Save or Publish the FrameMaker book, the **File Info** is automatically propagated to the output.
To enable metadata management, add the Prepare Document function to the ribbon (see References). Then add metadata:

1. (a) From the ribbon’s Prepare tab, click Prepare > Properties. (b) Click the Document Properties list and select Advanced Properties.

2. In the Summary tab, enter the metadata, and click OK.

Depending on your publishing workflow, the metadata can be automatically propagated to the output.
Flare provides a variety of Flare-specific metadata constructs.

- Use a topic’s **Topic Properties** (figure 1) to specify a topic style class (topic type) and to add a description (freeform text).

- Use **File Tags** (figure 2) to assign one or more attributes to topics.
Flare lets you add XHTML meta elements to identify and assign structured metadata
(HTML <meta> tags) to a document: author, expiration date, key words, and so on.

Use a text editor (ideally one that recognizes XHTML structure) to manually add metadata to a Flare topic XHTML file.

Flare includes all metadata in online targets, which enables it to manage project content and to add metadata and other tagging to online outputs.
7a. Multi-Channel Publishing: PDF

- Difficult with non-FrameMaker DITA!
- Easy with unstructured tools.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Polished, highly-customized output is “easy.” You need:</td>
<td>Polished, highly-customized output is “easy.” You need:</td>
<td>Good-quality PDF is easy out-of-the-box. Not excellent because:</td>
</tr>
<tr>
<td>• Good visual design(er).</td>
<td>• Good visual design(er).</td>
<td>• Flare PDF engine does not recognize every CSS element or attribute; for example, CSS <code>text-overflow</code>, <code>white-space</code>, and <code>overflow</code> attributes.</td>
</tr>
<tr>
<td>• Good templates.</td>
<td>• Good templates.</td>
<td>• Flare can't process PostScript fonts.</td>
</tr>
<tr>
<td>• Disciplined style use.</td>
<td>• Disciplined style use.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Professional PDF creator (Acrobat or Nuance).</td>
<td></td>
</tr>
</tbody>
</table>

Leveraging Structured Authoring/DITA ... With Unstructured Tools
### 7b. Multi-Channel Publishing: Non-PDF

- DITA tools excel.
- Varied support in unstructured tools.
- Implementation-By-Tool:

<table>
<thead>
<tr>
<th><strong>Frame (2015)</strong></th>
<th><strong>Word (2010)</strong></th>
<th><strong>Flare (11)</strong></th>
</tr>
</thead>
</table>
| The File > Publish feature lets you create Responsive HTML 5, Mobile App, WebHelp, EPUB, Kindle, and more. You can configure many aspects of the transformation:  
  • Style mapping from FrameMaker styles (para, char, etc.) to CSS classes.  
  • Topic split/merge settings. | Word has a Save As Web Page function, but it’s widely derided.  
  • Best approach is a 3rd party solution such as WebWorks ePublisher, [http://www.webworks.com/](http://www.webworks.com/), to enable single-sourcing for multiple outputs. | Easy to produce a variety of online, print outputs, and even Microsoft Word (which can be helpful to collect review comments).  
  • Supports CSS @media statements, which let you assert different CSS values based on media type or display type. See References. |
Summary: Tools† Support‡ for Features

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic-based structure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consistent topic structure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;shortdesc&gt; mimic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content reuse: whole topics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content reuse: fragments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content reuse: slight diffs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic styling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Para-styled navigation aids</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-channel publishing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

†May require 3rd party tool integration
‡As detailed in this presentation
Concluding Thoughts

- Full-on structured writing is based on useful principles, is demanding, and parts can be leveraged and adopted.
- Always look to your tool’s advanced features to gain structure/DITA benefits.
- Recognize the tradeoffs between structured/DITA and unstructured tools.
Some links are previously mentioned in the presentation. All links valid at presentation time.

**Structured Authoring**


- *DITA Best Practices, A Roadmap for Writing, Editing, and Architecting in DITA*; by Laura Bellamy, Michelle Carey, Jenifer Schlotfeldt, © 2012

**Metadata**

- Uses for metadata: [https://en.wikipedia.org/wiki/Metadata#Creation](https://en.wikipedia.org/wiki/Metadata#Creation)

Standards and Specifications

- DocBook: http://www.docbook.org/
- DITA: http://dita.xml.org/
- S1000D: http://www.s1000d.net/
- XHTML, CSS specifications: https://www.w3.org/
- RELAX NG schema language for XML: https://en.wikipedia.org/wiki/RELAX_NG
Tools
- oXygen XML Editor: https://www.oxygenxml.com/
- DITA Open Toolkit: http://www.dita-ot.org/
- WebWorks ePublisher: http://www.webworks.com

FrameMaker-Related Links
- Using Frame as a little CMS: http://www.mail-archive.com/framers@lists.frameusers.com/msg65018.html
- BookVars plugin to manage variables: http://leximation.com/tools/info/bookvars.php
- FrameUsers.com website and email group: http://www.frameusers.com/community/
Microsoft Word-Related Links:

- MS Word Helpers group on LinkedIn: https://www.linkedin.com/groups/1851284/
- Detailed information about Microsoft Word Building Blocks: http://gregmaxey.mvps.org/word_tip_pages/building_blocks_autotext.html
Microsoft Word-Related Links, continued:

- How to fake conditional text in Microsoft Word: http://www.technicalcommunicationcenter.com/2013/12/05/how-to-use-ms-words-hide-text-function-to-create-conditional-text-in-word/

- SmartDocs, which provides single-sourcing, variables, conditional text features, and more for Microsoft Word: http://www.thirtysix.net/, http://www.thirtysix.net/smartdocs/features

- RiverFloe, a “document generation machine for Microsoft Word”: http://www.riverfloe.com


- Details about creating paragraphs with automatic lead-in text: http://www.shaunakelly.com/word/numbering/numbering20072010.html; look for “Now we tell Word about the numbering itself for Level 1”.
Microsoft Word-Related Links, continued:

- Blog post about using Git for Word docs: [http://blog.martinfenner.org/2014/08/25/using-microsoft-word-with-git/](http://blog.martinfenner.org/2014/08/25/using-microsoft-word-with-git/) (this link is not resolving at the moment, but perhaps it will return)

Flare-Related Links

- Using CSS @media statements: [https://css-tricks.com/snippets/css/media-queries-for-standard-devices/](https://css-tricks.com/snippets/css/media-queries-for-standard-devices/)
- Users of MadCap Flare group on LinkedIn: [https://www.linkedin.com/groups/86373](https://www.linkedin.com/groups/86373)
I encourage you to share these ideas (and slides) with your colleagues—the more people adopt best practices, the easier writing becomes!

Any questions?

Tell me how it goes: