Reaching New Levels of Interoperability and Collaboration with DITA

Jerry **Silver**

Abstract

One of the obstacles to XML adoption has been poor interoperability between document types. DITA solves this problem through information typing and specialization, enabling new forms of collaboration and increased interoperability between applications.

Table of Contents

1. Introduction

One of the obstacles to widespread adoption of XML adoption for technical documentation has been the poor interoperability between document types. Applications customized for one type of document are often not able to easily handle other types. This is particularly true of authoring tools, where a significant amount of configuration is usually necessary to adapt the user interface and editing behaviors to the unique semantics of a particular document type. In addition, the level of customization and training required for each document type makes it difficult to "pass the document" among teams consisting of users with diverse technical skills and task requirements. For example, subject matter experts (SMEs), technical writers, and editors will all have different levels of knowledge of markup and document semantics, and will be required to perform different tasks in the authoring interface. Today's monolithic documents and surfeit of customized DTDs and schemas make this difficult to achieve.

Monolithic document structures can also have a detrimental impact on a team's ability to collaborate on content creation. When content is created and managed in large chunks, like a book or even chapters of a book, it can be difficult to get the necessary input from all team members. A single author is typically responsible for structuring, writing, and styling each deliverable. The monolithic deliverables are sent out for review by editors and subject matter experts, but the review context is often too large and/or the review cycle time frame is too short to get in-depth feedback. Consequently, the quality and accuracy of documentation can suffer.

DITA, or Darwin Information Typing Architecture, is a comprehensive framework for authoring, managing, and publishing topic-oriented information in XML. First developed by IBM, DITA goes beyond previous standardization efforts in helping organizations overcome barriers to XML adoption, expedite content reuse, and reduce information redundancies. DITA is enjoying widespread interest and growing support within the technical documentation community, and the DITA specification is now controlled by OASIS (Organization for the Advancement of Structured Information Standards).

The chief benefit of DITA is usually considered to be greater content reuse by enabling more granular, topic-based organization of content. But that same granular orientation can also improve interoperability and team collaboration through the architecture's support for information typing and specialization. The basic structure in DITA, the topic, can be specialized into new types that can handle diverse processing requirements, while remaining compatible with the style sheets, transforms, and processes that exist for their ancestor types. Documents can therefore flow from one application to another and from one user to another, adapting to both specialized and generalized tasks. This enables new forms of collaboration between organizations and increased interoperability between document-oriented applications.

The inherent modularity of DITA also opens up other opportunities for collaboration within documentation teams. XML in general supports specialization of tasks by separating content from presentation, and DITA's information typing carries this further by enabling specialization of topics, and by providing simpler, standardized methods for large-scale reuse. Team members can thus focus on the tasks for which they are most qualified, rather than attempting to manage all tasks (authoring, styling, structuring, etc.) within a monolithic document. They can also focus on the topics on which they are most knowledgeable, and take advantage of DITA's reuse features to leverage content created by other contributors. In addition, many common tasks can be automated, greatly improving the team's productivity.

This paper provides a conceptual overview of the specialization and reuse aspects of DITA, and how these capabilities can be used to enhance interoperability and collaboration. For implementation details and language syntax, please refer to the OASIS DITA Language Reference and the DITA Architectural Specification. These, and other references and resources, are listed at the end of the paper.

Additional references are listed at the end of the paper.

2. Specialization and Reuse in DITA

To understand how specialization and reuse can help improve interoperability and collaboration, it is helpful to understand how the architecture implements these capabilities.

2.1. Specialization

There are two types of specialization defined in the architecture: *Topic Specialization* and *Domain Specialization*.

2.1.1. Topic Specialization

The basic structure in DITA is the *topic*, defined by a Topic DTD (note that XML Schema definitions of all DITA information types are also available). A topic is a unit of information that can stand on its own, or be combined with other topics to create information products like books, marketing literature, Websites, online help, etc.

Topic Specialization (also called Structural Specialization) refers to the creation of new *information types* that inherit from and extend the base topic type, or one of its child types. The DITA 1.0 specification describes three specializations of topic that can serve as useful starter types for technical documentation: *concept*, *task*, and *reference*. These are described in the architecture as follows:[1]

- Concept: Concept topics answer "What is..." questions. They include a body-level element with a basic topic structure, including sections and examples
- Task: Task topics answer "How do I?" questions, and have a well-defined structure that describes how to complete a procedure to accomplish a specific goal
- Reference: Reference topics describe regular features of a subject or product, such as commands in a programming language

2.1.2. Domain Specialization

A *domain* is a set of elements associated with a particular subject area or authoring requirement. For example, the Programming domain contained in the DITA 1.0 specification defines elements for describing programming and programming languages, like `<apiname>`, `<codeblock>`, and `<parmname>`. Other domains contained in DITA 1.0 include Typographic (for non-semantic text highlighting), Software, User interfaces, and Utilities (for useful structures like imagemaps).

DITA domains allow a common semantic vocabulary to be used across different information types. They also allow new elements to be defined independently of information types. Thus, the elements in the Programming domain can be used in a concept, reference, task, or generic topic, without having to be redundantly defined for each of these types.

Equally as important, domains can be omitted from a topic. For example, the Typographic domain, which contains presentation tags like `<b>` (bold), `<I>` (italic), and `<u>` (underline), can be omitted to prevent authors from using non-semantic markup in their documents. Domains can also be omitted to simplify the authoring interface.

2.1.3. Specialization Example

Consider this example: an automotive manufacturer needs to produce documentation for its products. The firm uses many types of components in their products, including mechanical, electrical, body, computer, etc. Each of these have their own descriptive vocabulary, and the documentation for each is produced by different teams. All the teams need to produce or contribute to various information products (owner manuals, service manuals, diagnostic manuals, etc.) but each team only needs to be concerned with the vocabulary of their respective component; e.g. the mechanical documentation team is only concerned with the elements that describe mechanical components and doesn't need to know about the elements used for electrical or other components.

[1] OASIS Darwin Information Typing Architecture (DITA) Architectural Specification v1.0. http://docs.oasis-open.org/dita/v1.0/archspec/dita_spec_22_info_concepts.html

Without the specialization support in DITA, the firm's information architects would most likely take one of two approaches to designing their document types:

1. Develop a single, all-encompassing DTD that could handle all possible vocabularies and deliverables. The result would be large and complex, difficult to understand, hard to maintain, and challenging to develop applications for.

2. Develop multiple sets of document types for different deliverables or for different vocabularies. The smaller DTDs may be easier to develop and less complex, but there will be considerable redundancy and the overall effort of maintaining multiple document types will still be large. It will also be difficult to share content between teams and to combine information into deliverables that refer to multiple components.

With DITA, the firm can use topic specialization to define the common topic types that are used by all the component teams. In addition to the topic types provided by the DITA specification (topic, concept, reference, task), the firm's information architects determine that two new types are required: *parts list* and *frequently asked question* (FAQ). For the parts list, they start with the *reference* type, which is already structured to handle regular content like lists and catalogues, and simply define the differences that are needed for a parts list. This means adding new elements that are needed to describe automotive parts, and also removing elements that are contained in the parent type, if these elements are not needed for a parts list. Similarly, the architects can specialize *concept* to create the FAQ topic type.

The architects then define a domain for each vocabulary - a mechanical domain, electrical domain, body domain, and so on. These component-specific domains can be used with any of the topic types. A mechanical domain can be used with the parts list and FAQ to create information products that are specific to the mechanical components. The electrical domain can be used with the same information types to create information products for electrical components, and so on. Domains that are not relevant to a particular task or application can be hidden from view.

2.1.4. Specialization Benefits

The benefits of specialization for information architecture are apparent from the above example:

- Faster development: Because only differences from existing information types need to be defined, new information types can be developed faster. Early adopters of DITA report that new specializations can be developed and tested in days, compared to the months it had taken using previous methods[2]

- Easier comprehension by authors: Technical writers only need to understand and work with the elements that correspond to their expertise and assigned authoring tasks

- Reduced maintenance costs: The information types are simpler, fewer in number, and with less redundancy, making the architecture easier and cheaper to maintain

- Greater adaptability: The architecture can adapt to changing information by introducing new specializations. Applications and processes developed for the parent types can still work with the new, specialized types

- Increased compatibility: As DITA evolves, application-specific specializations will still be compatible with the standard

2.2. Reuse

Reuse of content is an important goal of all structured documentation systems. Reuse means that a piece of content can be created once and used in many different contexts. This yields important benefits for author productivity and for content quality and consistency. Reuse can also save substantial amounts of money, most dramatically when content

[2]Don Day, Erik Hennum, John Hunt, Michael Priestley, David Schell, Nancy Harrison. An XML Architecture for Technical Documentation: The Darvin Information Typing Architecture. http://www.writersua.com/articles/DITA/index.html.

needs to be translated. If the same text appears in multiple deliverables, that text may only have to be translated once if an effective reuse strategy is in place.

DITA supports three types of reuse: *Topic Reuse*, *Content Reuse*, and *Filtering Reuse*.

2.2.1. Topic Reuse

The topic orientation of DITA inherently enables documentation teams to maximize reuse. Topics, written as discrete units of information, can be combined in many different ways by using *DITA maps*. Each map is a hierarchical collection of topics and is typically associated with a particular deliverable, such as a user manual, online help, or Website. The same topic can appear in multiple maps, and is thus reused in the different information products produced from those maps.

2.2.2. Content Reuse

DITA also provides a mechanism known as a *content reference (conref)* for referencing commonly used content fragments, such as trademark notices, product descriptions, or glossary definitions. A content reference is implemented by adding a conref attribute to an element. The attribute holds an ID value that points to some other element, whose type must be the same as or equivalent to the referencing element. When the topic containing the conref is processed (e.g. parsed), the referencing element is replaced by the element pointed to by the conref attribute.

Any element can be referenced in a conref, as long as it contains an ID attribute with a valid, unique value. The element can be stored in one topic and used in others; when the original fragment is modified, the update will cascade to all the topics that reference it.

For example, to reuse a product description in multiple topics[3]:

1. Enclose the description in a paragraph (<p>) element (a more semantic element could also be used). This is known as the *referenced element*

2. Add an ID attribute to the referenced element and ensure that the ID is assigned a unique value. Here is an example using a brief description of a printer type:

```
<p id="12345"> Liquid ink-jet printers spray tiny drops of liquid ink onto the surface
of the paper.</p>
```

3. In another topic in which you want the product description to appear, add a conref attribute to a <p> (or compatible) element. This is known as the *referencing element*. The value for the conref attribute will be the ID of the referenced element that contains the product description. The referencing element can contain its own content, but that content will be replaced by the product description in the referenced element when the topic is published, or otherwise processed.

   The referencing element for the example shown above would be:

[3] A DITA-aware authoring tool can make this process simple by:

- Automatically generating unique ids for referenceable content
- Allowing users to automatically create content references by browsing lists of reusable content fragments and dragging-and-dropping those fragments into their topics
- Displaying a read-only view of the referenced element inline in the referencing topic. It should be possible to refresh the view as often as necessary to show up-to-date content
- Allowing authorized authors to easily navigate to a referenced element to modify it, if necessary

```
<p conref="12345"> Liquid ink-jet printers spray tiny drops of liquid ink onto the
surface of the paper. THIS DESCRIPTION WILL BE REFRESHED AT PUBLISHING TIME.</p>
```

2.2.3. Filtering Reuse

DITA supports the use of conditional attributes to allow different versions of a document to be generated from a single source file, minimizing the number of different topics that need to be managed, and allowing reused topics and elements to vary for different contexts. The attributes can identify different audiences, product names, platforms, etc. When an information product is produced, the processing application can use these attributes as filters that control which content is included in the output. (The attributes can also be helpful for other applications where metadata is needed, such as search.)

For example, the task for replacing a printer cartridge requires that the printer's cover be opened. In one printer model, the cover is opened from the front; in another model, it is opened from the top. The task can otherwise be generalized for both printers. Instead of creating two task topics, conditional attributes can be used to generate different text, depending on the printer model. This is shown in the following example:

```
<ph product="printflamingo3200">
  front cover
</ph>
<ph product="printflamingo3300">
  top cover
</ph>
```

When a user manual is generated for the Print Flamingo 3200 printer, the phrase "front cover" will be used in the task. When a manual is produced for the Print Flamingo 3300, the phrase "top cover" will be used. In this way, the same task is "reused" for both information products.

DITA allows multiple, space-separated values to be assigned to an attribute, and conditions can be nested. Complex processing conditions can thus be specified by combining these features.

3. DITA for Interoperability and Content Sharing

Understanding how information typing and specialization work in DITA helps us to see how a DITA-based documentation system can achieve greater interoperability among applications and easier interchange of information between organizations.

Each specialized topic type (also called *specialized type*, or just *specialization*) or domain is based on an existing type/domain (called the *parent*). The parent can be topic, one of the standard information types (concept, reference, or task), one of the standard domains, or another specialization based on any of these. The specialization inherits common structures from the parent and can introduce new elements that are mapped back to elements in the parent. The mapped elements can either replace the corresponding parent element, or coexist with it as a peer in the specialization. A *class* attribute is used to define the mapping between elements. The requirement that all specialized elements be mapped back to a parent ensures that the specialized element is at least as restrictive as its parent, with respect to validation rules.

This inheritance mechanism makes it possible for applications that are designed for a parent type to be used with a specialized type, without modification. Thus, new information types can be introduced quickly, without having to define a new set of processing applications. Where the new type does require some specialized processing that differs from its ancestor types, the new functionality can also be developed faster; since the application can already handle the elements that are common to parent and child, only the differences need development focus. Resulting applications are therefore more modular and easier to maintain.

Most commonly, organizations that adopt DITA will take advantage of its inherent interoperability to create generalized style sheets and transforms that can be applied across disparate information types. A high level of consistency can therefore be achieved across information products with relatively low development and maintenance costs. Similar benefits can be gained from creating generalized workflows, translation processes, indexing routines, and searches.

The advantages open up further when we consider the interchange of information between organizations. With complex, custom DTDs, it is difficult to share content with another party unless the recipient has compatible processing applications. With DITA, specializations can still be processed by applications that know how to handle the base DITA information types. This means that structured content can be more easily shared among many types of organizations with different information needs, like product groups in a single company, regulated organizations and government agencies, global publishers and translation services, and collaborating research institutions. Each organization can specialize their content to meet their individual needs, but still maintain compatibility with the base DITA types. The potential also exists to reuse content from external parties; for example, a manufacturer can directly embed information from parts suppliers and supply chain partners in its product documentation (this is often cited as a major motivator for the development of DITA at IBM). With the advent of standard, industry-specific specializations, organizations will also be able to more easily share semantically richer content while maintaining their unique specializations.

Finally, the XML tool and application vendors are able to offer more out-of-the-box functionality in their products to help their customers implement new systems faster, with less customization and configuration required. Customers have the confidence of knowing that DITA will make it easier for their applications to interact with their content, even as their information needs change.

4. DITA for Collaboration

How does DITA help to improve collaboration around content creation?

First, DITA's topic orientation means that content can be created in more granular units. Authors can productively apply their knowledge and expertise to their assigned topics while using topic and content reuse to leverage content created by others. Meanwhile, the entire set of topics can grow more quickly from the individual contributions, with topic maps allowing the various topics to be flexibly combined into multiple information products. The whole process is an efficient way of leveraging the collective talent of the community of content creators.

The granularity also lends itself to better control of the content development process. It is easier to measure progress in the delivery of small units than in the construction of large, monolithic book-like deliverables. Content development tasks can be more specialized by separating content from formatting and by leveraging the individual skills available within cross-functional teams. Workflows become more parallelized, with editing, translation, and design occurring simultaneously with the creation of new content.

Finally, simpler document structures make it easier to expand content creation beyond the technical documentation team to include more subject matter experts and non-technical content contributors. The granularity of topics and the intuitive structure of information types like tasks and references can enable these users to contribute valid content directly to the documentation effort. And techniques like domain hiding can simplify the user interface in structured authoring tools, making these tools more acceptable to end users.

5. Conclusion

DITA doesn't reinvent XML, but represents an evolution in the maturity of XML technology, especially for the creation and management of information intended for publication as technical documents. Indeed, many of the characteristics of the architecture, like topic orientation, reuse, and mapping, have been established as best practices and implemented in other standards and in custom and commercial systems. In this respect, DITA evokes the accumulated wisdom and experience of the technical communicator community, with a healthy sampling of sound principles borrowed from software engineering. The difference is that DITA provides a standardized reference implementation of those principles

and best practices, and for the first time provides a means of continuously evolving and improving the standard without affecting existing implementations.

It would be remiss to say that DITA is an instant panacea. To fully take advantage of DITA requires conversion of at least some legacy content, both in format (e.g. desktop publishing (DTP) formats to DITA/XML) and structure (book-oriented to topic-oriented). Change management is also needed to help users adopt new skills and to reorganize technical writing departments around topic-oriented publishing. However, the advantages of DITA offer a worthwhile incentive to take on these challenges. In addition, many technology vendors and service providers have stepped up to the plate to provide DITA-aware solutions that offer assistance to organizations that are looking for rapid adoption of a DITA-based system.

The standard is still young, having been approved by OASIS in May 2005, although DITA has been battle-tested at IBM and other companies for several years. At time of writing (September, 2005) there seems to be a preference among new adopters of XML to start with DITA as the basis for their documentation systems. These adopters are seeking the advantages listed in this paper - faster, cheaper implementation of an XML-based system for single-source publishing, increased reuse to reduce production costs and improve productivity, greater interoperability and information exchange, and expanded collaboration among content creators and documentation teams, which helps to accelerate the production of high quality information products.

6. References and Resources

- A good starting point for all DITA information are the **OASIS Cover Pages**. http://xml.coverpages.org/dita.html
- The **Blast Radius XMetaL** site has a collection of articles and FAQs related to authoring and management of DITA content. http://www.xmetal.com
- **OASIS DITA Language Reference v1.0**. http://docs.oasis-open.org/dita/v1.0/langspec/ditaref-type.toc.html
- **OASIS DITA Architectural Specification v1.0**. http://docs.oasis-open.org/dita/v1.0/archspec/ditaspec.toc.html
- Erik Hennum. **Specializing domains in DITA**. http://www-128.ibm.com/developerworks/xml/library/x-dita5/index.html
- Michael Priestley. **Specializing topic types in DITA**. http://www-128.ibm.com/developerworks/xml/library/x-dita2/index.html [http://www-128.ibm.com/developerworks/xml/library/x-dita2/index.html]
- Don Day, Erik Hennum, John Hunt, Michael Priestley, David Schell, Nancy Harrison. **An XML Architecture for Technical Documentation: The Darwin Information Typing Architecture**. http://www.writersua.com/articles/DITA/index.html

Biography

Jerry **Silver**
Director, Product Management
Blast Radius Inc. [http://www.blastradius.com]
Vancouver
British Columbia
Canada

Jerry Silver has over 20 years of IT experience, specializing in content management, collaboration, database and application modeling and design, application architectures, XML, and Web technologies. He has been a featured speaker on these topics at numerous industry conferences and a guest lecturer at several Computer Science faculties. Jerry spent 15 years at Oracle in a variety of technical roles, most recently as Principal Product Manager of Oracle Application Server Portal. He also served as Director of Product Strategy with content management vendor NCompass Labs, now part of Microsoft. Currently, Jerry is Director of Product Management at Blast Radius Inc., where he is responsible for collaboration products.