Introduction to the Darwin Information Typing Architecture

Toward portable technical information

Don Day (dond@us.ibm.com)
IBM Corporation

Michael Priestley (mpriestl@ca.ibm.com)
IBM Corporation

David Schell
IBM Corporation

Skill Level: Introductory

Date: 01 Mar 2001
(Updated 28 Sep 2005)

The Darwin Information Typing Architecture (DITA) is an XML-based, end-to-end architecture for authoring, producing, and delivering technical information. This architecture consists of a set of design principles for creating "information-typed" modules at a topic level and for using that content in delivery modes such as online help and product support portals on the Web. This document is a roadmap for DITA: what it is and how it applies to technical documentation.

The Darwin Information Typing Architecture (DITA) is an XML-based, end-to-end architecture for authoring, producing, and delivering technical information. This architecture consists of a set of design principles for creating "information-typed" modules at a topic level and for using that content in delivery modes such as online help and product support portals on the Web.

At the heart of DITA, representing the generic building block of a topic-oriented information architecture, is an XML document type definition (DTD) called "the topic DTD." The extensible architecture, however, is the defining part of this design for technical information; the topic DTD, or any schema based on it, is just an instantiation of the design principles of the architecture.

This document is a roadmap for the Darwin Information Typing Architecture: what it is and how it applies to technical documentation. It is also a product of the architecture, having been written entirely in XML and produced using the principles described here.

Background

This architecture and DTD were designed by a cross-company workgroup representing user assistance teams from across IBM. After an initial investigation in late 1999, the workgroup developed the architecture collaboratively during 2000 through postings to a database and weekly teleconferences. The architecture has been placed on IBM's developerWorks Web site as an alternative XML-based documentation system, designed to exploit XML as its encoding format. With the delivery of these significant updates, which contain enhancements for consistency and flexibility, we consider the DITA design to be past its prototype stage.

Information interchange, tools management, and extensibility

IBM, with millions of pages of documentation for its products, has its own very complex SGML DTD, IBMIDDoc, which has supported this documentation since the early 1990s. The workgroup had to consider from the outset, "Why not just convert IBMIDDoc, or use an existing XML DTD such as DocBook, or TEI, or XHTML?" The answer requires some reflection about the nature of technical information.

First, both SGML and XML are recognized as meta languages that allow communities of data owners to describe their information assets in ways that reflect how they develop, store, and process that information. Because knowledge representation is so strongly related to corporate cultures and community jargon, most attempts to define a *universal DTD* have ended up either unused or unfinished. The *ideal for information interchange* is to share the semantics and the transformational rules for this information with other data-owning communities.

Second, most companies rely on many delivery systems, or process their information in ways that differ widely from company to company. Therefore any attempt at a *universal tool set* also proves futile. The *ideal for tools management* is to base a processing architecture on standards, to leverage the contributed experience of many others, and to solve common problems in a broad community.

Third, most attempts to formalize a document description vocabulary (DTD, or schema) have been done as information modeling exercises to capture the *current business practices* of data owners. This approach tends to encode *legacy* practices into the resulting DTDs or vocabularies. The *ideal for future extensibility* in DTDs for technical information (or any information that is continually exploited at the leading edge of technology) is to build the fewest possible presumptions about the top-down processing system into the design of the DTD.

In the beginning, the workgroup tried to understand XML's role in this leading edge of information technology. As the work progressed, the team became aware that any DTD design effort would have to account for a plurality of vocabularies, a tools-agnostic processing paradigm, and a legacy-free view of information structures.

Many current DTDs incorporate ways to deal with some of these issues, but the breadth of the issues leads to more than just a DTD. To support many products, brands, companies, styles, and delivery methods, we had to consider the entire authoring-to-delivery process. What resulted was a range of recommendations that required us to represent our design, not just as a DTD, but as an information architecture.

Main features of the DITA architecture

As the "Architecture" part of DITA's name suggests, DITA has unifying features that serve to organize and integrate information:

- *Topic orientation.* The highest standard structure in DITA is the topic. Any higher structure than a topic is usually part of the processing context for a topic, such as a print-organizing structure or the helpset-like navigation for a set of topics. Also, topics have no internal hierarchical nesting; for internal organization, they rely on sections that define or directly support the topic.
- *Reuse.* A principal goal for DITA has been to reduce the practice of copying content from one place to another as a way of reusing content. Reuse within DITA occurs on two levels:
    - *Topic reuse.* Because of the non-nesting structure of topics, a topic can be reused in any topic-like context. Information designers know that when they reuse a topic in a new information model, the architecture will process it consistently in its new context.
    - *Content reuse.* The SGML method of declaring reusable external entities is available for XML users, but this has several practical limitations in XML. DITA instead leans toward a different SGML reuse technique and provides each element with a conref attribute that can point to any other equivalent element in the same or any other topic. This referencing mechanism starts with a base element, thus assuring that a fail-safe structure is always part of the calling topic (the topic that contains the element with the conref attribute). The new content is always functionally equivalent to the element that it replaces.
- *Specialization.* The class mechanism in CSS indicates a common formatting semantic for any element that has a matching class value. In the same way, any DITA element can be extended into a new element whose identifier gets added to the class attribute through its DTD. Therefore, a new element is always associated to its base, or to any element in its specialization sequence.
    - *Topic specialization.* Applied to topic structures, specialization is a natural way to extend the generic topic into new information types (or infotypes), which in turn can be extended into more specific instantiations of information structures. For example, a recipe, a material safety data sheet, and an encyclopedia article are all potential derivations from a common reference topic.
    - *Domain specialization.* Using the same specialization principle, the element vocabulary within a generic topic (or set of infotyped topics) can

be extended by introducing elements that reflect a particular information domain served by those topics. For example, a keyword can be extended as a unit of weight in a recipe, as a part name in a hardware reference, or as a variable in a programming reference. A specialized domain, such as programming phrases, can be introduced by substitution anywhere that the root elements are allowed. This makes the entire vocabulary available throughout all the infotyped topics used within a discipline. Also, a domain can be replaced within existing infotyped topics, in effect hiding the jargon of one discipline from writers dealing with the content of another. Yet both sets of topics can be appropriate for the same user roles of performing tasks or getting reference information.

- *Property-based processing.* The DITA model provides metadata and attributes that can be used to associate or filter the content of DITA topics with applications such as content management systems, search engines, processing filters, and so on.
    - *Extensive metadata to make topics easier to find.* The DITA model for metadata supports the standard categories for the Dublin Core Metadata Initiative. In addition, the DITA metadata enables many different content management approaches to be applied to its content.
    - *Universal properties.* Most elements in the topic DTD contain a set of universal attributes that enable the elements to be used as selectors, filters, content referencing infrastructure, and multi-language support. In addition, some elements, whose attributes can serve a range of specialized roles, have been analyzed to make sure that their enumerated values provide a rich basis for specialization (which usually constrains values and never adds to them).
- *Taking advantage of existing tags and tools.* Rather than being a radical departure from the familiar, DITA builds on well-accepted sets of tags and can be used with standard XML tools.
    - *Leveraging popular language subsets.* The core elements in DITA's topic DTD borrow from HTML and XHTML, using familiar element names like p, ol, ul, and dl within an HTML-like topic structure. In fact, DITA topics can be written, like HTML for rendering directly in a browser. In more ambitious designs, DITA topics can be written, like SGML, to be normalized through processing into a deliverable -- say, XHTML or a well-formed XML format targeted for a particular browser's ability to handle XML. Also, DITA makes use of the popular OASIS (formerly CALS) table model.
    - *Leveraging popular and well-supported tools.* The XML processing model is widely supported by a number of vendors. The class-based extension mechanism in DITA translates well to the design features of the XSLT and CSS stylesheet languages defined by the World Wide Web Consortium and supported in many transformation tools, editors, and browsers. DITA topics can be processed by a spectrum of tools ranging from shareware to custom tailored products, on almost any operating platform.

The topic as the basic architectural unit

The various information architectures for online deliverables all tend to focus on the idea of topics as the main design point for such information. A topic is a unit of information that describes a single task, concept, or reference item. The information category (concept, task, or reference) is its information type (or infotype). A new information type can be introduced by **specialization** from the structures in the base topic DTD. Typed topics are easily managed within content management systems as reusable, stand-alone units of information. For example, selected topics can be gathered, arranged, and processed within a *delivery context* to provide a variety of deliverables. These deliverables might be groups of recently updated topics for review, helpsets for building into a user assistance application, or even chapters or sections in a booklet that are printed from user-selected search results or "shopping lists."

Benefits of the DITA architecture

Through topic granularity and topic type specialization, DITA brings the following benefits of the object-oriented model to information sets:

- *Encapsulation.* The designer of the topic type only needs to address a specific, manageable problem domain. The author only needs to learn the elements that are specific to the topic type. The implementer of the processing for the topic type only needs to process elements that are special.
- *Polymorphism.* Special topic types can be treated as more generic topic types for common processing.
- *Message passing.* The class attribute preserves at all times the derivation hierarchy of an element. At any time, a topic may be generalized back to any earlier form, and if the class attributes are preserved, these topics may be re-specialized. One use of this capability would be to allow two separate disciplines to merge data at an earlier common part of the specialization hierarchy, after which they can be transformed into one, the other, or a brand new domain and set of infotyped topics.

DITA can be considered object-oriented due to:

- Data and processors that are separated from their environment and can be chunked to provide behaviors similar to object-orientation (such as override transforms that modify or redefine earlier behaviors).
- Classification of elements through a sequence of derivations that are progressively more specific, possibly more constrained, and always rigidly tied to a consistent processing or rendering model.
- Inheritance of behaviors, to the extent that new elements either fall through to behaviors for ancestors in their derivation hierarchy, or can be mapped to modified processors that extend previous behaviors.

With discipline and ingenuity, some of the benefits of topic information sets can be provided through a book DTD. In particular, techniques for chunking can generate

topics out of a book DTD. In DITA, the converse approach is possible: A book can be assembled from a set of DITA topics. In both cases, however, the adaptation is secondary to the primary purpose of the DTD -- that is, if you are primarily authoring books, it makes the most sense to use a DTD that is designed for books. If you are primarily authoring topics, it makes sense to use a DTD that is designed for topics and can scale to large, processable collections of topics.

DITA overview

The Darwin Information Typing Architecture defines a set of relationships between the document parts, processors, and communities of users of the information. DITA has the following layers that relate to specific design points expressed in its core DTD, `topic`.

Layers in the Darwin Information Typing Architecture

Delivery contexts		
helpset	aggregate printing	Web site; information portal

Typed topic structures			
topic	concept	task	reference

Specialized vocabularies (domains) across information types			
Typed topic:	concept	task	reference
Included domains:	highlighting software programming user interface		

Common structures	
metadata	OASIS (CALS) table

A typed topic -- whether concept, task, or reference -- is a stand-alone unit of ready-to-be-published information. Above the typed-topic layer are any processing applications that may be driven by a superset DTD; below it are the two types of content models that form the basis of all specialized DTDs within the architecture. Next, we'll look at each of these layers in more detail.

DITA delivery contexts

This domain represents the processing layer for topical information. Topics can be processed singly or within a delivery context that relates multiple topics to a defined deliverable. Delivery contexts also include document management systems, authoring units, packages for translation, and more.

delivery contexts		
helpset	aggregate printing	Web site; information portal

DITA typed topic specializations (infotyped topics)

The typed topics represent the fundamental structuring layer for DITA topic-oriented content. The basis of the architecture is the *topic* structure, from which the *concept*, *task*, and *reference* structures are specialized. Extensibility to other typed topics is possible through further specialization.

typed topic structures			
topic	concept	task	reference

The four information types (topic, concept, task, and reference) represent the primary content categories used in the technical documentation community. Moreover, specialized information types, based on the original four, can be defined as required.

As a notable feature of this architecture, communities can define or extend additional information types that represent their own data. Examples of such content include product support information, programming message descriptions, and GUI definitions. In addition to the ability to type topics and define specific content models therein, DITA also provides the ability to extend tag vocabularies that pertain to a domain. Domain specialization takes the place of what had been called "shared structures" in DITA's original design.

DITA vocabulary specialization (domains)

Generally, when a set of infotyped topics are used within a domain of knowledge, such as computer software or hardware, a common vocabulary is shared across the infotyped topics. However, the same infotyped topic can be used across domains that have different vocabularies and semantics. For example, a hardware reference topic might refer to diagnostic codes while a software reference topic might refer to error message numbers, with neither domain necessarily needing to expose the other domain's unique vocabulary to its own writers.

Using the same technique as specialization for topics, DITA allows the definition of domains of special vocabulary that can be shared among infotyped topics. Domains can even be elided entirely, to produce typed topics that have only the core elements. [1] The vocabulary of a domain can take the form of phrases, special paragraphs, and lists -- basically anything allowed within a section, the smallest organizing part of a topic.

specialized vocabularies (domains) across information types			
Typed topic:	concept	task	reference
Included domains:	highlighting software programming user interface		

The basic domains defined as examples for DITA include:

Domain	Elements

highlighting	b, u, i, tt, sup, sub
software	msgph, msgblock, msgnum, cmdname, varname, filepath, userinput, systemoutput
programming	codeph, codeblock, option, var, parmname, synph, oper, delim, sep, apiname, parml, plentry, pt, pd, syntaxdiagram, synblk, groupseq, groupchoice, groupcomp, fragment, fragref, synnote, synnoteref, repsep, kwd
user interface	uicontrol, wintitle, menucascade, shortcut

By following the rules for specializing a new domain of content, you can extend, replace, or remove these domains. Moreover, content specialization enables you to name and extend *any* content element in the scope of DITA infotyped topics for a more semantically significant role in a new domain.

To enable specialized vocabulary, you declare a parameter entity equivalent for every element used in a DTD (such as a topic or one of its specializations), and then use the parameter entities instead of literal element tokens within the content models of that DTD. Later, after entity substitution, because an element's parameter entity is redefined to include both the original element and the domain elements derived from that element, anywhere the original element is allowed, the other derived domain elements are also allowed. In effect, a domain-agnostic topic can be easily extended for different domains by simply changing the scope of entity set inclusions in a front-end DTD "shell" that formalizes the vocabulary extensions within that typed topic or family of typed topics

DITA common structures

One of the design points of DITA has been to exploit the reuse of common substructures within the world of XML. Accordingly, the topic DTD incorporates the OASIS table model (known originally as the CALS table model). It also has a defined set of metadata that might be shared directly with the metadata models of quite different DTDs or schemas.

common structures	
metadata	OASIS (CALS) table

The metadata structure defines document control information for individual topics, higher level processing DTDs, or even for HTML documents that are associated to the metadata as sidefiles or as database records.

The table structure provides presentational semantics for body-level content. The OASIS/CALS table display model is supported in many popular XML editors.

Elements designed for specialization

DITA provides a rich base for specialization because of the general design of elements used in its archetype-like topic DTD.

For example, a section in the base topic DTD can contain both text and element data. However, a section can be specialized to eliminate PCDATA, yielding an element-only content model similar to the body level of most DTDs. Specialized another way, a section can eliminate most block-like elements and thus be characterized as a description for definitions, field labels, parts, and so forth.

With DITA, an effort has been made to select element names that are popular or that are common with HTML. Some semantic names have been borrowed from industry DTDs that support large SGML libraries, such as IBMIDDoc and DocBook.

The attribute lists within the topic DTD reflect this design philosophy. For example, one of the "universal attributes" (which appear on most elements) is `importance`, which defines values for weightings or appraisals that are often used as properties in specialized elements. This attribute shows up in several elements of the task topic specialization with only two allowed values out of the original set: "optional' and "required." In other domains, the elements are more appropriately ranked as "high" or "low" -- again values that are provided at the topic level.

The values of specialization

A company that has specific information needs can define specialized topic types. For example, a product group might identify three main types of reference topic: messages, utilities, and APIs. By creating a specialized topic type for each type of content, the product architect can ensure that each type of topic has the appropriate content. In addition, the specialized topics make XML-aware search more useful, because users can make fine-grained distinctions. For example, a user could limit a search for *xyz* only in messages or only in APIs. A user could also search for *xyz* across reference topics in general.

Rules govern how to specialize safely: Each new information type must map to an existing one, and must be more restrictive in the content that it allows. With such specialization, new information types can use generic processing streams for translation, print, and Web publishing. Although a product group can override or extend these processes, they get the full range of existing processes by default, without any extra work or maintenance.

A corporation can have a series of DTDs that represent a consistent set of information descriptions, each of which emphasizes the value of specialization for those new information types.

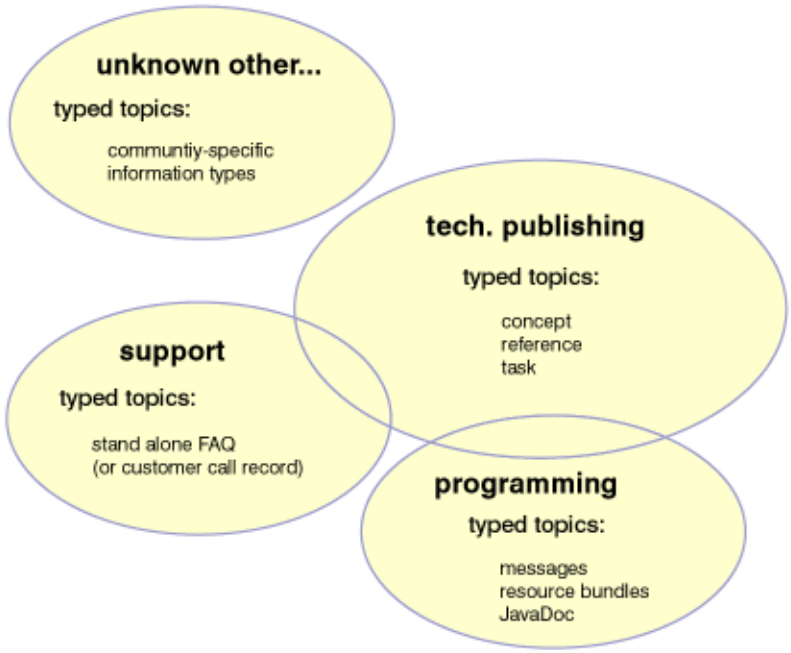
Role of content communities in DITA

The technical documentation community that designed this architecture defined the basic architecture and shared resources. The content owned by specified communities (within or outside of the defining community) can reuse processors, styles, and other features already defined, but those communities are responsible for

defining their unique business processes based on the data that they manage. They can manage data by creating a further specialization from one of the base types.

The following figure represents how communities, as content owners at the topic level, can specialize their content based on the core architecture.

Figure 1. Relationship of specialized communities to the base architecture



In this figure, the overlap represents the common architecture and tools shared between content-owning communities that use this information architecture. New communities that define typed documents according to the architecture can then use the same tools at the outset, and refine their content-specific tools as needed.

End notes

[1]In the original design of DITA, all of the shared vocabulary had been made global to all information types by being defined in the topic DTD, which had two undesireable effects:

- New vocabulary could not be added without increasing the size of the core DTD.
- Certain domain-specific vocabulary could not be prohibited for DTDs specialized for a different domain.

Notices

implied. The use of this information or the implementation of any of these techniques described in this document is the reader's responsibility and depends on the reader's ability to evaluate and integrate them into their operating environment. Readers attempting to adapt these techniques to their own environments do so at their own risk.

Resources

- IBM donated DITA to the OASIS standards organization in March of 2004, where it is now managed by the OASIS DITA Technical Committee (http://www.oasis-open.org/committees/dita/). In April of 2005, OASIS approved Version 1.0 of the DITA specification, which consists of the following documents:
  - OASIS Darwin Information Typing Architecture (DITA) Language Specification: http://xml.coverpages.org/DITAv10-OS-LangSpec20050509.pdf
  - OASIS Darwin Information Typing Architecture (DITA) Architectural Specification: http://xml.coverpages.org/DITAv10-OS-ArchSpec20050509.pdf
  - A consolidated .zip file with all specifications, DTDs, and Schemas is publicly available in the documents section of the OASIS DITA Technical Committee site: http://www.oasis-open.org/committees/download.php/12091/cd2.zip

  A reference implementation toolkit for both the developerWorks and OASIS 1.0 versions of the DITA DTDs/Schemas is available at the DITA Open Toolkit project site on SourceForge: http://dita-ot.sourceforge.net. The DITA Open Toolkit supercedes all previous versions published on developerWorks, the last version of which was commonly called "dita132".
- Find out more about DITA in a companion article, Specializing topic types in DITA, which outlines how to implement DITA specialization (developerWorks, updated September 2005).
- Read Erik Hennum's article Specializing domains in DITA, which shows you how to leverage the extensible DITA DTD to describe new domains of information (developerWorks, updated September 2005).
- Find out how to join the discussion in the DITA forum, moderated by Don Day and Michael Priestley.
- Go directly to the DITA forum.
- Download the latest DITA DTDs, stylesheets, and sample documents.
- Refer to the DITA FAQ set (developerWorks, updated September 2005).
- Get some background on the topic of information architecture at the Argus Center for Information Architecture .

About the authors

Don Day

Besides his main work as husband, father, and cat lover, Don designs and supports publishing tools for IBM's Information Development community and has represented IBM on the W3C XSL and CSS Working Groups. He has B.A.s in English and Journalism and an M.A. in Technical and Professional Communication from New Mexico State University. You can contact Don at dond@us.ibm.com.

Michael Priestley

Michael Priestley is an information developer for the IBM Toronto Software Development Laboratory. He has written numerous papers on subjects such as hypertext navigation, singlesourcing, and interfaces to dynamic documents. He is currently working on XML and XSL for help and documentation management. You can reach Michael at mpriestl@ca.ibm.com.

David Schell

Dave Schell is IBM's chief strategist and tools lead in support of its technical writing (User Technology) community. You can reach Dave at dschell@us.ibm.com.