IXIASOFT

Optimizing Content Reuse with DITA

Keith Schengili-Roberts
DITA Specialist, IXIASOFT

Agenda

- Introduction
- Content Reuse in DITA: The Mechanics
  - A look at the tools built into DITA for creating reusable content
- If You Can't Find it, You Can't Reuse It
  - How to make your content more findable so that it can be reused
- Content Reuse in DITA: How to Make it Work
  - Making reusable content efficient and findable
- Q/A

Who's This Guy?

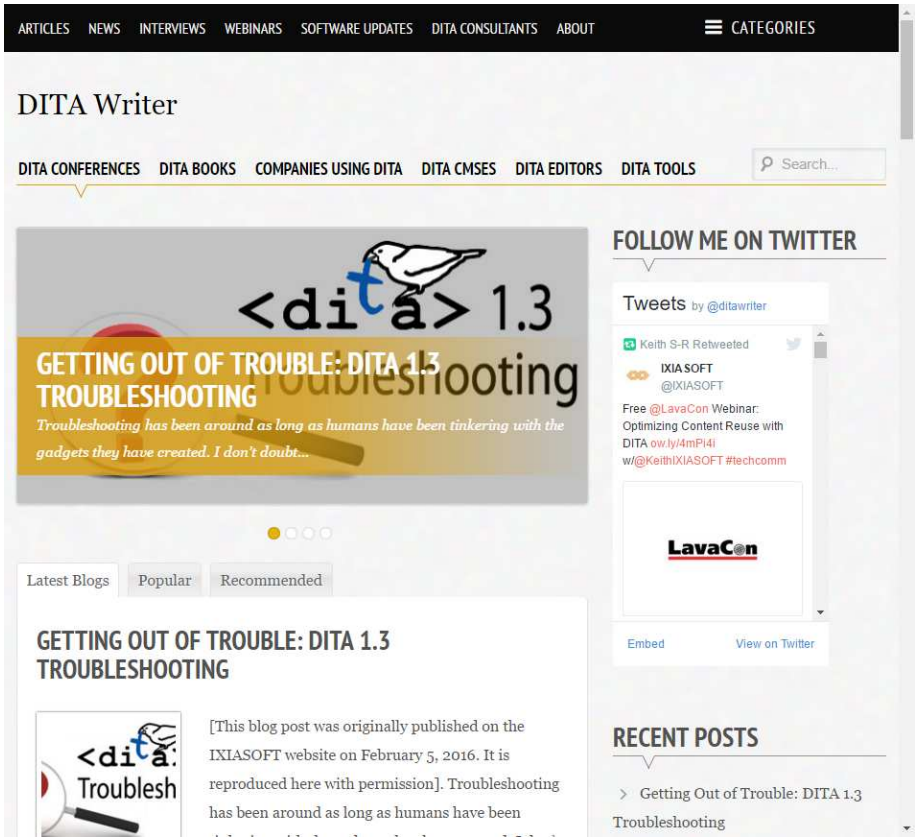
Keith Schengili-Roberts,
IXIASOFT DITA Specialist

What I do:

- DITA evangelist
- Liaison with OASIS; on DITA Adoption and Technical Committees
- Industry researcher
- Lecturer on Information Architecture, University of Toronto
- 10+ Years of DITA XML experience

Also Known As “DITAWriter”

- Industry blog started +5 years ago
- Just over 200,000 hits
- Regularly updated info on:
  - DITA Conferences
  - DITA Books
  - Companies Using DITA
  - DITA CMSes
  - DITA Editors
  - Other DITA Tools
  - DITA Consulting Firms
- News/views on DITA use
- Features interviews with those making a difference in the world of DITA



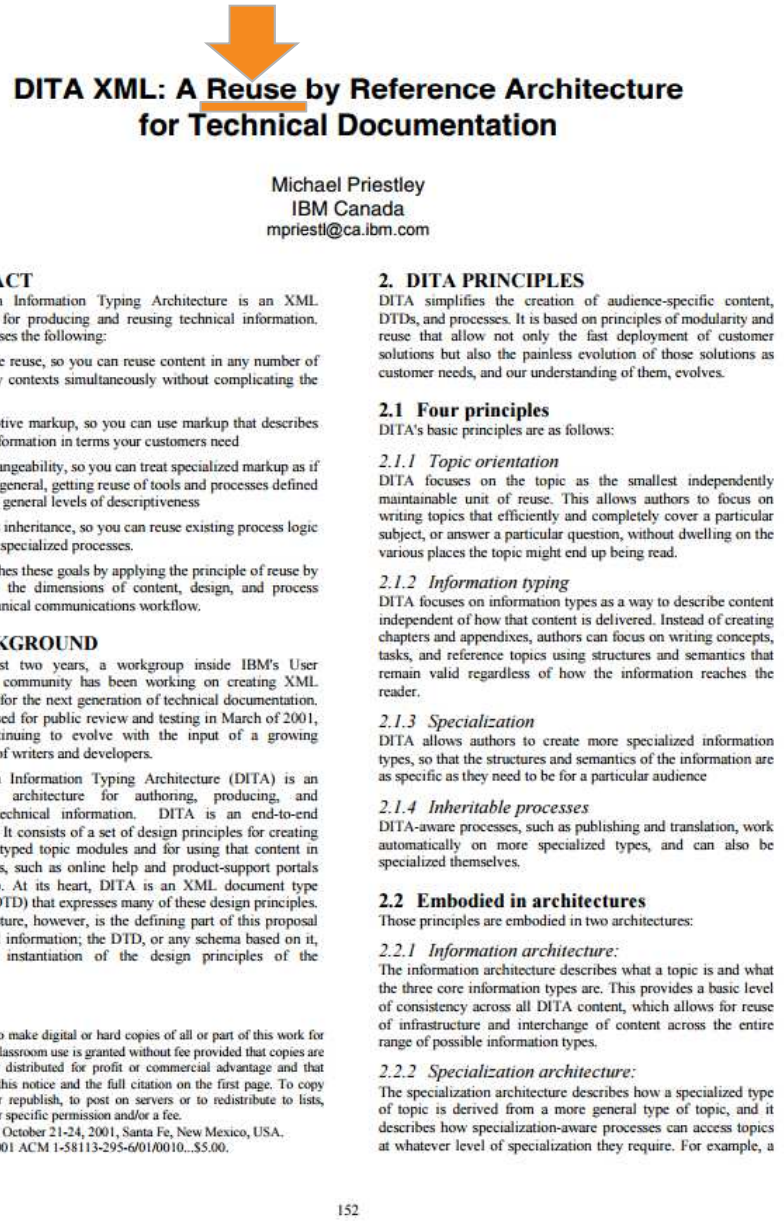
IXIASOFT

A look at the tools in DITA for creating reusable content

CONTENT REUSE IN DITA: THE MECHANICS

Reuse is Built-in to DITA

- DITA was built around the idea of content reuse
  - This has helped make DITA the fastest growing XML-based technical communications standard



DITA XML: A Reuse by Reference Architecture for Technical Documentation

Michael Priestley
IBM Canada
mpriestl@ca.ibm.com

ABSTRACT

The Darwin Information Typing Architecture is an XML architecture for producing and reusing technical information. DITA promises the following:

- Scalable reuse, so you can reuse content in any number of delivery contexts simultaneously without complicating the source
- Descriptive markup, so you can use markup that describes your information in terms your customers need
- Interchangeability, so you can treat specialized markup as if it were general, getting reuse of tools and processes defined at more general levels of descriptiveness
- Process inheritance, so you can reuse existing process logic in your specialized processes.

It accomplishes these goals by applying the principle of reuse by reference to the dimensions of content, design, and process within a technical communications workflow.

1. BACKGROUND

For the past two years, a workgroup inside IBM's User Technology community has been working on creating XML architecture for the next generation of technical documentation. It was released for public review and testing in March of 2001, and is continuing to evolve with the input of a growing community of writers and developers.

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering technical information. DITA is an end-to-end architecture. It consists of a set of design principles for creating information-typed topic modules and for using that content in various ways, such as online help and product-support portals on the Web. At its heart, DITA is an XML document type definition (DTD) that expresses many of these design principles. The architecture, however, is the defining part of this proposal for technical information; the DTD, or any schema based on it, is just an instantiation of the design principles of the architecture.

SIGDOC'01, October 21-24, 2001, Santa Fe, New Mexico, USA.

2. DITA PRINCIPLES

DITA simplifies the creation of audience-specific content, DTDs, and processes. It is based on principles of modularity and reuse that allow not only the fast deployment of customer solutions but also the painless evolution of those solutions as customer needs, and our understanding of them, evolves.

2.1 Four principles

DITA's basic principles are as follows:

2.1.1 Topic orientation

DITA focuses on the topic as the smallest independently maintainable unit of reuse. This allows authors to focus on writing topics that efficiently and completely cover a particular subject, or answer a particular question, without dwelling on the various places the topic might end up being read.

2.1.2 Information typing

DITA focuses on information types as a way to describe content independent of how that content is delivered. Instead of creating chapters and appendixes, authors can focus on writing concepts, tasks, and reference topics using structures and semantics that remain valid regardless of how the information reaches the reader.

2.1.3 Specialization

DITA allows authors to create more specialized information types, so that the structures and semantics of the information are as specific as they need to be for a particular audience

2.1.4 Inheritable processes

DITA-aware processes, such as publishing and translation, work automatically on more specialized types, and can also be specialized themselves.

2.2 Embodied in architectures

Those principles are embodied in two architectures:

2.2.1 Information architecture:

The information architecture describes what a topic is and what the three core information types are. This provides a basic level of consistency across all DITA content, which allows for reuse of infrastructure and interchange of content across the entire range of possible information types.

2.2.2 Specialization architecture:

The specialization architecture describes how a specialized type of topic is derived from a more general type of topic, and it describes how specialization-aware processes can access topics at whatever level of specialization they require. For example, a

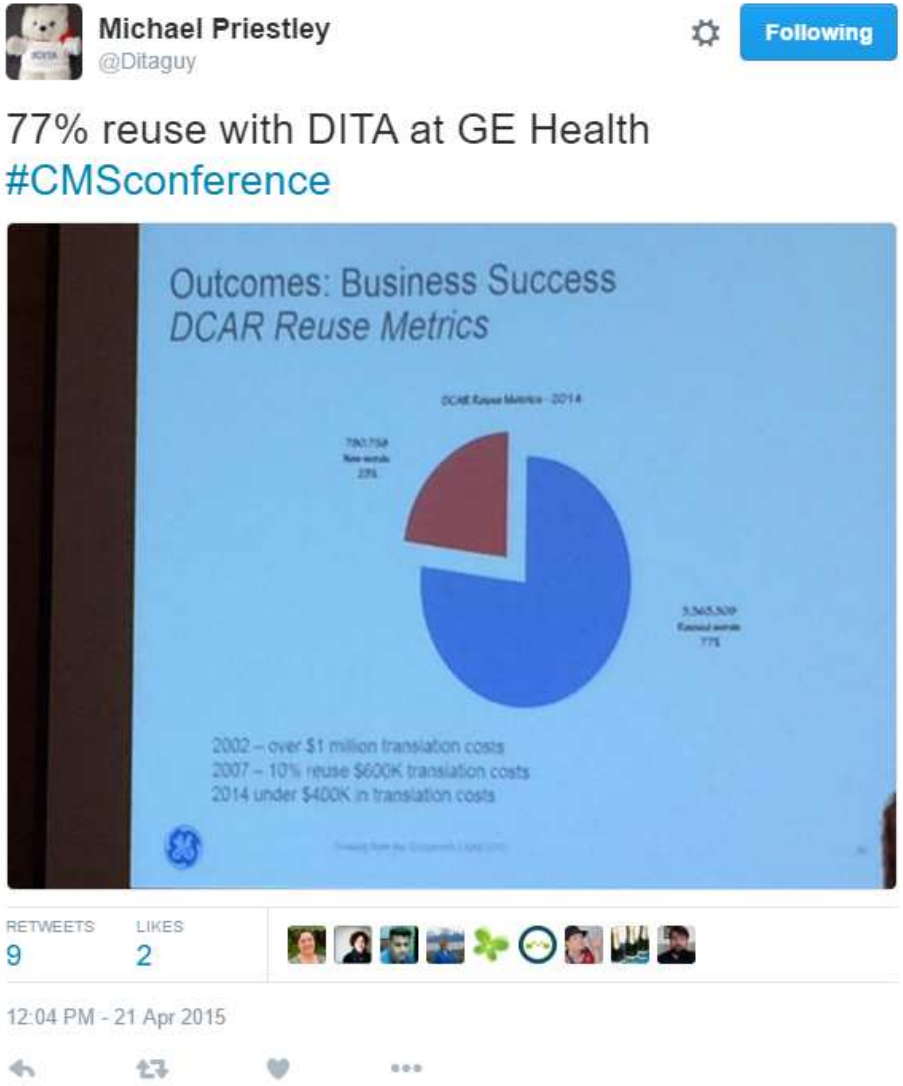
Content Reuse = Consistent Messaging

- Benefit of consistent content and messaging
- Consistent content means consistent user experience
  - Along with being seamless, available and context-specific

Reduced Localization Costs with DITA

- Content reuse in English = localization savings
- If many target languages, ROI argument for move to DITA (+ CCMS) is easier

DITA-related Localization Savings Example



The Levels of DITA Reuse

- Structural (ditamap, map)
- Topic
- Conref (and Conkeyref, and Conref “push”)
- Conditional processing
- Keys
- DITA 1.3: branch filtering and keyscopes

Other:

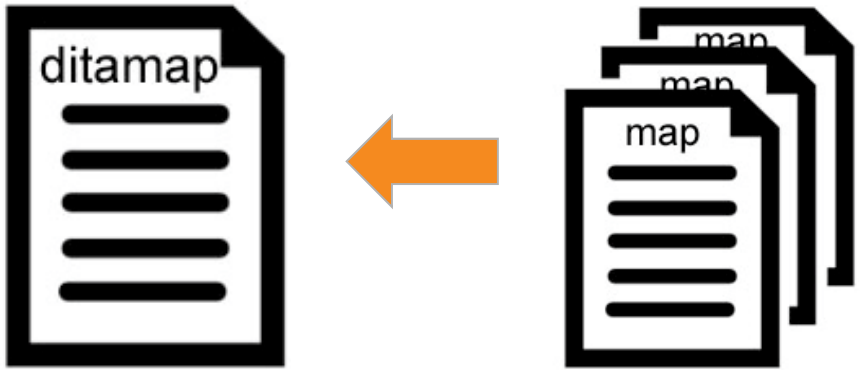
- Special considerations for reusing images
- Metadata to help find content to reuse (more on this later)

Matryoshka (Russian Doll) Reuse

- With each of these levels, it is possible to add the next sub-level of reuse
- So a reused map might (for example) contain a topic from another map, which in turn uses a conref, conditional processing and keys

Structural Reuse

- Topmost level of reuse, involving reuse of entire ditamaps and maps
- ditamaps: goes hand-in-hand with either conditional processing and/or keys
- maps: ditto above, but when used as “chapters” can also be published as separate documents
  - IXIASOFT documents use both scenarios

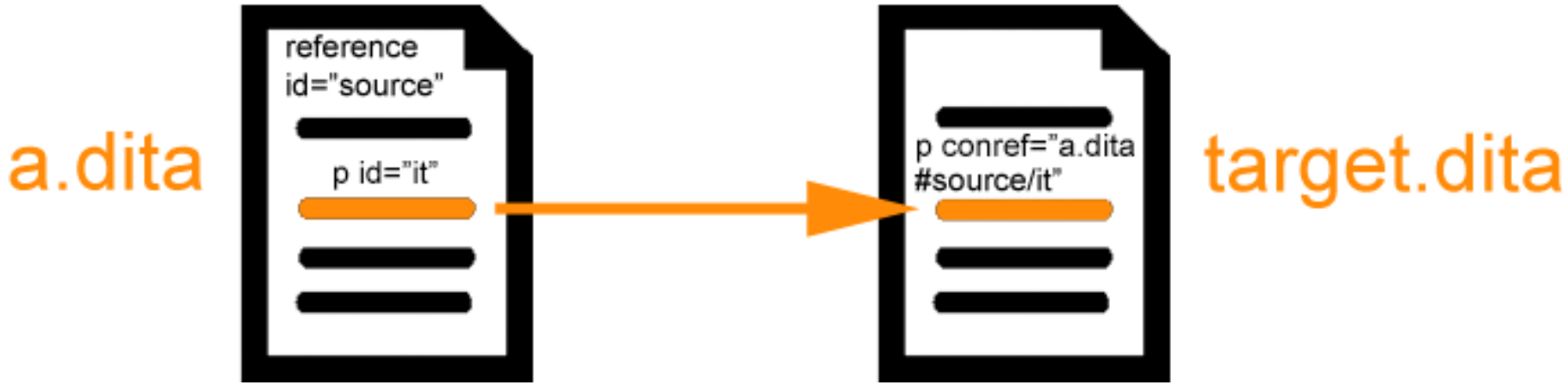


Topic Reuse

- At its easiest, it is simply taking a topic and using it elsewhere
- Can be done at the topicref level, or by using a conref (+ key)
- May also involve use of conditions and keys used within the map

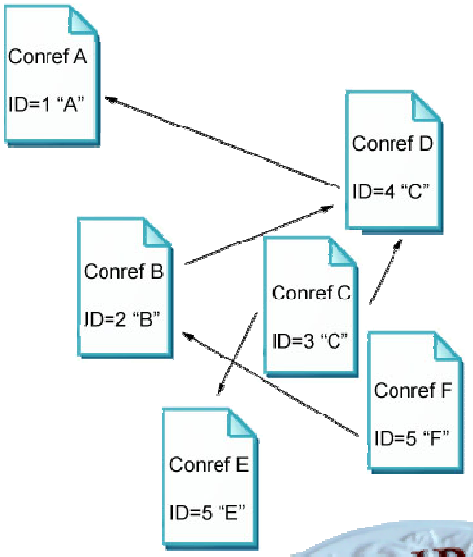
Conrefs

- Involves taking paragraph(s), sentences or phrases
  - Handy when you just want to reuse something smaller than a whole topic
- Requires addition of an ID to the conref being targeted
  - This should be coordinated with others
  - Should use a standard style/convention



Avoid Spaghetti Conrefs!

- Avoid having nested conrefs (i.e. conrefs that contain conrefs), or conrefs that could be subject to change
- IXIASOFT DITA CMS includes a specialization designed specifically for holding conref-able content
  - Can do the same thing by simply agreeing to have conref-able content on designated topics



Conref-ing Down the Rabbit Hole

- There's reuse, and then there's ridiculous levels of reuse: ex. you could conref every single letter in your topics
  - There comes a point where making the conref takes more time than simply writing "new" content
- In general, don't conref below the level of a phrase

Conkeyrefs (Conref “Pull”)

- These are indirect content references defined by keys at the map level
- Good for pulling in short, term values referenced elsewhere
  - It “pulls” in values referenced elsewhere
- Pro: Change terms easily by pointing to different values in the referred topic
- Con: referred topic needs to be managed over time

- Here are some conkeyref targets in a table:

Product Name Variables

Short Description: Reused variable definitions.

Variable names

```
colspec[colname:c1, colnum:1, colwidth:1.0*]
colspec[colname:c2, colnum:2, colwidth:1.0*]
```

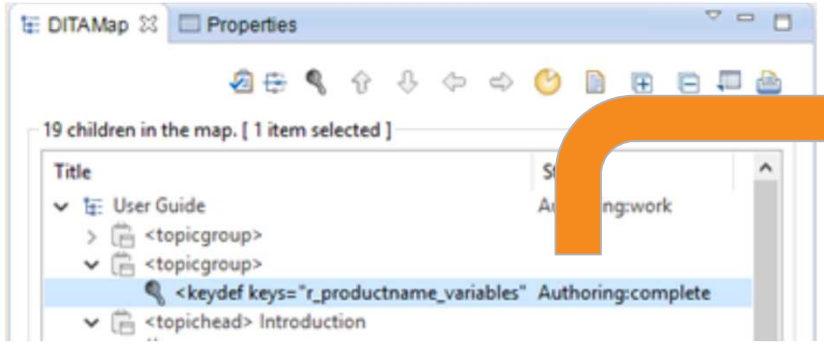
Used as...	Name to use
Primary product name	*STA* **Comment by :** Change value here when primary product name changes.
Company name	*Thunderbird* **Comment by :** Change value here when the company name changes.
Reporting system	*ClusterView*
Controller system	*ClusterControl*
End User	*MobileView*
Data Syncrhonization System	*ClusterBalance*
Analytics Server	*ClusterAnalyzer*
Data Persistence	*ClusterStore*

- Which can then be referenced within topics:

```
<p>Manage the performance of the <term
        conkeyref="productname_variables/ph_prodname"/>
    environment</p>
```

Using conkeyref to Hold a Product Term

Map with keydef topic highlighted




```
<term id="ph_prodname">Vebulon 5000
</term>
```

Reference to the keydef term in another topic:

```
<title>Key <term conkeyref="r_productname_variables/ph_prodname"/>
benefits</title>
```

How this is expressed at output: **Key Vebulon 5000 benefits**

- Reuse emerges when you use change product name to something else (e.g. “Vebulon 6000”)

Best Practice: Hold Your Conrefs / Keys in a Table

- Put your content references (terms, phrases, and images) in a topic designed to hold them, and use a two column table that both describes the reference and what it resolves to
- All product terms in one place, all images in another, etc.

Product Name Variables

Short Description: Reused variable definitions.

Variable names

```
colspec[colname:c1, colnum:1, colwidth:1.0*]
colspec[colname:c2, colnum:2, colwidth:1.0*]
```

Used as...	Name to use
Primary product name	*STA*
	Comment by : Change value here when primary product name changes.
Company name	*Thunderbird*
	Comment by : Change value here when the company name changes.
Reporting system	*ClusterView*
Controller system	*ClusterControl*
End User	*MobileView*
Data Syncrhonization System	*ClusterBalance*
Analytics Server	*ClusterAnalyzer*
Data Persistence	*ClusterStore*

Image Warehouse

Short Description: All images used throughout the *STA* documentation.

STA Icon

```
colspec[colname:c1, colnum:1, colwidth:1.0*]
colspec[colname:c2, colnum:2, colwidth:1.0*]
```

Icon	Description
[Error Icon]	Error Icon
[Warning Icon]	Warning Icon
[Operational Icon]	Operational Icon

STA Marketing Images

Figure: *Thunderbird* *STA* **Computing Management Solution**

Conref Push

- Designed to “push” content at a marked place
- Pro: perfect for a “quick fix” where you need to insert something extra, like an extra step to handle regional variances for example
- Con: arguably trickier/less straightforward than keys; without good communication can lead to unwanted surprises for other writers

- Content from task topic:
  1. Remove the old oil filter.
  2. Drain the old oil.
  3. Install a new oil filter and gasket.
  4. Add new oil to the engine.
  5. Check the air filter and replace or clean it.
  6. Top up the windshield washer fluid.

But for cars sold in North Carolina (for example), we need to include an extra step:

3. Recycle the motor oil. In Durham, North Carolina, you can take it to the Waste Disposal and Recycling Center.

Conref push allows you to add this intermediate step *within* the original task topic

Conditional Processing (Profiling)

- Allows conditions to be applied within topics so that elements can be included or excluded during output processing: DITAVAL is your friend!

- Pro: great for creating content aimed at multiple audiences
- Con: can become unwieldy as more conditions that are added

Sample DITAVAL file:

```
<val>
<prop action="exclude"/>
<prop action="include" att="audience"
val="everybody"/>
<prop action="include" att="audience"
val="novice"/>
<prop action="include" att="product"
val="productA"/>
<prop action="include" att="product"
val="productB"/>
</val>
```

This example excludes all values not specified in the DITAVAL, and includes audience=everybody and novice plus product=product and productB

Recommended Best Practices for Conditional Processing

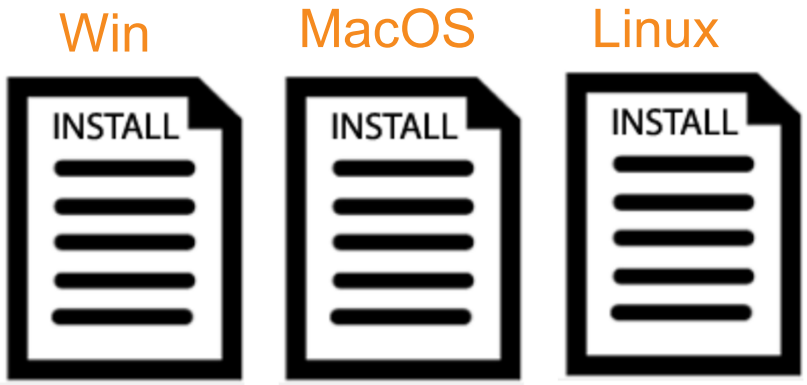
IXIASOFT

- Write your conditions as you create you content
  - And keep track of what you use!
  - A good internal Style Guide will state commonly used conditions and expected values
- Don’t go overboard: if things appear to be getting too complex, they probably are. Consider writing separate topics
- Test your output! There’s nothing worse than revealing content not intended for a particular audience / product

Using ditavalref for Branch Filtering

- In DITA 1.3 conditions can be set within the map and then filtered using a ditavalref (which links to a ditaval file with specific conditions set)
- Allows for greater flexibility when publishing content; if for example you have install instructions for Windows, MacOS and Linux editions of your software, you can use Branch Filtering to render them at one go

- Previously, with just ditaval you could only publish one install set per OS:



- With branch filtering this content can be combined in a single pass

Keys

- An indirect addressing mechanism where key names are attached to resources which are then referenced elsewhere
- Very flexible, highly adaptable

file.dita:

```
<topic id="topicid">
<title>Example referenced topic</title>
<body>
<section id="section-01">Some content.</section>
</body>
</topic>
```

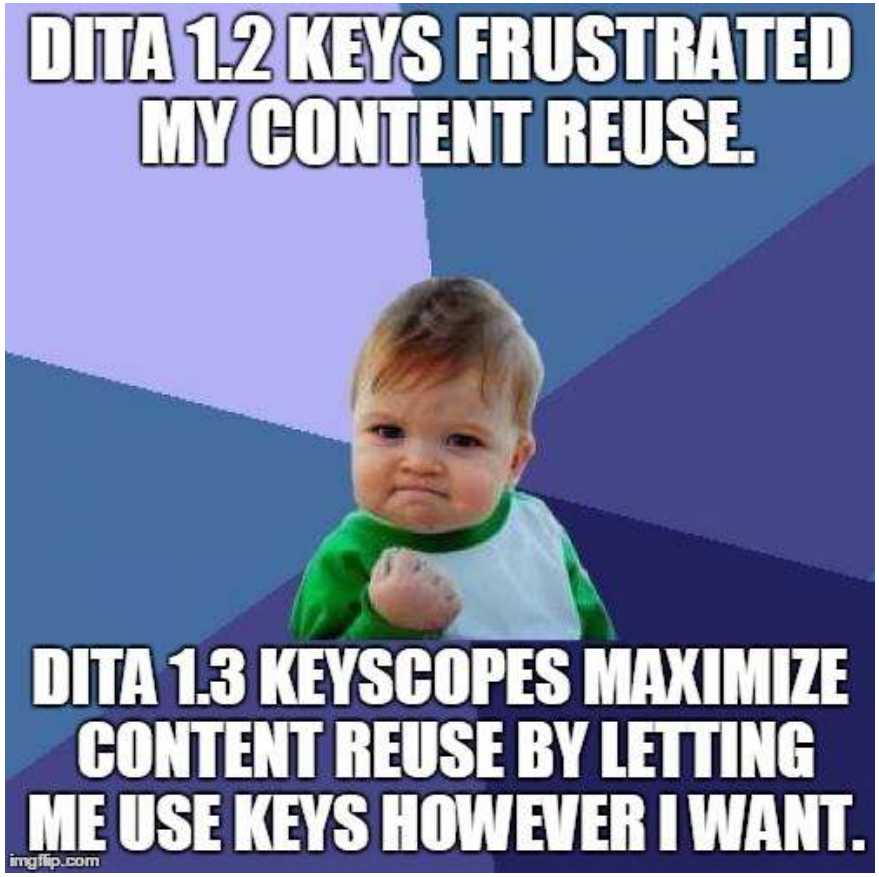
Key definition defined in the map:

```
<map>
<topicref keys="myexample" href="file.dita"/>
</map>
```

Best Practice for Keys

- Eliot Kimber's opinion: "use keys everywhere"
  - I really think he's on to something
- Basically there are two types of keys: resource and navigation only
  - Resource keys link to URLs, images, etc., contained in "warehouse" topics
  - Navigation keys (or "normal keys"), which are named key for topic links
    - Eliot also recommends using keys with conrefs (conkeyref)

DITA 1.3 keyscopes and Reuse

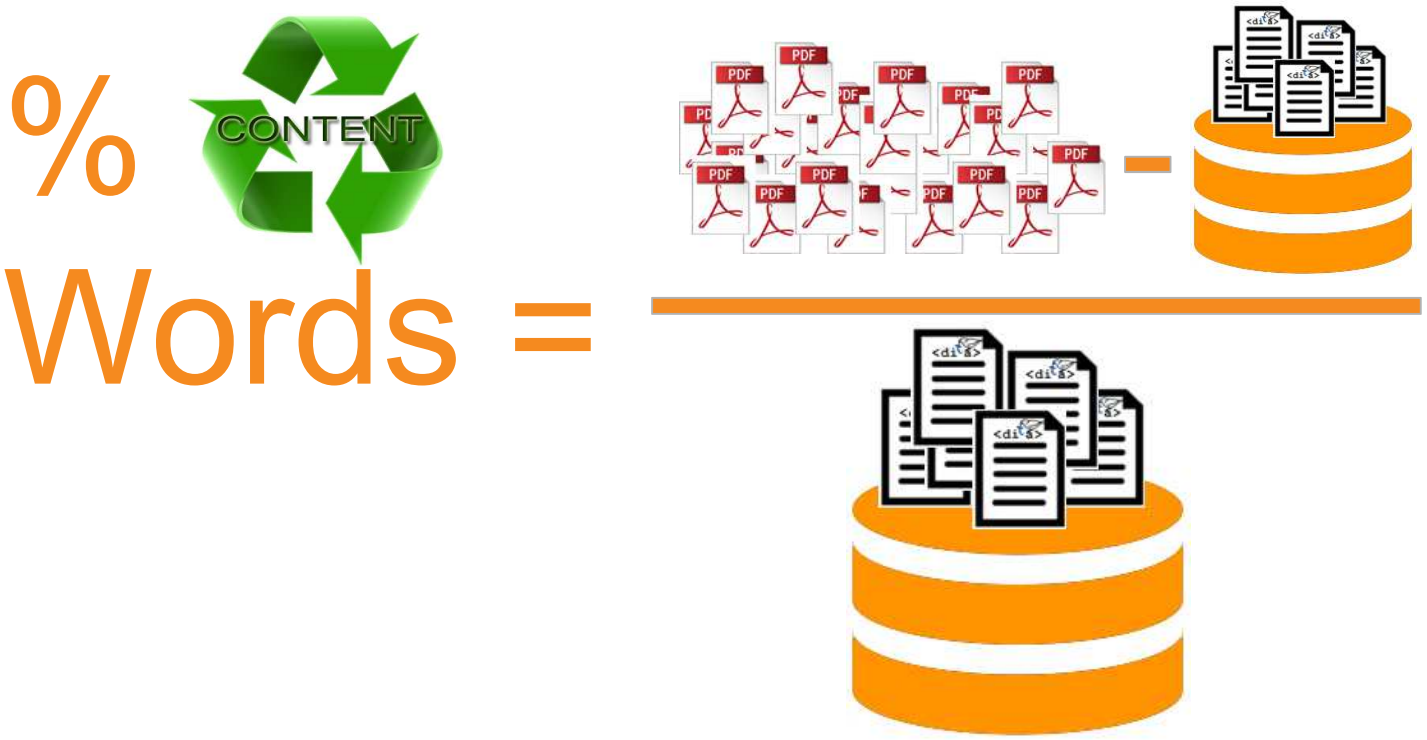


DITA Reuse Metrics

- “Reuse of DITA Topics? What is the Best Metric to Measure the Success of Your Reuse of DITA Topics?” (http://ow.ly/X7mzM) by Bill Hackos

Bill Hackos' Reuse Formula

- Percent Repository Words Reused in Context = (Words in All Produced Content – Words in the Repository) / (Words in the Repository)



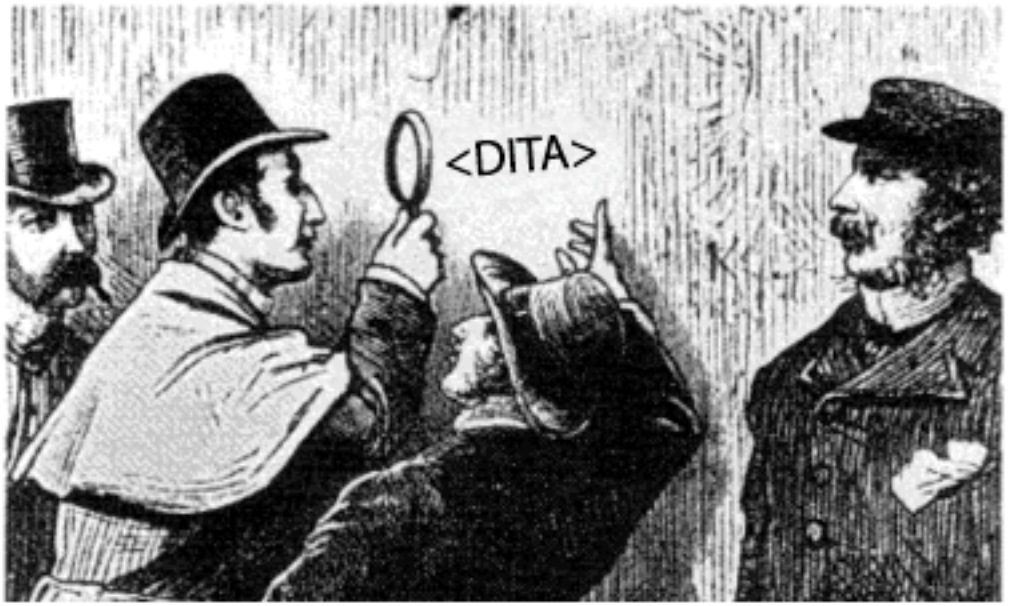
Example Based on IXIASOFT DITA Documents for 2015

IXIASOFT

Based on 2015 numbers:

- Total number of words in the repository: 268,663
- Words in All Produced Content: 623,078
- PRWRC = (623,078* – 268,663)/268,663
- PRWRC = 354,415 / 268,663 = **132%**
- How is 100%+ reuse value possible?
  - Easy: extensive conditional processing
  - Number of publications that are created based on a series of DITAVAL value, as much as 21 per bookmap

IXIASOFT

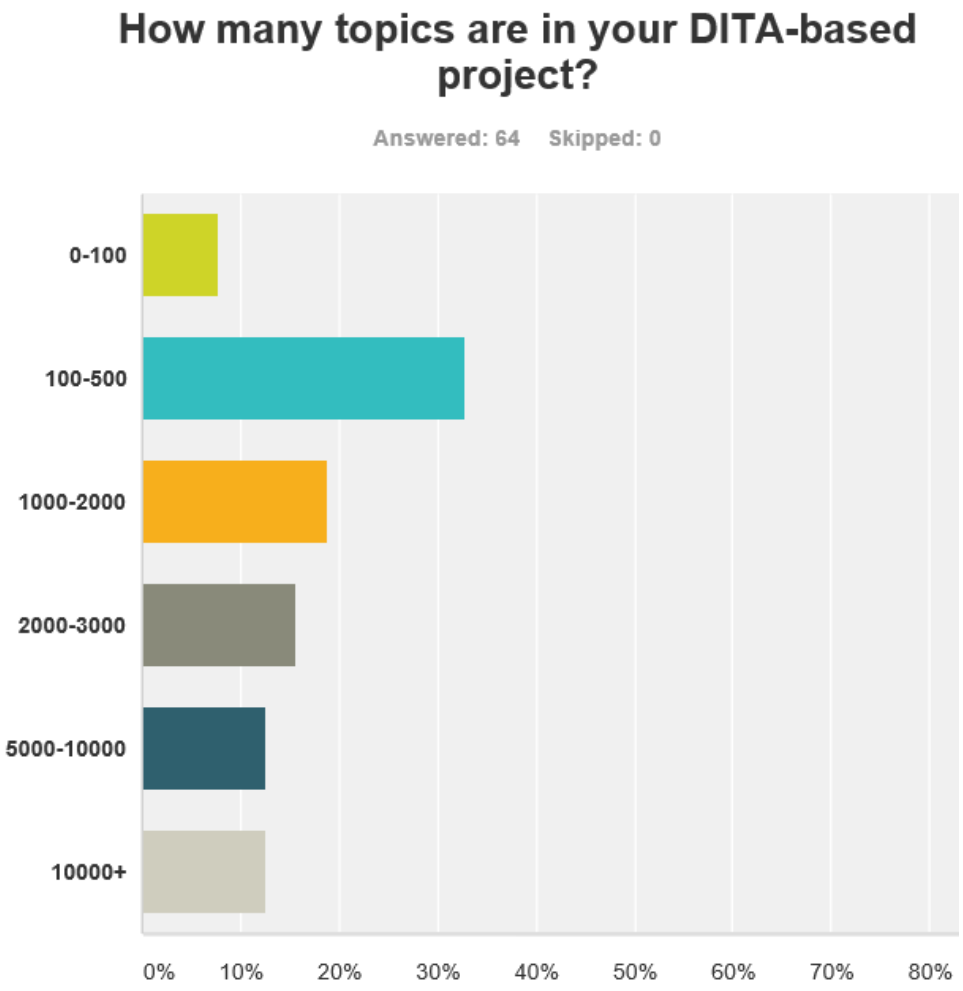


How to make your content more findable so that it can be reused

IF YOU CAN'T FIND IT, YOU CAN'T REUSE IT

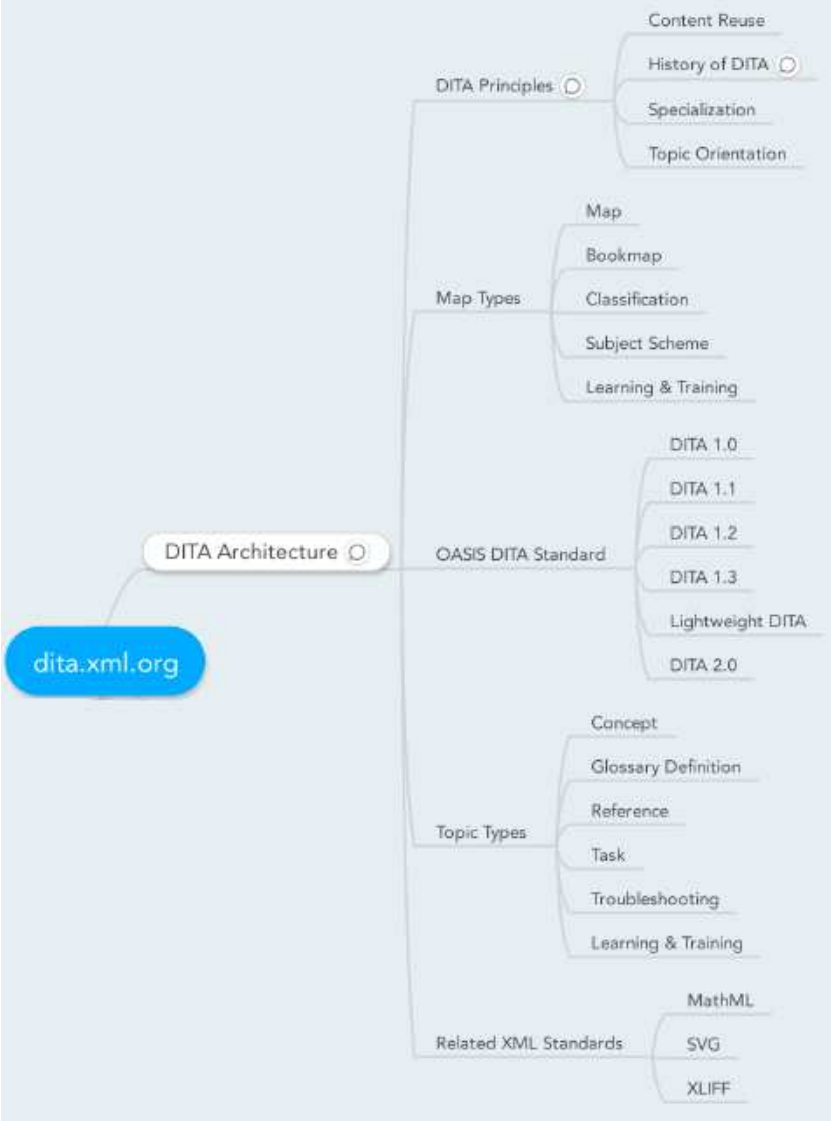
Need for Effective Reuse Grows with Time

- Results of a recent survey where people were asked how many topics are in a given doc project
- While peak is between 100-500 topics, many are considerably larger
- But you need to find content in order to reuse it!



Finding Content for Reuse: Taxonomy

- Check to see if your company or industry already uses a taxonomy
- If you don't already have one, brainstorm a taxonomy with your colleagues
- Example "mind map" was used for revamping dita.xml.org website, using MindMeister.com



Inserting Metadata into Topic and Maps

- Resulting MindMap can be outputted to text, then converted to XML
- Individual elements can then be used within a Subject Scheme map, (or whatever your CMS supports) so you can select elements to insert as metadata values in maps and topics
- When added to maps, metadata is inherited at topic level

In DITA source:

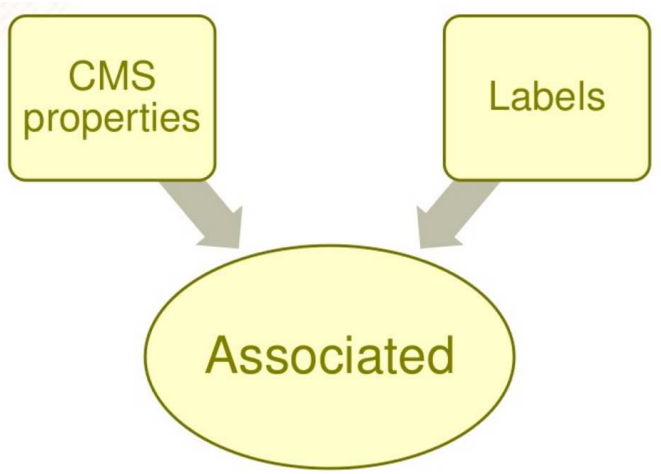
```
<prolog>
...
  <metadata>
    <keywords>
      <indexterm>specifications</indexterm>
      <indexterm>dimensions</indexterm>
    </keywords>
  </metadata>
</prolog>
```

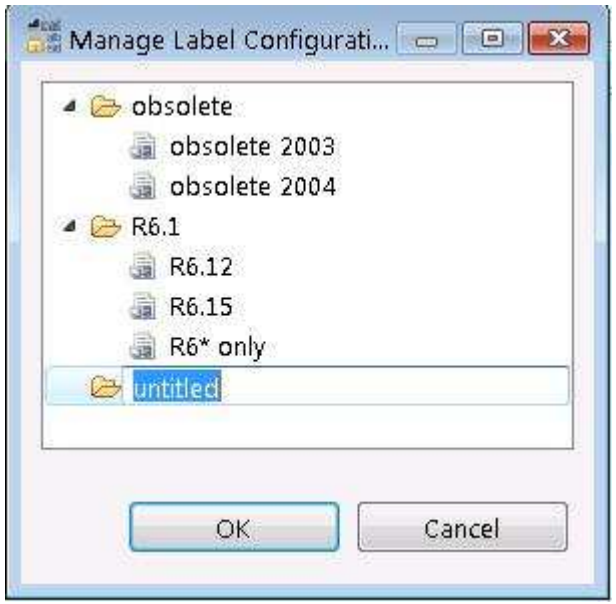
Expressed in XHTML at output:

```
<meta name="DC.subject"
      content="specifications, dimensions"/>
<meta name="keywords"
      content="specifications, dimensions"/>
```

Advantage of CMS Object Approach

- When it comes to finding your own content, some CMSes (such as the IXIASOFT DITA CMS) associate additional metadata with content in the associated property object for each DITA file in the system





Finding Content for Reuse

- Solution: devise effective metadata and tag your topics accordingly!
- Use descriptive titles
- Use short descriptions!

> "Findability precedes usability. In the alphabet and on the Web. You can't use what you can't find."
>
> - *Ambient Findability* by Peter Morville

IXIASOFT

Making reusable content efficient and findable

CONTENT REUSE IN DITA: HOW TO MAKE IT WORK

The Human Angle

- So far we have only talked about the mechanics, but it is people who have to make this work (with or without benefit of a CCMS)
- Past the mechanics, effective reuse within a documentation team comes down to:
  - Writing content for reuse
  - Finding content to reuse
  - Communicating about reusable content

Moving from Legacy Content

- Few who move to DITA have the luxury of starting fresh; legacy content needs to be converted
- Do an audit of this material, look for what is common
- Port only the content that will be used in the future

	Doc #1	Doc #2	Doc #3	Doc #4	Doc #5	
Introduction						
Title	X	X	X	X	X	
Introduction Intro Statement	X	X	X	X	X	
(links to subsections)	X	X	X	X	X	
About the processor	X	X	X	X	X	
(Product) variants					X	
(Product) architecture						
Features	X	X	X	X	X	
System diagram		X				
Software support		X				
Driver architecture overview		X				
Driver APIs		X				
Interfaces	X	X	X	X	X	
YBOB test features		X				
Configurable options	X		X	X	X	
Configurable multiplier			X			
Test Features					X	
Product documentation	X		X	X	X	
Documentation	X		X	X	X	
Design Flow	X		X	X	X	
Architecture and protocol information	X		X	X		
IC architecture	X		X	X		
Advanced Microcontroller Bus Architecture	X			X		
Debug Access Port architecture	X			X		
Embedded Trace Macrocell	X			X		
Floating Point Unit	X					
Product revisions	X	X	X	X	X	
Differences in functionality between (x) and (y)	X			X	X	

Create a Reuse Strategy

Once the content audit is done, think about:

- Naming conventions for filenames, IDs, etc.
- Determine how granular reuse will be: topic, element or phrase
- What criteria to use when content should be forked

Add this material to your DITA Style Guide!

Communicate About Reusable Content

- Create metadata to describe your content to make it findable
- Determine where and what reusable content will be stored
- Coordinate about how and when content can be reused within the team

Prevent this from happening!

Don’t Go Overboard

Evaluate reuse efforts against the returns:

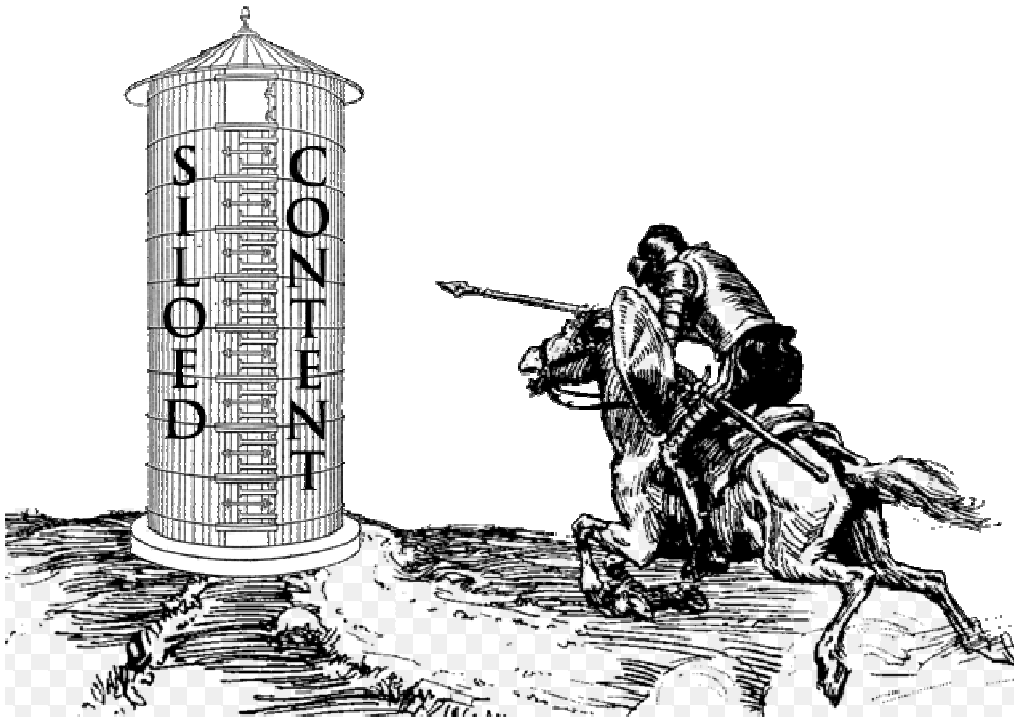
- It is just easier to simply write “Click OK” rather than conref-ing it

“Apples to apples, dogs to ducks”

- When things are very different, don’t try to squeeze reuse out of them

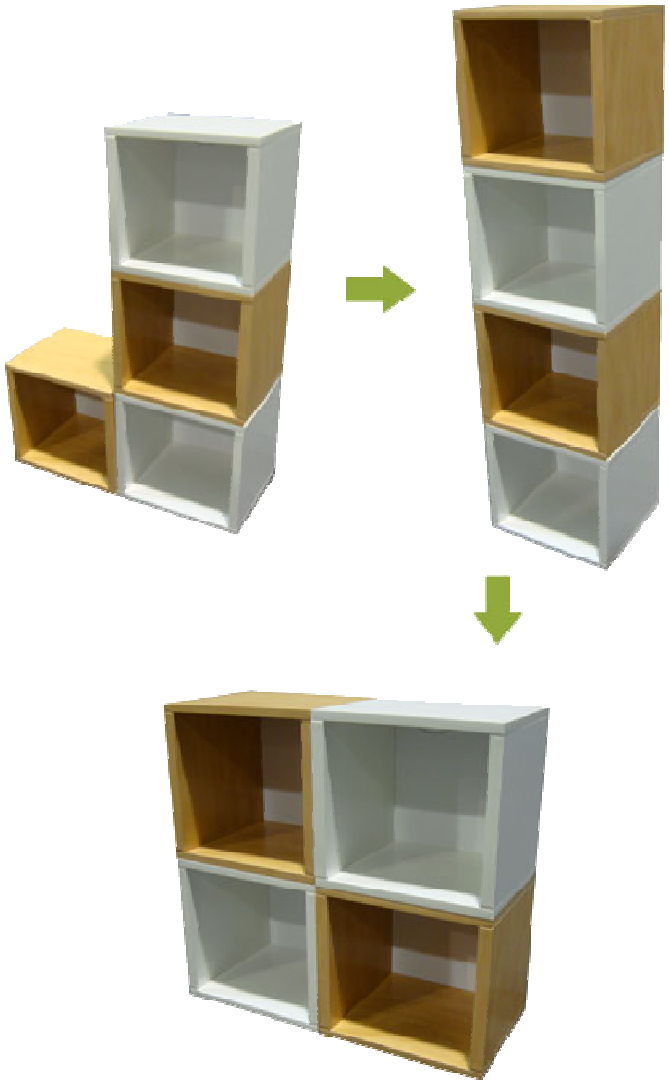
Knock Down Those Silos!

- Effective reuse means that content can no longer be siloed
- By necessity, content "ownership" needs to be shared and communicated
  - One person or group should not "own" their content, **and** change to content that is already shared needs to be coordinated



Writing Content for Reuse

- Keep things concise: content reuse and minimalism go hand-in-hand!
- Omit terms like “see the following” or “the above image…” etc.
- Think about reuse possibilities as you write
  - DITA content is inherently modular, so drop any pretense of narrative



Further Reading

- “DITA 1.2 feature article: conref push”, by Kris Eberlein (dita.xml.org/resource/dita-12-feature-article-conref-push)
- Thunderbird DITA demo files: https://github.com/gnostyx/dita-demo-content-collection
- “The Elusive Promise of Content Reuse” by Leigh White (article: ow.ly/XS0qs, SlideShare: ow.ly/XS0vb)
- “Managing Deliverable-Specific Link Anchors: New Suggested Best Practice for Keys (YouTube: ow.ly/XS0gz)
- Taxonomy Warehouse: www.taxonomywarehouse.com
- “@conref @conkeyref @conrefpush: Reuse Strategies in DITA When Migrating Legacy Content” by Adam Sanyo ow.ly/XS0db
- “Reuse of DITA Topics? What is the Best Metric to Measure the Success of Your Reuse of DITA Topics?” by Bill Hackos ow.ly/X7mzM
- oXygen DITA usage survey: http://ow.ly/4nkLpa (prelim. results: http://ow.ly/4nkLuh)

QA

- Don’t forget to attend my IXIASOFT colleague Nolwenn Kerzreho’s (@NolwennIXIASOFT) sessions at LavaCon Dublin!
  - *Snakes & Ladders: Content Collaboration in the Real World* Sun June 5 @ 8:30am
  - *DITA Metrics for a Continual Improvement Process* Mon June 6 @ 2:40pm
- Blog: www.ixiasoft.com/en/news-and-events/blog
- Twitter: @IXIASOFT (and @KeithIXIASOFT)
- IXIASOFT DITA CMS Users LinkedIn group: www.linkedin.com/groups?gid=3820030
- Member of OASIS DITA Technical Committee

Additional Materials

Conref-ing Down the Rabbit Hole Example

Ridiculous Conref Example

The quick brown fox jumps over the lazy dog.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN" "concept.dtd">
<concept id="concept_gnf_qnf_v5">
 <title>Ridiculous Conref Example</title>
 <shortdesc/>
 <conbody>
  <p><ph conref="t.dita#concept_t/CapT"/><ph conref="h.dita#concept_h/SmlH"/><ph
    conref="e.dita#concept_e/SmlE"/>
   <ph conref="q.dita#concept_q/SmlQ"/><ph conref="u.dita#concept_u/SmlU"/><ph
    conref="i.dita#concept_i/SmlI"/><ph conref="c.dita#concept_c/SmlC"/><ph
    conref="k.dita#concept_k/SmlK"/>
   <ph conref="b.dita#concept_b/SmlB"/><ph conref="r.dita#concept_r/SmlR"/><ph
    conref="o.dita#concept_o/SmlO"/><ph conref="w.dita#concept_w/SmlW"/><ph
    conref="n.dita#concept_n/SmlN"/>
   <ph conref="f.dita#concept_f/SmlF"/><ph conref="o.dita#concept_o/SmlO"/><ph
    conref="x.dita#concept_x/SmlX"/>
   <ph conref="j.dita#concept_j/SmlJ"/><ph conref="u.dita#concept_u/SmlU"/><ph
    conref="m.dita#concept_m/SmlM"/><ph conref="p.dita#concept_p/SmlP"/><ph
    conref="s.dita#concept_s/SmlS"/>
   <ph conref="o.dita#concept_o/SmlO"/><ph conref="v.dita#concept_v/SmlV"/><ph
    conref="e.dita#concept_e/SmlE"/><ph conref="r.dita#concept_r/SmlR"/>
   <ph conref="t.dita#concept_t/SmlT"/><ph conref="h.dita#concept_h/SmlH"/><ph
    conref="e.dita#concept_e/SmlE"/>
   <ph conref="l.dita#concept_l/SmlL"/><ph conref="a.dita#concept_a/SmlA"/><ph
    conref="z.dita#concept_z/SmlZ"/><ph conref="y.dita#concept_y/SmlY"/>
   <ph conref="d.dita#concept_d/SmlD"/><ph conref="o.dita#concept_o/SmlO"/><ph
    conref="g.dita#concept_g/SmlG"/><ph conref="period.dita#concept_period/Period"/>
  </p>
 </conbody>
</concept>
```

- In this code example, the pangram “The quick brown fox jumps overs the lazy dog” uses a conref for each and every letter in the sentence
- Just because you *can* do something doesn’t mean you *should* ;)

A conref/conkeyref Example Using a Table

- Here are some conkeyref targets in a table:

Product Name Variables

Short Description: Reused variable definitions.

Variable names

```
colspec[colname:c1, colnum:1, colwidth:1.0*]
colspec[colname:c2, colnum:2, colwidth:1.0*]
```

Used as...	Name to use
Primary product name	*STA* **Comment by :** Change value here when primary product name changes.
Company name	*Thunderbird* **Comment by :** Change value here when the company name changes.
Reporting system	*ClusterView*
Controller system	*ClusterControl*
End User	*MobileView*
Data Syncrhonization System	*ClusterBalance*
Analytics Server	*ClusterAnalyzer*
Data Persistence	*ClusterStore*

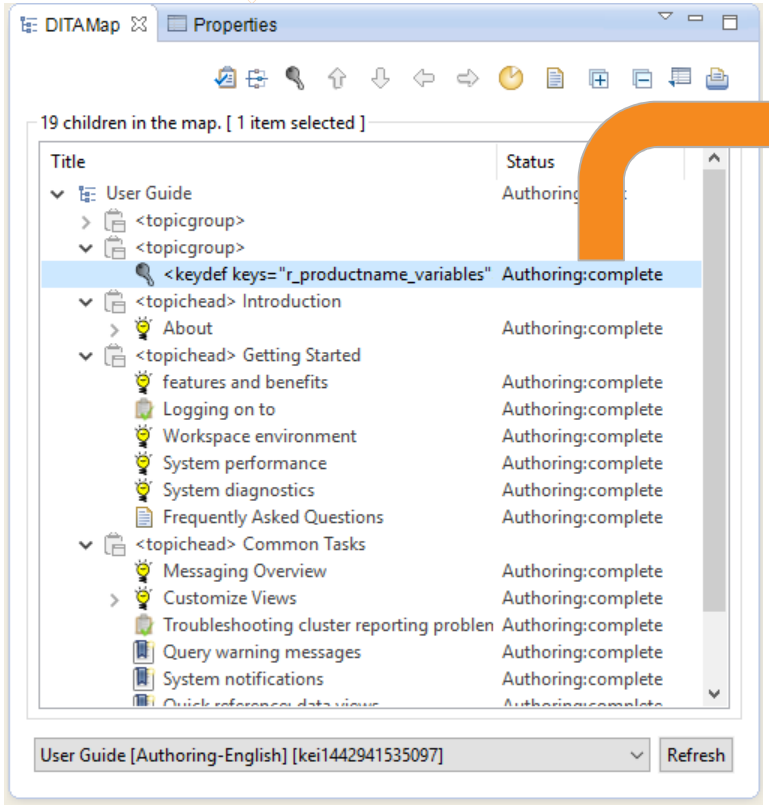
- Which can then be referenced within individual topics:

```
<p>Manage the performance of the <term
        conkeyref="productname_variables/ph_prodname"/>
    environment</p>
```

- The conkeyref points to the filename/id (productname_variables.dita + id="ph_prodname" used in the table

Using conkeyref for Holding a Product Name

Map with keydef topic highlighted



From r_productname_variables.dita

```
<tbody>
    <row>
        <entry>
            <p><ph>Primary product name</ph></p>
        </entry>
        <entry>
            <p><term id="ph_prodname">Vebulon 5000</term></p>
            <draft-comment>Cha  value here when primary
                product name  ges. </draft-comment>
        </entry>
    </row>
    <row>
        <entry>
            <p><ph>Company nar  ph></p>
        </entry>
        <entry>
            <p><term id="ph_co  nyname">Acme Synthotron Inc.</term></p>
            <draft-comment>Cha  value here when the
                company name  ges. </draft-comment>
        </entry>
    </row>
```

keydef filename (minus “.dita”)

term id being referenced

In a separate topic:

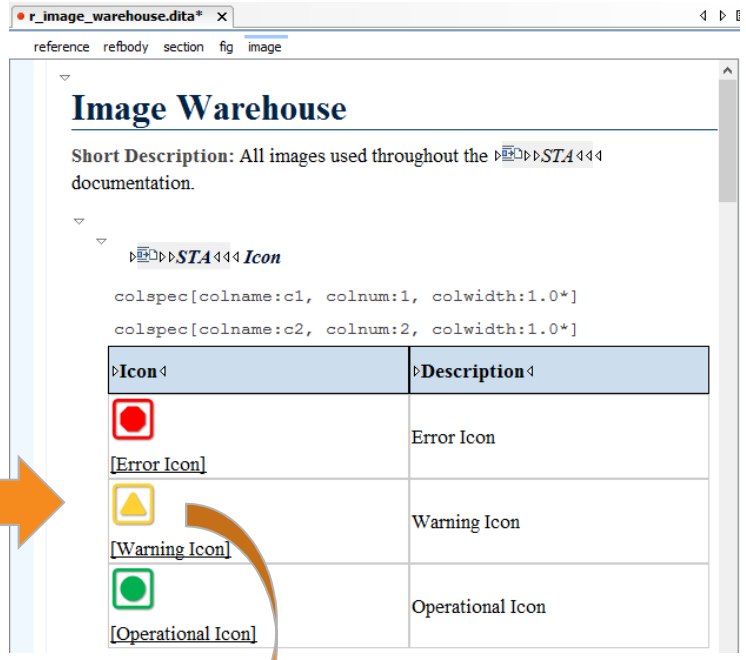
```
<section id="section_p1h_jcz_qr">
 <title>Key <term conkeyref="r_productname_variables/ph_prodname"/> benefits</title>
 <p>
```

= Key Vebulon 5000 benefits

Creating an Image Warehouse Using Keydefs

```
<keydef
  keys="warning_icon"
  href="warning_icon.png"
  format="png">
  <topicmeta>
    <navtitle>warning_icon.png</navtitle>
  </topicmeta>
</keydef>
```

keydef for warning icon image set in a separate file (images-key.ditamap)



Arranging the images in a table (r_image_warehouse.dita)

```
<row>
    <entry><image keyref="warning_icon" id="image_yxw_m3m_mr">
      <alt>Warning Icon</alt>
    </image></entry>
    <entry>Warning Icon</entry>
</row>
```

Sample row from table linking warning icon to an ID

```
<row>
    <entry colname="col2">
        <p><image conkeyref="image_warehouse/image_yxw_m3m_mr" id="image_t3
    </entry>
    <entry colname="col3">
        <p>Warning</p>
    </entry>
    <entry colname="col4">
        <p>Indicates that the cluster is not performing optimally or cannot
            address projected workloads.</p>
    </entry>
</row>
```

Image is referenced/displayed in a topic using conkeyref

Conref Push

- Designed to “push” content at a marked place
- Pro: perfect for a “quick fix” where you need to insert something extra, like an extra step to handle regional variances for example
- Con: arguably trickier/less straightforward than keys; without good communication can lead to unwanted surprises for other writers

Original topic with id added:

```
<steps>
<step><cmd>Remove the old oil filter.</cmd></step>
<step id="drain-oil"><cmd>Drain the old oil.</cmd></step>
<step><cmd>Install a new oil filter and gasket.</cmd></step>
```

Conref “push” topic that points to id in original topic, then specifies content to include:

```
<title>Conref file</title>
<taskbody>
   <steps>
1.    <step conref="changingtheoil.xml#changeoil/drain-oil" conaction="mark">
         <cmd/>
      </step>
2.    <step conaction="pushafter">
         <cmd>Recyle the motor oil. In Durham, North Carolina, you can take it
            Disposal and Recycling Center.</cmd>
      </step>
   </steps>
</taskbody>
```

Resulting output:

3. Recyle the motor oil. In Durham, North Carolina, you can take it to the Waste Disposal and Recycling Center.

- Example from Kris Eberlein’s article on conref push at: http://dita.xml.org/resource/dita-12-feature-article-conref-push

Using ditavalref for Branch Filtering

- In DITA 1.3 conditions can be set within the map and then filtered using a ditavalref (which links to a ditaval file)
- Example below shows install section where ditavalref processes each OS branch (win/mac/linux)

```
<map>
  <topicref href="intro.dita"/>
  <!-- Begining of installing branch -->
  <topicref href="install.dita">
    <ditavalref href="win.ditaval"/>
    <ditavalref href="mac.ditaval">
      <ditavalmeta>
        <dvrResourceSuffix>-apple</dvrResourceSuffix>
      </ditavalmeta>
    </ditavalref>
    <ditavalref href="linux.ditaval">
      <ditavalmeta>
        <dvrResourceSuffix>-linux</dvrResourceSuffix>
      </ditavalmeta>
    </ditavalref>
    <topicref href="do-stuff.dita">
      <topicref href="mac-specific-stuff.dita" platform="mac"/>
    </topicref>
    <!-- End of installing branch -->
    <topicref href="cleanup.dita"/>
  </topicref>
</map>
```

- In this case, the contents of the installing branch appear three times, once for each referenced ditaval (where each platform name is set to include exclusively), so one set of installation instructions per platform (win, mac, and linux)
- For mac content processed as XHTML, install.dita and do-stuff.dita will be processed with mac values included, and filename will be have “-apple” appended to them; because mac-specific.dita is processed in the mac pass, its contents are included here, but excluded in the win and linux processing passes