Relationships in Assemblies

Table of Contents

This article describes various types of relationships that exist among topics in documents. For simplicity, it will be assumed that all the topics in question are represented by topic elements imported into an assembly as resources. The resource definitions are *not* included. Links are a method of expressing relationships in documentation; where the term "links" is used, it indicates that an appropriate mechanism is used in each media (title and page reference for paper, hyper links in electronic media, etc.) unless other description is provided.

Simple Relationship

The simplest relationship will be assumed to one that all the topics share in common. Each is equally related to the others and all would include links to each of the others.

Example 1. Simple Relationship

```
<relationship>
  <association>Related Tasks</association>
  <instance linkend="simpletask1"/>
  <instance linkend="simpletask2"/>
  <instance linkend="simpletask3"/>
</relationship>
```

In this example, three tasks are related to one another and each would have links, under a tile “Related Tasks” to the other two tasks.

Directional Relationships

In more complex situations, it may be necessary to add information about the direction of links. Consider a reference page that is appropriate to multiple topics in a document. In this case, the idea of target and source designations becomes useful. This example adopts the DITA `linking` attribute to express the type of linking that is to be applied to the topic.

Example 2. Directional Links

```
<relationship>
  <association>Reference</association>
  <instance linkend="simpleconcept1" linking="sourceonly"/>
  <instance linkend="simpleconcept2" linking="sourceonly"/>
  <instance linkend="simpleconcept3" linking="sourceonly"/>
  <instance linkend="simplereference1" linking="targetonly"/>
</relationship>
```

In this example, a reference topic is linked to by each of the concept topics. Since it is marked `targetonly`, it does not have links to the concept nodes. Since they are marked `sourceonly`, they do not have links to each other. The link to the reference would be under a heading “Reference”.

If the concepts were related to one another, another relation would be used to specify another relation to indicate that set of links; while leaving off `linking="sourceonly"` for each of the concept instances would provide the information for the linking, it would put them under the “Reference” title, which would be misleading, so another simple relation with an `association` of “Related Concepts” would be used.

Hierarchical Relationships

Another set of directional relationships among topics is the parent-child relationship in a hierarchical structure. Since there is a `structure` element available to express this set of relationships, it should be used for providing the information about hierarchical structure (even if multiple hierarchical structures are provided, they can all be represented as structures in the `assembly`. Otherwise the `linking` attribute could have additional values, but it would require many `relation` elements and would be tiresome to generate. Let's stick with `structure`.

Path Relationships

The concept of a path has become prominent in the theory and practice of hypermedia. A path is a recommended sequence through a series of topics. There might be a set of introductory paths through a help system for different user populations (one for new users, emphasizing basic functionality of a product, another for experienced users, emphasizing new features in the product. This probably represents the most radical departure in markup, but represents an important set of relationships in hypermedia.

Example 3. Path Relationships

```
<relationships type="pathlist">
  <instance linkend="pathshome" linking="pathshome"/>
  <relationship xml:id="new.user.tour" type="path">
    <association>New User Tour</association>
    <instance linkend="overview"/>
    <instance linkend="simpleconcept1"/>
    <instance linkend="simpleconcept3"/>
    <instance linkend="simpletask3"/>
    <instance linkend="simpletask1"/>
  </relationship>
  <relationship  xml:id="experienced.user.tour" type="path">
    <association>Experienced User Tour</association>
    <instance linkend="simpleconcept3"/>
    <instance linkend="simpletask2"/>
    <instance linkend="simplereference1"/>
  </relationship>
</relationships>
```

In this example, two paths are provided, one for a new user and the other for an experienced user. The first `instance` with the `linking` attribute is used to identify a page that provides a list of paths through the help system. The page would have introductory information about using paths and list the two available paths (this might best be coded in the assembly file directly rather than in a separate file, to allow easy cross references to the `relationship` elements by the `xml:id` attributes.

When the user selects one of the paths, a path navigation control is posted and the topic referenced by the first instance in the relationship is displayed. When the next button in the navigator is pressed, the topic referenced by the second instance is displayed, and so on.

In a printed document, the page listing the path would provide references to the first page in each path, and a labeled reference to the next page referenced in the path would be provided with each topic in the path (not as easy as the navigator model, but hypermedia in print existed before Vannevar Bush wrote [As We May think]-- it goes back at least to ancient India).

List Relationship

Another type of relationship that is important in documentation is that of the associative list, that is a list of things about the same thing. Indexes are based on expressing this type of relationship, but there are higher-level lists in most books (list of titles of various sorts). However there are also lists like what are the new features of the product, what are the critical concepts about security, etc. These are frequently provided in the front of a document and can be laborious to assemble.

Example 4. List Relationship

```
<relationships type="lists">
  <relationship xml:id="security.considerations" type="list">
    <association>Security Considerations</association>
    <instance linkend="simplereference1"/>
    <instance linkend="simpleconcept3"/>
    <instance linkend="simpletask2"/>
    <instance linkend="security.overview" linking="listdestination"/>
  </relationship>
  <relationship xml:id="mothballing" type="list">
    <association>Mothballing Systems</association>
    <instance linkend="simpletask2"/>
    <instance linkend="simpleconcept1"/>
    <instance linkend="mothballing.systems" linking="listdestination"/>
  </relationship>
</relationships>
```

This example provides information for two lists, one of topics related to security considerations and the other of topics related to mothballing systems. The final entry in each has a `linking` attribute with a value of `listdestination`. A list of links to the other topics in the relationship will be put at the end of the topic identified as the `listdestination`.

During initial review of this proposal, it was suggested that the `toc` element be used instead of the list `type` on the `relationship` element. Keeping in mind that the model used for relationships assumes that they are expressed in the assembly relationship so that they can be aware of the content in the currently described document (where a module may not be aware of what is being included in the document it is included in) a possible alternate markup for the preceding example would be:

Example 5. Alternate Expression of List Relationship

```
<relationships type="lists">
  <relationship xml:id="security.considerations" type="list">
    <toc>
      <title>Security Considerations</title>
      <tocentry><xref linkend="simplereference1"/></tocentry>
      <tocentry><xref linkend="simpleconcept3"/></tocentry>
      <tocentry><xref linkend="simpletask2"/></tocentry>
    </toc>
    <instance linkend="mothballing.systems" linking="listdestination"/>
  </relationship>
  <relationship xml:id="mothballing" type="list">
    <toc>
      <title>Mothballing Systems</title>
      <tocentry><xref linkend="simpletask2"/></tocentry>
      <tocentry><xref linkend="simpleconcept1"/></tocentry>
    </toc>
    <instance linkend="mothballing.systems" linking="listdestination"/>
  </relationship>
</relationships>
```

While there are likely other models that could be used for this (and this one could likely be modified) the following issues need to be considered with this alternate solution:

- The increased complexity of the markup.
- The render expectations of the table of contents (or list of titles, which is currently implemented using the `toc` element) is fairly heavy weight. What was intended was a simple list, while a table of contents is typically rendered in a much more intrusive manner than a list.
- The content model of the `toc` element allows significantly more complexity than the `relationship` element. Do we modify the content model when used in this context or allow much more than a simple list to be generated with this relationship element (which the other relationships do not currently allow).