Structuring Information Before DITA

Boston DITA Users Group

Steven Jong, September 2020

Steven Jong

about.me/stevefjong

- MS, Science Communication, Boston University
- Extensive experience (template-design level) writing in pre- and non-DITA environments
- Ten year's experience with three DITA implementations at three companies

First, I'd like to explain why I think I'm qualified to share my experience and perspective with you today
I have a master's degree from a journalism school
I created departmental templates (styles proliferate fast when nested)
(CMS: Vasont, SDL, SDL; Editor: XMetaL and oXygen)
I could say more, but for the purposes of this presentation, these are the pertinent bits

“Where do I begin?”

“Where do I begin?”
I don’t mean where do I begin my presentation, but where does a writer begin in the DITA environment?
When new writers in a mentoring setting ask me where to begin with DITA, it’s hard for me to answer
DITA is not a tool thing or a style guide thing
“You don't have to know [all the characteristics of DITA] to use DITA, but if there is no one in your organization who knows why you should use them, you may have a problem”—Bob Doyle, “History of DITA”

In my experience, some writers exposed to DITA never get the hang of it—this is what I’ve seen:
- Some quit rather than use it
- Some get fired for not grasping the concepts (and making every topic a concept)
- Some can never give up separating format from content
- Some are overwhelmed by the number of choices (Q: how many semantic elements in DITA?)
- Some try to start from scratch (a blank file) and have no idea how to proceed
- Some think you can code DITA like HTML or CSS
- Some hear the familiar words “concept” and “task,” push the instructor aside, and say, “Yah, yah, yah, everyone knows this

stuff—I'm a power Word user, so I'll pick this up in two hours"

If you don't know what you're doing, the tool you use won't help you
If you *do* know what you're doing, the tool you use isn't crucial (I have published from Word, PowerPoint, and Excel)
So: where do you begin? *Organize and structure the information.* What's the best organizing tool?

The most powerful organizing tool is the writer's mind

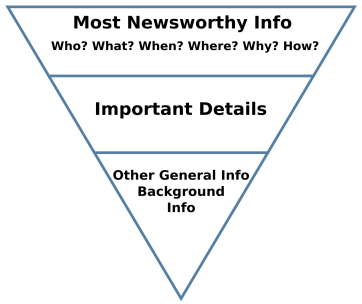
Instead of starting with the first XML tag to include in a document, you have to start with how you're going to organize the information you've collected
Design work is top down, not bottom up

So let's talk about different principles for collecting and organizing information,
some practical methodologies developed to get there,
some tools that helped us in the past,
and how DITA owes its existence to, and is designed around, these principles and methodologies

Collecting Information:

The lost art of journalism

- Who, what, when, where, why, how (5Ws + H)
- Lede (lead) paragraph
- Inverted pyramid



By Inverted_pyramid.jpg: The Air Force Departmental Publishing Office (AFDPO)derivative work: Makeemlighter - Inverted_pyramid.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=12793490

I want to start here: how do you collect information?

If you're lucky enough to get a functional specification to work with these days, it tends to describe a single new feature or product including all the developer can say about it, all jammed together
It's easy and tempting to copy and paste the information into a document as is (after all, every page is Page One)
But not all information in a spec is visible, important, or even necessary to users
One of the most difficult things I've learned to do is *not* include something from a spec
So what information *should* you include?

Newspaper reporters learn the skill of collecting important information and organizing it for easy consumption by readers (Sometimes both writer and reader are under intense time pressure!)

It's a mechanical skill you can learn:

1. Ask pertinent questions
2. Sort the answers by importance

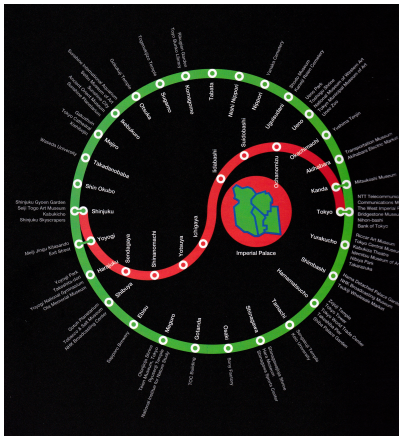
3. Write simple, clear sentences

A *lede* (lead paragraph) summarizes the story in one sentence, so if readers only get that far they still get the gist of the story
Then important details, typically one or two sentences per paragraph
Articles are written assuming they might be cut from the end for space (Q: why?), so the most important elements are mentioned as soon as possible (*inverted pyramid*)
Today we would call the lead paragraph a summary or abstract, and the organizing approach *progressive disclosure*
But we didn't invent the principle—it dates back to the American Civil War (Q: why?)
And information naturally breaks down into articles based on print space constraints and the readers' attention spans (such as the length of a subway ride)

We're big on explaining how to do things, but even in journalism you need to keep your audience in mind and explain *why* something is newsworthy
("Free dental care" story pitched in Journalism 101 in grad school)
(If you want to discuss something that's not part of the article proper, you write a *sidebar*, which could be considered a reference or concept topic)

Organizing Information:
Information architecture

- Richard Saul Wurman
- Organizations (LATCH):
  - Location
  - Alphabet
  - Time
  - Category
  - Hierarchy



Tokyo Access Guide, in *Information Architects*

Once you've collected information, are there other ways to organize it?

Richard Saul Wurman invented the term "information architecture," and claimed these were the only possible organizations:

- By location: "You are here," schematic map, hardware breakdown, software architecture diagram
- Alphabetically: arbitrary but very common (phone directory, dictionary); error messages numerically, with no need to organize further (Q: How are Chinese dictionaries organized?)
- In time sequence or order: do this first, this second (Intellivision "Bomb Squad"), as in a task
- By category: can include audience (installer, administrator, user, programmer) or function; Dewey Decimal System
- By hierarchy (taxonomy): outlining; the writers in the group that first hired me were demon outliners (never single subitems); but most of it was connective tissue ("This section contains the following topics") without content

Information Architects is a gorgeous book, full of aesthetically pleasing illustrations such as this one:
the Tokyo railway stops, arranged around the Imperial Palace not in GPS-accurate positions but arranged symmetrically in a familiar shape

Structuring Information: Methodologies

This is good theory, but as workaday writers we need practical advice

Let's take a historical look at the development of four information structuring methodologies that predate DITA
Three of them are direct ancestors of DITA; if you don't explicitly or implicitly know about them, you're going to struggle with DITA

Sequential Thematic Organization of Publications (STOP)

John Tracey, David Rugh, and Walter Starkey (1965)

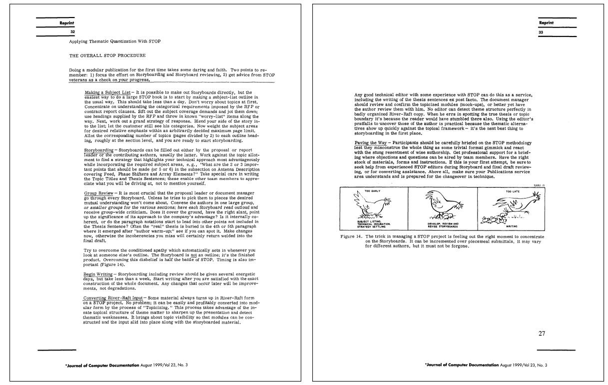
- Hughes Aircraft, Fullerton Ground Systems Group, CA (radar)
- *STOP - How to Achieve Coherence in Proposals and Reports*
- Two-page “topical units of discourse”:
  - Meaningful heading
  - Thesis statement
  - 500-word text limit
  - Illustration (required) close to relevant text
- Arranged in a sequence of themes
- “Storyboard wall,” reviewable immediately

Proposal writers at Hughes Aircraft
Hughes was chasing Cold War money, so they responded to a lot of RFPs
Each response required hundreds of pages (Dad)—too much for them to handle
They recognized that responses had many standard elements, if they could find a way to reuse them
Their solution was *topics* (inspired by index cards), with elements as shown
Arranged in a sequence of themes as a *storyboard* (Q: suggested by Hollywood dilettante Howard Hughes?)
Storyboard posted on a wall and reviewable
The STOP team also introduced the concept of a topic template

These guys did great work! Their ideas live on today
This is the earliest mention of topics that I’m aware of

STOP Topic

Sample



https://faculty.washington.edu/farkas/TC510-Fall2011/TraceySTOPReport-DF-Ann.pdf

The original report on the STOP methodology is in STOP format, and it is delightful—a page-turner!

(This is not even the funniest illustration)

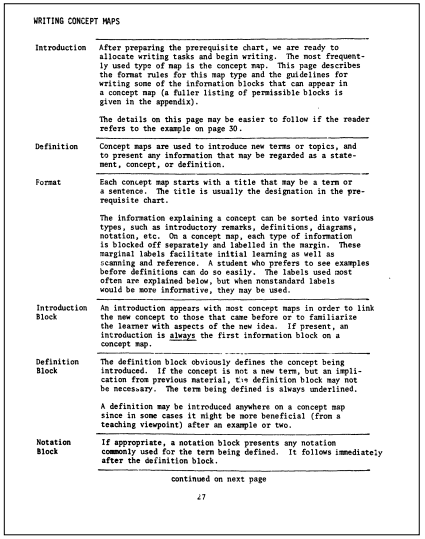
The URL points to a reprint in the *Journal of Computer Documentation*, 23 (3), August 1999 annotated by Professor David Farkus of the University of Washington

Information Mapping™

Robert E. Horn (1967)

- Information blocks classified in several types; presented in labeled chunks
- 7 ± 2 elements
- Blocks grouped into one of seven map types; presented on separate pages
- Integrated graphics
- Maps grouped into documents

Information Mapping for Learning and Reference (1969)



WRITING CONCEPT MAPS

Introduction	After preparing the prerequisite chart, we are ready to allocate writing tasks and begin writing. The most frequently used type of map is the concept map. This page describes the format rules for this map type and the guidelines for writing some of the information blocks that can appear in a concept map (a fuller listing of permissible blocks is given in the appendix). The details on this page may be easier to follow if the reader refers to the example on page 30.
Definition	Concept maps are used to introduce new terms or topics, and to present any information that may be regarded as a statement, concept, or definition.
Format	Each concept map starts with a title that may be a term or a sentence. The title is usually the designation in the prerequisite chart. The information explaining a concept can be sorted into various types, such as introductory remarks, definitions, diagrams, notation, etc. On a concept map, each type of information is blocked off separately and labelled in the margin. These marginal labels facilitate initial learning as well as scanning and reference. A student who prefers to see examples before definitions can do so easily. The labels used most often are explained below, but when nonstandard labels would be more informative, they may be used.
Introduction Block	An introduction appears with most concept maps in order to link the new concept to those that came before or to familiarize the learner with aspects of the new idea. If present, an introduction is always the first information block on a concept map.
Definition Block	The definition block obviously defines the concept being introduced. If the concept is not a new term, but an implication from previous material, the definition block may not be necessary. The term being defined is always underlined. A definition may be introduced anywhere on a concept map since in some cases it might be more beneficial (from a teaching viewpoint) after an example or two.
Notation Block	If appropriate, a notation block presents any notation commonly used for the term being defined. It follows immediately after the definition block.

continued on next page

Horn formalized STOP

His research showed that all information could be presented as a series of blocks collected into maps

In his proprietary format, blocks were labeled and separated by horizontal lines

Horn claimed there were only seven map types:

- Concept
- Procedure
- Process (Q: what's the difference between a procedure and a process?)
- Principle
- Fact
- Structure
- Classification

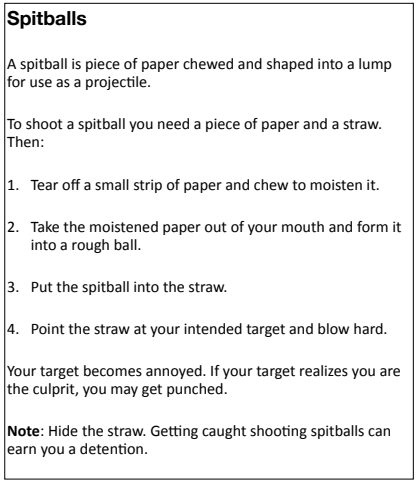
This sample was achieved using nothing more than a typewriter

It's painful to look at today

Task Orientation

Fred Bethke *et al* (1981)

- *IBM Improving usability of publications*
- Classically, a task has four elements:
  1. Starting point
  2. Actor
  3. Action(s)
  4. Endpoint



Spitballs

A spitball is piece of paper chewed and shaped into a lump for use as a projectile.

To shoot a spitball you need a piece of paper and a straw. Then:

1. Tear off a small strip of paper and chew to moisten it.
2. Take the moistened paper out of your mouth and form it into a rough ball.
3. Put the spitball into the straw.
4. Point the straw at your intended target and blow hard.

Your target becomes annoyed. If your target realizes you are the culprit, you may get punched.

Note: Hide the straw. Getting caught shooting spitballs can earn you a detention.

The next advance was the study of tasks at IBM
In the original formulation, all four elements were required (as in this example)

I went to STC conferences in the 1980s when IBM writers described this methodology
Believe it or not, this was novel at the time
(Q: can you apply this methodology to software wizards?)

If I were writing this procedure before I know about task theory I probably wouldn't have thought to include the starting and ending point
Today, minimalism dictates only an implicit actor and steps; consequently, it's often ambiguous where to start and what success looks like

The elements of a task are tightly structured
If you include actor steps and system responses, you can see all the common elements of a DITA task topic
Indeed, this is the parent of the DITA task, which encodes task orientation in the tags

Structured Documentation

Dr. Edmund Weiss

- *How to Write a Usable User Manual* (1985)
- Both a product (structured doc) and a process for creating it
- Two-page modules (topics)
- Four standard module elements, always in same order: headline, summary, text, exhibit
- Modules roll up with no hierarchy
- Process: Equalize work across life cycle
- Process: User/topic brainstorming meeting
- Process: Module specifications
- Process: Write modules in any order
- Process: Storyboard walkthrough

I learned this methodology from Paula Berger, SOLUTIONS, Inc. in 1984
I've since realized that much of Weiss came from STOP; he extended the concept from proposal writing to software doc
Elements are placed consistently in modules, always starting on even-numbered pages, creating a relaxed layout that the reader doesn't have to hunt through (Q: does this have any relevance on the web?)
The four module elements supported reader access styles (scanning, skimming, reading, and visual learning)
During the workshop, when Paula said to use only one level of heading, one of our principal-level writers began to physically tremble: "I don't think I can live without my fifth-level headings"

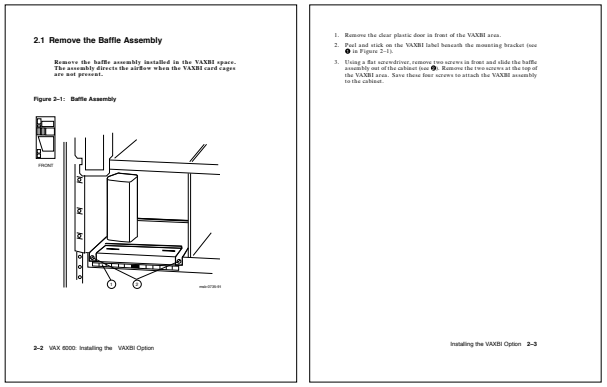
Not just a format but a process for creating it

I used and taught this methodology at several companies
When the time came to convert our structured documents into DITA, conversion went very smoothly

Sample Structured Document

VAX 6000: Installing the VAXBI Option **(1991)**

- 7.5 x 9 inch page size
- 2-page spreads
- L2 head (headline) always starts on left-hand page
- Abstract (summary) in boldface
- Figure (exhibit)
- Text



2.1 Remove the Baffle Assembly

Remove the baffle assembly installed in the VAXBI space. The assembly directs the airflow when the VAXBI card cages are not present.

Figure 2–1: Baffle Assembly

FRONT

mkv-0736-91

2–2 VAX 6000: Installing the VAXBI Option

1. Remove the clear plastic door in front of the VAXBI area.
2. Peel and stick on the VAXBI label beneath the mounting bracket (see ❶ in Figure 2–1).
3. Using a flat screwdriver, remove two screws in front and slide the baffle assembly out of the cabinet (see ❷). Remove the two screws at the top of the VAXBI area. Save these four screws to attach the VAXBI assembly to the cabinet.

Installing the VAXBI Option **2–3**

I was very pleased to find this on the web when I looked for samples
I worked at DEC, and this is from a VAX workstation hardware manual
This doc team took my workshop! From the preface:

> “This manual uses a structured documentation design. There are many topics, organized into small sections for efficient reference. Each topic begins with an abstract. You can quickly gain a comprehensive overview by reading only the abstracts. Next is an illustration or example, which also provides quick reference. Last in the structure is descriptive text.”

There’s room to squeeze everything onto one page, but squeezing is not the idea; consistent placement is

I think structured documentation is the best layout for topics on physical pages
UI designers will tell you to lay out web pages consistently as well, so the principle is still recognized as valid

DITA:

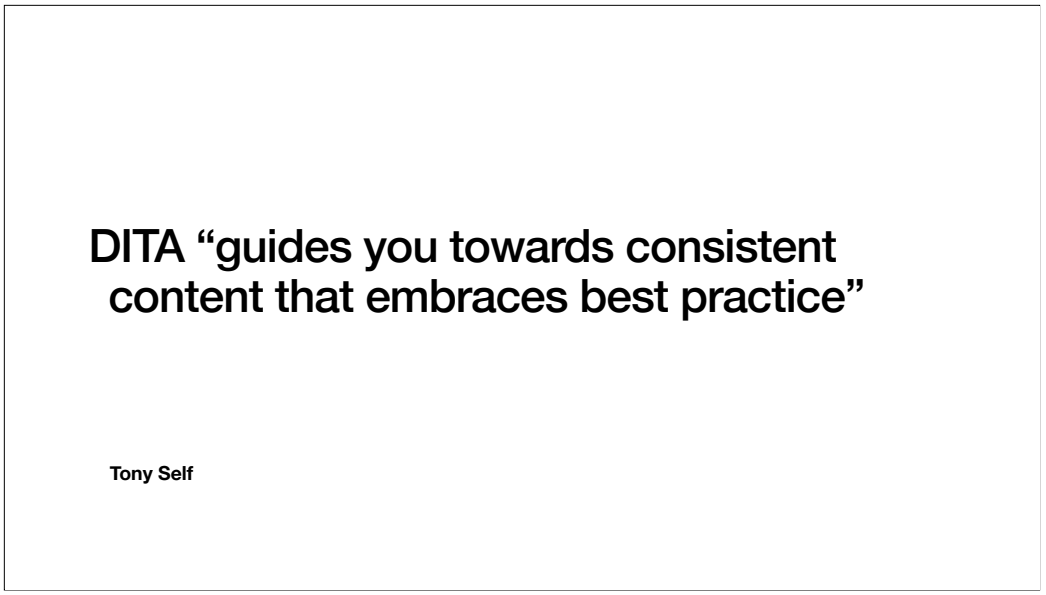
Horn's law made theory

- Darwin Information Typing Architecture (2005)
- Developed by IBM Technical Publications, principally Don Day, Michael Priestley, and Dave Schell
- "A standard designed to help authors create topic-based structured content"
- Typed Horn's information blocks: title, paragraph, list, step, note, context…
- Reduced Horn's map types to three (five) topic types:
  - Concept
  - Task
  - Reference
  - (Glossary)
  - (Troubleshooting)

DITA takes Horn's laws (all information blocks and maps can be classified as one of a few types) and offers a theory (all information must include these types of blocks and topics)
Information design thus goes from descriptive to prescriptive: to document anything you need concept, task, and reference topics, each containing titles, blocks, steps, notes, etc.
DITA thus becomes a checklist guiding your work
(We'll hear from Don Day directly in November, and he can correct any misstatements I've made here)

At a recent BDUG meeting you bemoaned the lack of short descriptions by writers who don't know what short descriptions are for; from information design we know a short description derives from the summary/abstract (which in turn comes from the lede)
Task tags such as <prereq>, <context>, <step>, and <result> descend straight from tasks

What's the value of typing topics?
Theory: All information must include these topic types
Concepts can be reused as a product overview and tasks can be reused as a quick-reference guide
(Q: Is the list of topic types complete?)



(Or, because of validation, it forces you)

For example, developers love to sneak CRUD (create, read, update, delete) procedures into specs, but a task can only have one procedure, so you need to split them into four tasks (plus an overview concept)
If you don’t know how to organize information, even with the guidance of DITA you’ll just thrash around—which often is what we see
If you do know how to organize information and you follow a methodology, DITA is supportive
(For example, it can remind you to consider starting points, prerequisites, ending points, and post-requisites)

Tool Support

Speaking of tools… if you'll allow me a digression down memory lane
Two tool families with two conceptual models of information:

- Create a stream of information (mostly text)
- Create stuff and place it on a page

I'll go roughly in historical order, jumping back and forth between the families

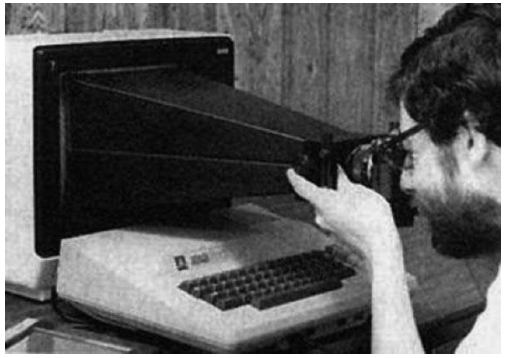
Q: what tools supported the implementation of information structuring before DITA?

My First Tool
(more or less)

http://www.etsy.com/listing/159785179/vintage-smith-corona-coronamatic

For the STOP team, the only tools support was to physically move around pieces of typewritten paper
You saw how little Horn had to work with to format his maps
Even when I started, my first computer was still a typewriter, not quite this powerful
(This one was offered on Etsy as “vintage”)
My cube mate, more senior, wrote out his first drafts longhand in pencil
So structuring consisted of your outline and your experience

Screen shot
(1984)



Courtesy John Hedtke

Q: is this a joke?

RUNOFF
MIT (1964)

```
.ul
Spitballs
.sp
A spitball is piece of paper chewed and shaped into a
lump for use as a projectile.
.sp
To shoot a spitball you need a piece of paper and a
straw. Then:
.sp
.in 4
.un 4
1. Tear off a small strip of paper and chew to moisten
it.
.sp
.un 4
2. Take the moistened paper out of your mouth and form
it into a rough ball.
.sp
.un 4
3. Put the spitball into the straw.
.sp
.un 4
4. Point the straw at your intended target and blow
hard.
.sp
.in
Your target becomes annoyed. If your target realizes
you are the culprit, you may get punched.
.sp
Hide the straw. Getting caught shooting spitballs can
earn you a detention.
```

The first computerized tools to assist writers assumed that documents were a stream of text
One-dimensional, sliced into pages
RUNOFF was written for the Compatible Time-Sharing System (CTSS) in 1964 by Jerome H. Saltzer
“Type out text segments in manuscript form”
Formatting commands embedded in the stream
(I remember “indent/undent” pairs)
Q: how do you format item 10?
Bob Morris and Doug McIlroy re-implemented RUNOFF on Multics, then UNIX
I wrote a manual using COMPOSE, the Multics successor to RUNOFF
COMPOSE paired formatting commands but assumed you knew what you were doing
It was possible to make mistakes such as turning a whole book into a footnote

I think there was no command in RUNOFF to specify a heading (that appeared in COMPOSE)
So structure was format
As you see, nothing whatsoever was semantic

VAX DOCUMENT

Digital Equipment (c. 1985)

```
<HEAD2>(Spitballs)
A <emphasis>(spitball\italic) is piece of paper chewed
and shaped into a lump for use as a projectile.
<p>
To shoot a spitball you need a piece of paper and a
straw. Then:
<list>(numbered)
<le>Tear off a small strip of paper and chew to
moisten it.
<le>Take the moistened paper out of your mouth and
form it into a rough ball.
<le>Put the spitball into the straw.
<le>Point the straw at your intended target and blow
hard.
<endlist>
<p>
Your target becomes annoyed. If your target realizes
you are the culprit, you may get punched.
<p>
Hide the straw. Getting caught shooting spitballs can
earn you a detention.
```

A subset of SGML, as is XML
At that time the output format had lots of thick horizontal lines between blocks
I think the format originated with Information Mapping, but I found the lines oppressive

With DOCUMENT, you became a software engineer—you could write a book that wouldn't compile
(The principal writer on my team congratulated me when I got my first document to compile; I later learned this was a major accomplishment)
DOCUMENT was intensely frustrating; writing groups had as many support staff as writers
No tool support (IDE); later Language-Sensitive Editor (LSE), I think, which was an important productivity advance
You can see some semantic tagging, but documents were still rivers of text
So still no help
DEC at least had very strong book-level templates; you could write an installation manual in your sleep (and I think some of them were)
Q: today, is Markdown akin to DOCUMENT or RUNOFF?

Word: “Alas, the master document method isn’t perfect.”

Dan Gookin, *Word 2019 For Dummies*

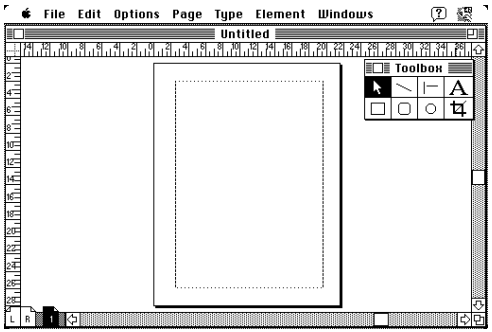
[https://www.dummies.com/software/microsoft-office/word/how-to-use-word-2019s-master-document-feature/, retrieved 16 August 2020]

In 1983 came Microsoft Word
Word is a word processor, originally designed for office memos and student papers, given the Frankenstein treatment (features bolted on) and then force-fed steroids (used for large, complex documents)
Fundamentally designed around creating a single, short, continuous document and controlling its format and presentation
Unenforceable styles, and Style Transmitted Disorder (STD): copy and paste imports every style from every writer who’s ever touched the file
(Neil Perlin's horror story of structuring a doc set in Word where every style was based on Normal paragraphs)
Within a document you can manipulate the outline, but it’s very hard to reuse a third-level item as a second-level item

For modularity, Word offers the master document feature (ptui)
For decades, I’ve been told it doesn’t work, and when I’ve tried it, it didn’t work
Q: After 35 years of Word, have they gotten it to work?

Desktop Publishing

Aldus PageMaker (1985)

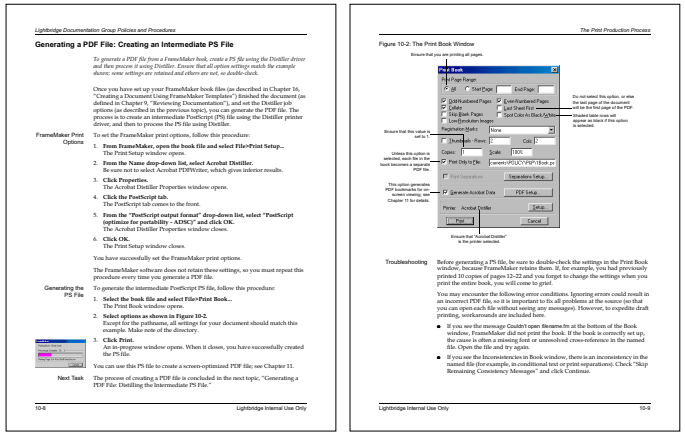


https://viljamis.com/2017/design-tools-processes/

Like Word, the other class of tools is format-based
Apple MacPublisher, 1984; Aldus PageMaker, 1985
Totally page oriented
Killer app: took full advantage of the Macintosh GUI interface, allowed placement of page elements to within a thousandth of an inch, and output PostScript to a laser printer that could render to that degree of accuracy

But no semantic tagging and no reuse; every page of every project could be unique

FrameMaker
(1987)



Lightbridge Documentation Group Policies and Procedures

Generating a PDF File: Creating an Intermediate PS File

To generate a PDF file from a FrameMaker book, create a PS file using the Distiller driver and then process it using Distiller. Ensure that all option settings match the example shown; some settings are retained and others are not, so double-check.

Once you have set up your FrameMaker book files (as described in Chapter 16, "Creating a Document Using FrameMaker Templates") finished the document (as defined in Chapter 9, "Reviewing Documentation"), and set the Distiller job options (as described in the previous topic), you can generate the PDF file. The process is to create an intermediate PostScript (PS) file using the Distiller printer driver, and then to process the PS file using Distiller.

FrameMaker Print Options

To set the FrameMaker print options, follow this procedure:

1. **From FrameMaker, open the book file and select File>Print Setup...**
   The Print Setup window opens.
2. **From the Name drop-down list, select Acrobat Distiller.**
   Be sure not to select Acrobat PDFWriter, which gives inferior results.
3. **Click Properties.**
   The Acrobat Distiller Properties window opens.
4. **Click the PostScript tab.**
   The PostScript tab comes to the front.
5. **From the "PostScript output format" drop-down list, select "PostScript (optimize for portability - ADSC)" and click OK.**
   The Acrobat Distiller Properties window closes.
6. **Click OK.**
   The Print Setup window closes.

You have successfully set the FrameMaker print options.

The FrameMaker software does not retain these settings, so you must repeat this procedure every time you generate a PDF file.

Generating the PS File

To generate the intermediate PostScript PS file, follow this procedure:

1. **Select the book file and select File>Print Book...**
   The Print Book window opens.
2. **Select options as shown in Figure 10-2.**
   Except for the pathname, all settings for your document should match this example. Make note of the directory.
3. **Click Print.**
   An in-progress window opens. When it closes, you have successfully created the PS file.

You can use this PS file to create a screen-optimized PDF file; see Chapter 11.

Next Task

The process of creating a PDF file is concluded in the next topic, "Generating a PDF File: Distilling the Intermediate PS File."

Figure 10-2: The Print Book Window

Ensure that you are printing all pages.

Ensure that this option is set to 1.

Unless this option is selected, each file in the book becomes a separate PDF file.

This option generates PDF bookmarks for an Adobe Framemaker book. See Chapter 11 for details.

Do not select this option, or else the last page of the document will be the first page of the PDF.

Shaded table rows will appear as black if this option is selected.

Ensure that "Acrobat Distiller" is the printer selected.

Troubleshooting

Before generating a PS file, be sure to double-check the settings in the Print Book window, because FrameMaker retains them. If, for example, you had previously printed 10 copies of pages 12–22 and you forget to change the settings when you print the entire book, you will come to grief.

You may encounter the following error conditions. Ignoring errors could result in an incorrect PDF file, so it is important to fix all problems at the source (so that you can open each file without seeing any messages). However, to expedite draft printing, workarounds are included here.

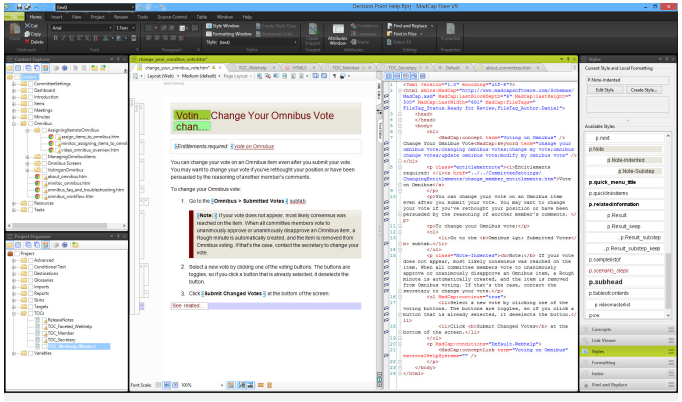
- If you see the message Couldn't open *filename*.fm at the bottom of the Book window, FrameMaker did not print the book. If the book is correctly set up, the cause is often a missing font or unresolved cross-reference in the named file. Open the file and try again.
- If you see the Inconsistencies in Book window, there is an inconsistency in the named file (for example, in conditional text or print separations). Check "Skip Remaining Consistency Messages" and click Continue.

Lightbridge Documentation Group *Policies and Procedures*

Q: Why did Frame conquer Interleaf?
First Interleaf (self-contained environment), then FrameMaker (cheaper, native app), brought desktop publishing to technical publications
Not a river-of-text model; much more effective at placing figures and tables
This sample is from an internal policies and procedures guide that I wrote using FrameMaker in 2001
It was a structured document, and it remains the nicest topic I've ever written
You can create the appearance of structured information
You can label styles semantically, but you can't enforce their use

Structured FrameMaker (post-DITA) does support information structuring

Flare
(1995)



https://jaymanalotoibm.wordpress.com/2013/11/27/breathing-oxygen-xml-in-windows-7/desktop-06-madcap-flare/#main

Q: is online help a river of text?
From its first appearance in 1990, the World Wide Web was organized around pages (topics)
(So this is not, conceptually, a stream of text)
RoboHELP (1992) was designed around creating topics collected in TOCs
RoboHELP, Flare, and similar tools unlocked the efficiency of reusable topics but (for better or worse) retained Word's ability to define styles freely
(At my current job, two writers disagreed on the set of styles, so instead of ~100 styles on the menu we see ~200, and don't know which is which)
It's not clear to me Flare supports information structuring

DITA

Sample task

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<task xml:lang="en" id="Spitball">
  <title>Spitballs</title>
 <shortdesc>A <term>spitball</term> is piece of paper chewed and shaped into
a lump for use as a projectile.</shortdesc>
  <taskbody>
    <prereq>To shoot a spitball you need a piece of paper and a straw.
    </prereq>
    <context>Then:</context>
    <steps>
      <step>
        <cmd>Tear off a small strip of paper and chew to moisten it.</cmd>
      </step>
      <step>
        <cmd>Take the moistened paper out of your mouth and form it into a
rough ball.</cmd>
      </step>
      <step><cmd>Put the spitball into the straw.</cmd></step>
      <step>
        <cmd>Point the straw at your intended target and blow hard.</cmd>
      </step>
    </steps>
    <result><p>Your target becomes annoyed.</p></result>
    <note type="warning">If your target realizes you are the culprit, you may
get punched.</note>
   <postreq><p>Hide the straw. Getting caught shooting spitballs can earn you
a detention.</p></postreq>
  </taskbody>
</task>
```

DITA, of course, uses almost fully semantic tagging

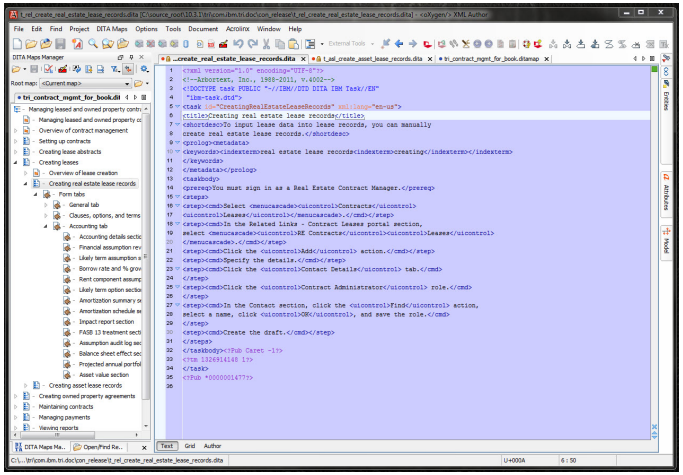
Notice how, both functionally and descriptively, topic types descend from Horn and tags descend from STOP and IBM tasks

And, of course, topics are typed, separate, and fully reusable at any level of a hierarchy

This supports information structuring at the topic level, and pushes you to create topics of the appropriate type

But at the higher levels, information structuring is still mostly in the mind of the writer

oXygen
(2002)



https://jaymanalotoibm.wordpress.com/2013/11/27/breathing-oxygen-xml-in-windows-7/desktop-04-oxygen-text/#main

The current generation of DITA editors is designed around creating maps and reusable topics
oXygen, XMetaL, and other tools enforce XML validation and are schema sensitive, preventing the user (in “author” view, at least) from entering the wrong tag in the wrong place
So finally we have tools that support information structuring and solve the problem of tag overload (limiting choices to only the allowable subset)
But I confess I still spend time struggling with the editor over how to present information, and sometimes I still go into text mode and hack XML tags

Summary

- If you don’t know what you’re doing, tools won’t help you
- If you understand information structuring, you can organize material (in your mind) and then successfully create documents and topics
- Methodologies exist to break down information development into manageable activities that precede the creation of consumable content
- DITA was created with existing methodologies in mind
- DITA-aware editors guide (force) you to use the right tags in the right sequence, and support information structuring at the topic and block level
- No tool replaces the writer’s mind

For more information…

References

- Ament, Kurt. *Single Sourcing: Building Modular Documentation*. Norwich, NY: William Andrew Publishing, 2003.
- Day, Don, Hennum, Erik, Hunt, John, Priestley, Michael, and Schell, David. “An XML Architecture for Technical Documentation: The Darwin Information Typing Architecture.” STC *Annual Conference*, 2003.
- Doyle, Bob. “History of DITA.” dita-archive.xml.org, 13 April 2008, http://dita-archive.xml.org/book/history-of-dita (retrieved 15 August 2020).
- Horn, Robert E. *Mapping Hypertext: Analysis, Linkage, and Display of Knowledge for the Next Generation of On-Line Text and Graphics*. Arlington, TX: Lexington Institute, 1989.
- Self, Tony, hyperwrite.com:
  - “Can you explain that again from the beginning? What is DITA?” *tcWorld*, March 2011
  - “Semantic, Structured Authoring.” http://www.hyperwrite.com/Articles/showarticle.aspx?id=61 (retrieved 16 August 2020)
  - “Writing to STOP.” http://www.hyperwrite.com/Articles/showarticle.aspx?id=85 (retrieved 16 August 2020)
- Tracey, J. R., Rugh, D. E., and Starkey, W. S. *Sequential Thematic Organization of Publications (STOP): How to Achieve Coherence in Proposals and Reports*. Fullerton, CA: Hghes Aircraft Company, 1965.
- Weiss, Edmond, Ph.D. *How to Write Usable User Documentation*, 2nd Ed. Phoenix, AZ: Oryx Press, 1991.
- Wurman, Richard Saul. *Information Architects*. New York: Graphis, Inc., 1997.
- Wikipedia.org:
  - Information Mapping: https://en.wikipedia.org/wiki/Information_mapping
  - Topic-based authoring: https://en.wikipedia.org/wiki/Topic-based_authoring