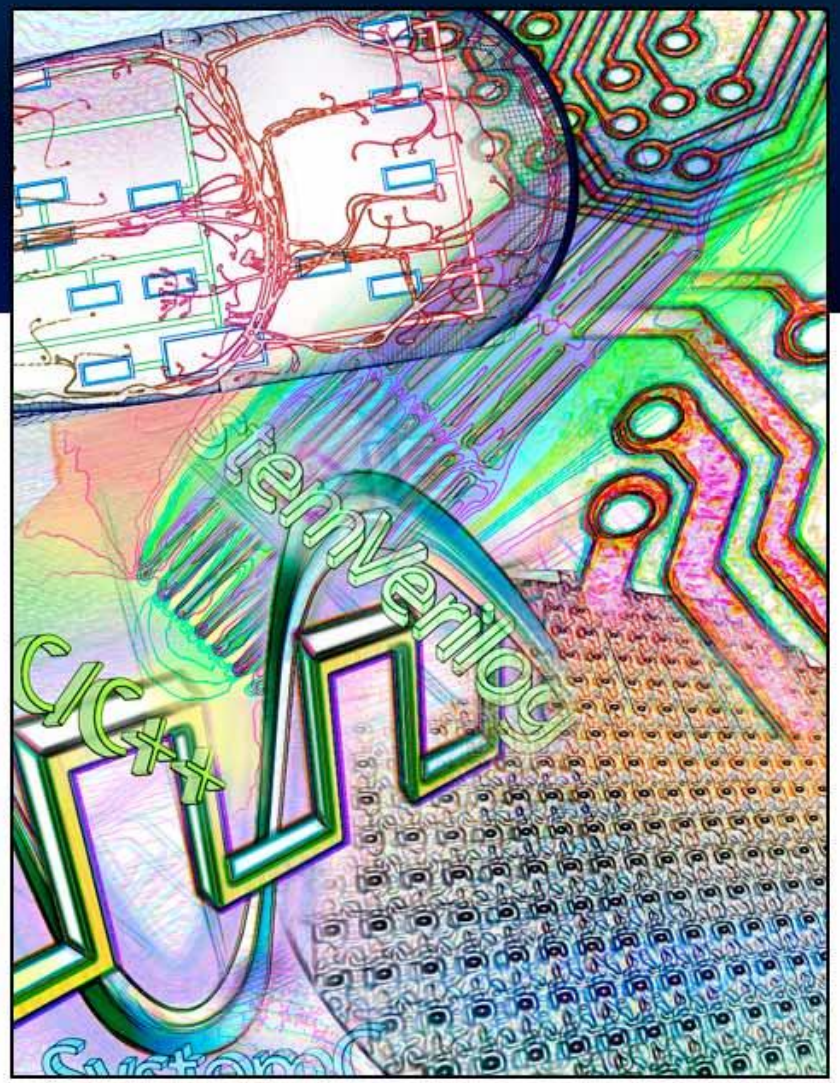


Structured Authoring Training: 7 Tips

Sarah Leritz-Higgins

Manager, Technical Publications

September 2016

Agenda

- We have 15 minutes; let's get right to it!
  - Quick summary of our environment
  - Quick timeline of how we got to structure
  - 7 tips

Corporate Environment

- Centralized Tech Pubs group that serves the entire corporation
  - Includes an infrastructure team, dedicated to structured app & support
- Dedicated personnel to train the writers "in-house"

Authoring and Publishing Environment

- Standard: Customized DITA 1.2
- Authoring tool: FM12 (structured) with DITA-FMx plugin
- Deliverables: PDF and HTML
  - We also do videos (MP4)!
- Channels: local installation and web
- CCMS: none (some teams use source control)

Getting to Structure: Our Timeline

- 2008: Defined and published our content models
- 2008/2009: Offered training classes on using content models
- 2010-2013: Reinforced the benefits of topic-based authoring
- 2014:
  - Conducted pilot: migrated a set of books to structure
  - Created a new internal guide, "Structured Authoring Guide" (SAG)
  - Developed the curriculum for a "Structured Authoring" internal class
  - Trained a small team of writers on the new structured system
- 2015: Migrated content to structure & trained the entire staff
- 2016: Lived to tell about it!

Tip 1: Choose Evangelists and Train Them First

- Select a small team of writers to serve as "guinea pigs"
  - This is your pilot team; critically important that this team is:
    - Technically skilled
    - Eager to embrace change
    - Positive and can "spread the word" about the benefits of structure
    - Willing to exert some blood, sweat and tears

These writers are key to your migration to structure!

Tip 2: Eat Your Own Dog Food

- Create an authoring guide with your structured environment
- Develop a training curriculum based on your expertise
  - In-house expertise can only be developed by using the authoring system

Use it so you can teach it!

Tip 3: Offer the Training Classes "In Person"

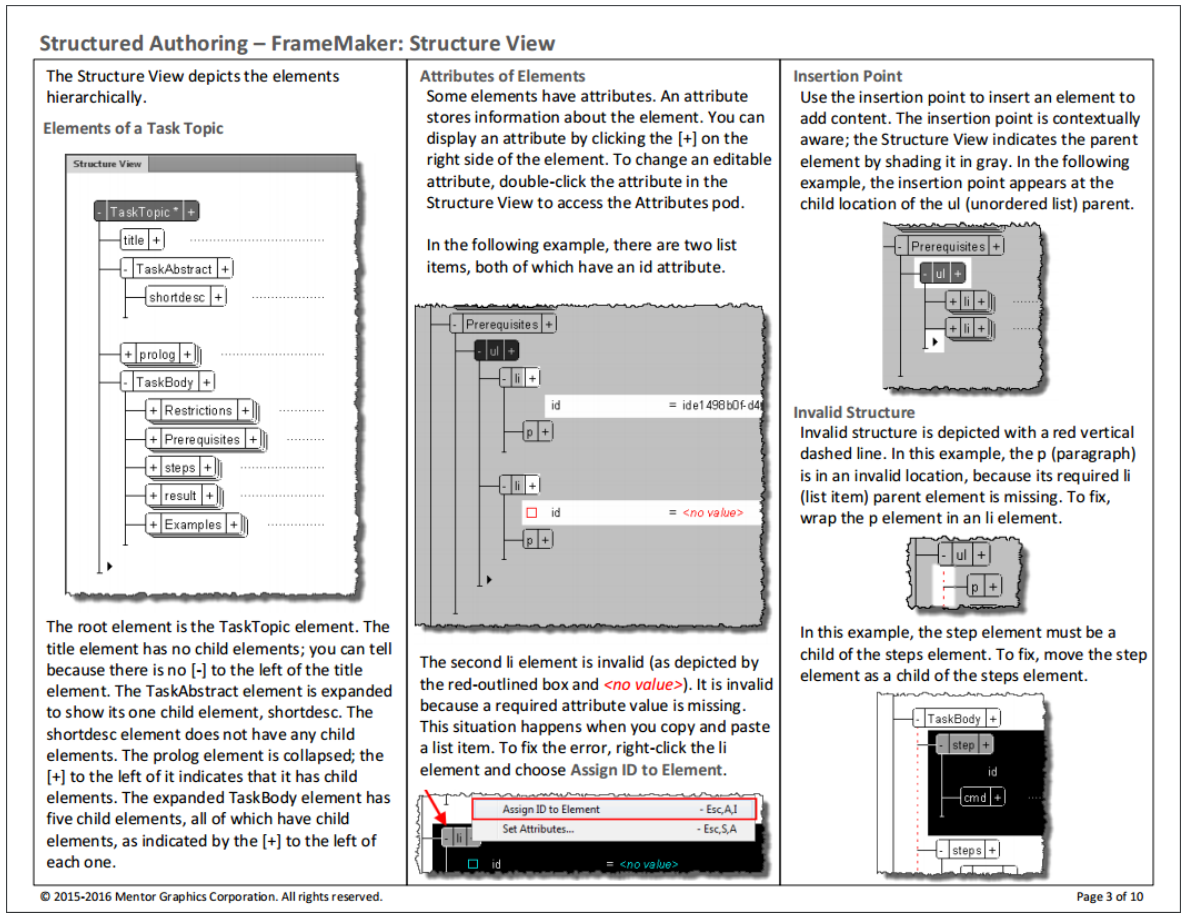
- Plan the logistics and budget
- Ensure the classes are "hands on" – develop lab materials
  - Learning by doing is critical for structured authoring

Webinars are "ok" but nothing beats a real classroom!

Tip 4: Create Quick Reference Cards

- Give writers something tangible that's particularly useful

Sample cards available!



Structured Authoring – FrameMaker: Structure View

The Structure View depicts the elements hierarchically.

Elements of a Task Topic

Structure View

- TaskTopic * +
  - title +
  - TaskAbstract +
    - shortdesc +
  - prolog +
  - TaskBody +
    - Restrictions +
    - Prerequisites +
    - steps +
    - result +
    - Examples +

The root element is the TaskTopic element. The title element has no child elements; you can tell because there is no [-] to the left of the title element. The TaskAbstract element is expanded to show its one child element, shortdesc. The shortdesc element does not have any child elements. The prolog element is collapsed; the [+] to the left of it indicates that it has child elements. The expanded TaskBody element has five child elements, all of which have child elements, as indicated by the [+] to the left of each one.

Attributes of Elements

Some elements have attributes. An attribute stores information about the element. You can display an attribute by clicking the [+] on the right side of the element. To change an editable attribute, double-click the attribute in the Structure View to access the Attributes pod.

In the following example, there are two list items, both of which have an id attribute.

- Prerequisites +
  - ul +
    - li +
      - id = ide1498b0f-d4
      - p +
    - li +
      - id = <no value>
      - p +

The second li element is invalid (as depicted by the red-outlined box and <no value>). It is invalid because a required attribute value is missing. This situation happens when you copy and paste a list item. To fix the error, right-click the li element and choose Assign ID to Element.

- li
  - Assign ID to Element - Esc,A,I
  - Set Attributes... - Esc,S,A
  - id = <no value>

Insertion Point

Use the insertion point to insert an element to add content. The insertion point is contextually aware; the Structure View indicates the parent element by shading it in gray. In the following example, the insertion point appears at the child location of the ul (unordered list) parent.

- Prerequisites +
  - ul +
    - li +
    - li +

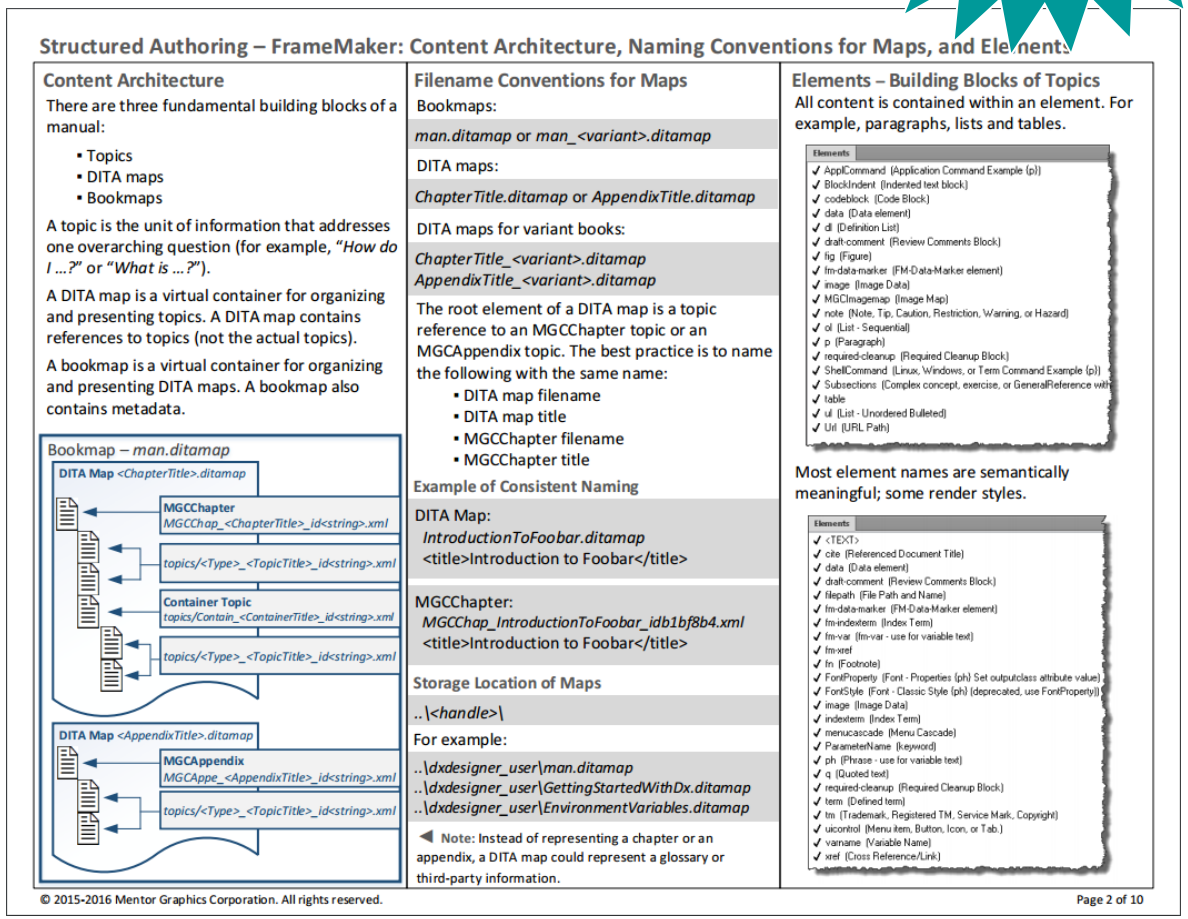
Invalid Structure

Invalid structure is depicted with a red vertical dashed line. In this example, the p (paragraph) is in an invalid location, because its required li (list item) parent element is missing. To fix, wrap the p element in an li element.

- ul +
  - p +

In this example, the step element must be a child of the steps element. To fix, move the step element as a child of the steps element.

- TaskBody +
  - step +
    - id
    - cmd +
  - steps +



Structured Authoring – FrameMaker: Content Architecture, Naming Conventions for Maps, and Elements

Content Architecture

There are three fundamental building blocks of a manual:

- Topics
- DITA maps
- Bookmaps

A topic is the unit of information that addresses one overarching question (for example, *"How do I ...?"* or *"What is ...?"*).

A DITA map is a virtual container for organizing and presenting topics. A DITA map contains references to topics (not the actual topics).

A bookmap is a virtual container for organizing and presenting DITA maps. A bookmap also contains metadata.

Bookmap – *man.ditamap*

DITA Map <ChapterTitle>.ditamap

- MGCChapter — MGCChap_<ChapterTitle>_id<string>.xml
- topics/<Type>_<TopicTitle>_id<string>.xml
- Container Topic — topics/Contain_<ContainerTitle>_id<string>.xml
- topics/<Type>_<TopicTitle>_id<string>.xml

DITA Map <AppendixTitle>.ditamap

- MGCAppendix — MGCAppe_<AppendixTitle>_id<string>.xml
- topics/<Type>_<TopicTitle>_id<string>.xml

Filename Conventions for Maps

Bookmaps:

man.ditamap or *man_<variant>.ditamap*

DITA maps:

ChapterTitle.ditamap or *AppendixTitle.ditamap*

DITA maps for variant books:

ChapterTitle_<variant>.ditamap
AppendixTitle_<variant>.ditamap

The root element of a DITA map is a topic reference to an MGCChapter topic or an MGCAppendix topic. The best practice is to name the following with the same name:

- DITA map filename
- DITA map title
- MGCChapter filename
- MGCChapter title

Example of Consistent Naming

DITA Map:
IntroductionToFoobar.ditamap
<title>Introduction to Foobar</title>

MGCChapter:
MGCChap_IntroductionToFoobar_idb1bf8b4.xml
<title>Introduction to Foobar</title>

Storage Location of Maps

..\<handle>

For example:

..\dxdesigner_user\man.ditamap
..\dxdesigner_user\GettingStartedWithDx.ditamap
..\dxdesigner_user\EnvironmentVariables.ditamap

◀ Note: Instead of representing a chapter or an appendix, a DITA map could represent a glossary or third-party information.

Elements – Building Blocks of Topics

All content is contained within an element. For example, paragraphs, lists and tables.

Elements

- ✓ ApplCommand (Application Command Example (pl))
- ✓ BlockIndent (Indented text block)
- ✓ codeblock (Code Block)
- ✓ data (Data element)
- ✓ dl (Definition List)
- ✓ draft-comment (Review Comments Block)
- ✓ fig (Figure)
- ✓ fm-data-marker (FM-Data-Marker element)
- ✓ image (Image Data)
- ✓ MGCImagemap (Image Map)
- ✓ note (Note, Tip, Caution, Restriction, Warning, or Hazard)
- ✓ ol (List - Sequential)
- ✓ p (Paragraph)
- ✓ required-cleanup (Required Cleanup Block)
- ✓ ShellCommand (Linux, Windows, or Term Command Example (p))
- ✓ Subsections (Complex concept, exercise, or GeneralReference with
- ✓ table
- ✓ ul (List - Unordered Bulleted)
- ✓ Url (URL Path)

Most element names are semantically meaningful; some render styles.

Elements

- ✓ <TEXT>
- ✓ cite (Referenced Document Title)
- ✓ data (Data element)
- ✓ draft-comment (Review Comments Block)
- ✓ filepath (File Path and Name)
- ✓ fm-data-marker (FM-Data-Marker element)
- ✓ fm-indexterm (Index Term)
- ✓ fm-var (fm-var - use for variable text)
- ✓ fm-xref
- ✓ fn (Footnote)
- ✓ FontProperty (Font - Properties (ph) Set outputclass attribute value)
- ✓ FontStyle (Font - Classic Style (ph) (deprecated, use FontProperty))
- ✓ image (Image Data)
- ✓ indexterm (Index Term)
- ✓ menucascade (Menu Cascade)
- ✓ ParameterName (keyword)
- ✓ ph (Phrase - use for variable text)
- ✓ q (Quoted text)
- ✓ required-cleanup (Required Cleanup Block)
- ✓ term (Defined term)
- ✓ tm (Trademark, Registered TM, Service Mark, Copyright)
- ✓ uicontrol (Menu item, Button, Icon, or Tab.)
- ✓ varname (Variable Name)
- ✓ xref (Cross Reference/Link)

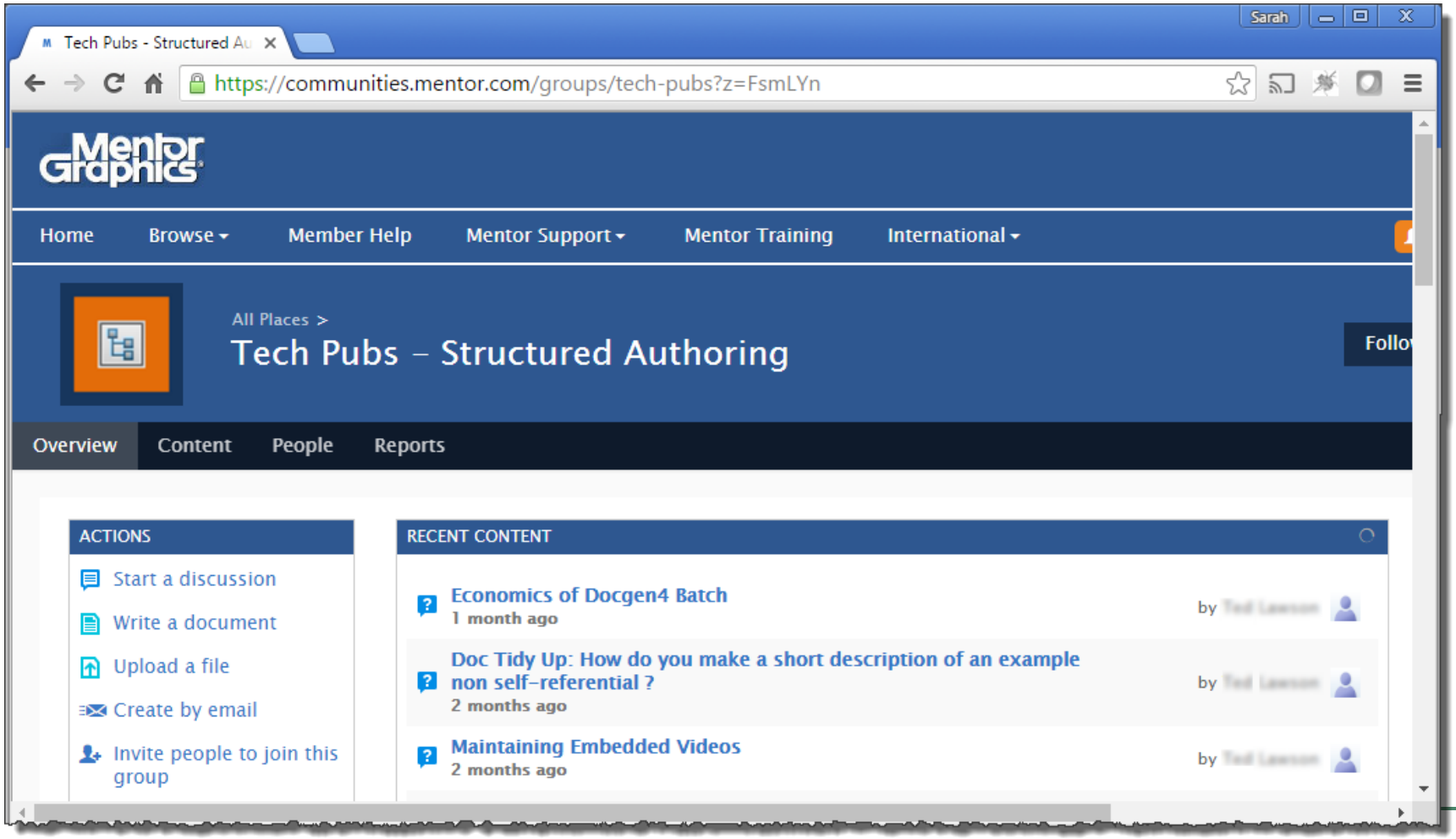
Tip 5: Offer "Continuing Education" Webinars

- After the core training, offer regularly-scheduled 1-hour webinars
  - Cover new features and fixes
  - Provide an "open forum" for Q & A and demos
  - Record and publish these sessions! (Related to tip 7.)

You can't cover everything in one class; keep the momentum going!

Tip 6: Establish a Communities Site

- Empower writers to post questions, provide feedback and help one another



Tip 7: Record and Publish Each Training Module

- Enables new-hires and veterans alike to review the material

Give writers tools for "self-help"

Summary of Tips

- Choose Evangelists and Train Them First
- Eat Your Own Dog Food
- Offer the Training Classes “In Person”
- Create Quick Reference Cards
- Offer “Continuing Education” Webinars
- Establish a Communities Site
- Record and Publish Each Training Module

THANK YOU!