Status and Direction of XML in Information Development in IBM: DITA

Authors: Dave A. Schell, Ph.D.,
Michael Priestley,
John P. Hunt,
Don R. Day

Abstract

For the past two years a workgroup inside IBM's User Technology community has been working on creating an XML architecture for the next generation information deliverables. In this paper we describe the current state of that work, the status of the Darwin Information Typing Architecture, and our directions for XML. We also discuss our guiding principles for our work on XML and our activities related to validation and proof-of-concept.

Introduction

One of the best indicators of XML's immense popularity as a markup language is the number of proposed XML applications maintained at the XML Cover Pages web site (http://xml.coverpages.org/siteIndex.html#toc-applications). At last count, this list had some 437 entries! What else could possibly be new under the sun? Why not just keep extending the scope of existing DTDs to do more things with fewer new "standards?"

But when you consider that these proposed DTDs represent only the tip of the entire universe of ontology, perhaps the more relevant question is whether there is a better model in the world itself to help us describe its complexity and to deal with it with fewer tools? We believe there is such a model.

In March of 2001, IBM released for public awareness and commentary a new architecture for authoring, producing, and delivering technical information. The **D**arwin **I**nformation **T**yping **A**rchitecture (DITA) deals with the complexity of information at two levels: it goes beyond book oriented DTDs by addressing typed data at the topic level, and it features specialization which allows derivation of new topic types (and their specialized vocabularies) from base types. Topics of any type can be assembled into books, webs, and helpsets without rewriting, owing again to the specialization methodology which allows new vocabularies to be processed reliably by previous tools. DITA was developed by a team lead by Don R. Day, Michael Priestley, and Dave A. Schell of IBM.

The popular hype surrounding XML is that it promises greater reuse, semantic specificity, and interchangeability. Out of the hundreds of XML applications currently proposed or deployed, very few deliver on all three promises because the promises are basically in conflict with each other. For example, in most XML applications if your vocabulary is more specific, then your documents are less interchangeable. DITA helps by having base semantics from which new vocabularies are progressively defined, and this same base semantic continues to support the processing for the new vocabularies.

DITA's progressive features deliver on the promise of XML by including semantic specificity, an interoperable processing architecture, support for multiple relationships within the defined vocabularies, and openness to any number of aggregating or selection mechanisms for processing or search.

Semantic specificity based on derivation has other advantages as well. Derivation avoids the sort of semantic overloading that may occur in other DTDs when an unrelated existing element is used as a base for a new semantic. With no derivation architecture, such newly defined elements may be used in contexts that have nothing to do with the semantics of that new element. Derivation ensures that an entire context for a new vocabulary is properly represented, and that the content models are therefore reasonable. Moreover, one can limit the number of elements in new content models so that fewer non-applicable elements are visible. This is a tremendous authoring aid! In effect, the choices available for writers within a particular specialized vocabulary need only be those that are appropriate for that context, and yet the topic is a proper subset, relationally, to a base structure upon which the processing tools and stylesheets are based.

The following sections not only provide a brief overview of DITA, but also how it was designed and validated.

Overview of DITA

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering technical information. The DITA is an end-to-end architecture for creating and delivering modular technical information. The architecture 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 the heart, the 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.

- **DITA allows for more flexible presentation of information.** Technical writing is moving away from the book and towards smaller "chunks" of information organized around single topics. A focus on topic chunks lets writers build helpsets, webs, or even books for particular audiences, without rewriting. By categorizing the topics by type of information (concept, task, or reference), writers can make sure they're covering the information their customers need. DITA is designed to support these information types.

- **With DITA you can better match your data to your customers' needs.** The power of XML is having "smart" content, that lets a user search on things that matter to them. A one-size-fits-all DTD is undifferentiated and says that the same things matter equally to all users. DITA lets authors refine information types to describe the things that matter to a particular customer set, without breaking the existing categories that matter to a general audience, and without breaking existing processes or tools.

- **With DITA how your information is processed does not change if the amount or type of data being processed does.** By creating a set of core information types for a company, you can ensure that all your company's information follows the same guidelines and gets processed the same way. DITA allows specialization from an existing type to create new types. As long as the new types are specialized from a company's core set, the new types won't break the existing processes, and won't break existing guidelines. Even if you change the processes or guidelines later, the new types pick up the changes automatically.

History

Like many other companies, IBM has long been using and advocating information architectures for its product information, and more recently for its Internet presence. With the arrival of the XML specification in February 1998, the time was right for a reevaluation of our SGML strategy and an assessment of XML's future part in our documentation strategy.

IBM's User Technology organization commissioned a workgroup to do this analysis. Comprised of representatives from Information Development sites throughout IBM and its subsidiary companies, the workgroup was given four charges to work on:

- The first issue for the workgroup was the challenge to preserve the huge investment in legacy documents of both IBM and our customers.

  The workgroup was asked to focus on preservation of legacy material to support our millions of pages of legacy content in SGML and BookMaster documents, as well as the huge volume of content in non-SGML formats, such as word-processing files, help authoring systems, and Notes .nsf databases. Prior to the start of the workgroup, Don Day had demonstrated the conversion of BookMaster documents to well-formed XML and then delivering that content to the Web through transcoding from XML to HTML. The workgroup later applied those ideas to a DTD that represented even earlier IBM documents, this time to a wider variety of browsers. A team at Lotus also successfully demonstrated migrating the entire Notes/Domino help content from a non-SGML, word processor-based publishing system to well-formed and valid XML. In this way, we've demonstrated use of XML as an intermediate representation of legacy data, which allows the use of more current tools for migration, data mining, etc.. The lesson learned is that legacy documents that can be represented as XML and thereby may continue to provide valuable information assets, the use of which is described in the third issue in this list.

- The second issue for the workgroup was to focus the XML activity on the topic-based Information Development information architecture. This architecture moves us from authoring and delivery of large books and manuals to smaller focused articles, or topics, targeted at specific types of content, based on the users' specific tasks and environments. Such topics provide the core information inputs for help systems and other types of online user assistance content, as well as other deliverables and outputs.

  As a parallel effort, the XML work group is focused on providing an infrastructure for our information centers. Information Centers are the framework support for that new generation of information which is based on articles - information that is chunked into smaller, more semantically defined content. For us this is a move to a next generation of tools and tool exploitation including the ability to intelligently use information presentation based on preference and task. It also includes other side benefits such as better search.... all leveraged by using XML

- The third issue on which the workgroup focused was to emphasize the role of stylesheets for transformation and rendering by providing numbers of core styles (Lotus, Tivoli, Transarc) across numbers of widely different devices (workstations to cell phone displays). This also provides us another method to preserve our long term investment in our legacy markup.

  This was one of the earlier proof-points --- we had working prototypes of this type of transcoding to different form-factors with different style sheets from early on in our investigation phase. We were transforming our BookMaster and other legacy data into well-formed XML, applying new style sheets, and rendering it on new form factors such as PDA Web browsers and various levels of HTML/CSS within the first few months of our work effort.

- The fourth issue that the workgroup focused on is an outreach concept --- working with other e-business initiatives in IBM to coordinate our proposed DTD vocabularies with theirs and to promote mutual education at the same time on issues related to XML and documentation.

  Over time, we have reviewed many DTDs and we continue to do so. As a result, we've applied our understanding of the DITA to those DTDs and fed that information back to the other DTD owners for increased coordination and education. In addition, we've learned more about what we need to support in DITA, such as making metadata shareable and developing more author-friendly content models.

The workgroup initially spent some time getting up to speed about XML and W3C technologies and learning about the various XML activities within IBM. Then they set out on a two-phase approach of discovery and application.

Phase 1, completed in January 2000, was to explore the XML technologies in the context of the publishing models that existed at IBM, Lotus, and Tivoli, particularly IBM's ID Workbench based on the SGML DTD, IBMIDDoc. One team converted a simple legacy DTD, the GML Starter Set (or GDOC), into an XML equivalent for use in prototyping activities. Another team developed a tool to port IBMIDDoc source into the XGDOC format. A third team worked on XSL and CSS stylesheets to exercise a number of publishing ideas.

Phase 2 began immediately after that to apply the best practices to a proposed new XML publishing model. Most of the activity occurred within a team that developed a small DTD that incorporated the best practices, from which we derived a set of guiding principles.

Among the guiding principles applied in this phase were:

- rejection of porting existing SGML DTDs already in-place and in use
- repudiation of developing Yet Another Large DTD
- rejection of top-down analysis of current business practice (which we accepted was still solidly instantiated in a book model)
- emphasis on information typing
- emphasis on topic orientation
- emphasis on extensibility and interchange
- reuse by reference
  - content
  - design
  - stylesheets and transforms
- validate and test these principles by prototyping against actual product content

This activity concluded in December 2000 with a design that represented these goals of reuse and scalability. During the next few months, workgroup members tested the materials and documented the activity in a set of articles published on IBM's developerWorks XML Zone, "Introduction to the Darwin Information Typing Architecture" (http://www-106.ibm.com/developerworks/xml/library/x-dita1/index.html), and the Notes/Domino user assistance team at Lotus released "Notes Help 5.0.3 in DITAbase XML" on notes.net (http://notes.net/notesua.nsf/find/notes503xml), as a proof-of-concept of the DITA architecture and DTDs.

Guiding principles of DITA

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.

Principles

DITA's basic principles are as follows:

- 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.

- Information typing:

  DITA focuses on information types as a delivery-independent way to describe content. 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.

- Specialization:

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

- Inheritable processes

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

Embodied in architectures
These principles are embodied in two architectures:

- An 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.

- A 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 generic print process may treat all types as just topics, and a task indexing system may treat all task types as just tasks.

Based on reuse by reference
Both architectures are based around reuse by reference:

The information architecture, by dividing content up into topics and freeing it of delivery-specific elements, allows content to be reused in multiple contexts without being rewritten. This means that many contexts can reuse the same topics, without the reused topics being affected. By contrast, large documents that are conditionally processed to produce multiple outputs quickly run into problems of scalability: with each new output, new conditions must be added to the source, and ultimately the source becomes unmaintainable. Reuse by reference, on the other hand, makes sure that the source stays maintainable, by moving control of the reuse into the reusing context, keeping the source simple.

The specialization architecture, by creating a hierarchy of information types, allows reuse by reference both of design and of process.

- When authors create a specialized type, they can reuse the design of a more general type by reference. This means that the new type can be created faster, and requires less maintenance. And since the new types can be processed just as if it were the more general type, content in the new type can be deployed immediately, giving customers immediate access to the more specific content categories and structures.
- When programmers create a specialized process, they can reuse the code from a more general process by reference. This means that the new process can be created faster, and requires less maintenance. And since the new process reuses the more general process, it can even handle less specialized information, as well as more specialized information.

Reuse by reference lets DITA solutions be deployed quickly, maintained centrally, and improved cheaply.

Gives us the results our customers need
As a result, we can afford to provide customer-specific solutions (delivering the content our customers need, sorted and categorized according to their priorities) without compromising portability or flexibility: meeting the needs of users today, but not at the cost of meeting their needs tomorrow.

Validation

So, how does this work in practice? What happens when we use the DITA architecture and XML DTD against actual content? A small team at Lotus has done just this, examining 7,500 topics in the Lotus Notes and Domino help and how they might fit into DITA information types. For now we've focussed mostly on the help we provide with the Lotus Notes client. Notes client help is task-oriented and directed at end users. The content is currently authored and stored in a word processing system. This Notes help content uses consistent styles and adheres to strict editorial standards. However, it's not been created with a validating SGML or XML editor, which presents some challenges in moving it to DITA topic types and the DITA XML DTDs.

A basic assumption in converting this Notes help content to DITA information types is that all topics were authored as either tasks or concepts. This assumption is based on rules by which the material was originally created by trained technical writers. Based on this set of assumptions, the breakdown of the material follows a pretty simple rule: if a source topic contains a numbered list that's not in a table, it's a task topic; all other topics are concepts. This leaves a few loose ends, such as the question-and-answer troubleshooting topics and the A-Z glossary, but it accommodates the vast bulk of other topics quite nicely.

Starting point: DITA "generic" topics

The actual process of applying DITA markup to this Notes help content takes place in several stages. The first stage involves building XML that validates against the "generic" DITA topic types. These "generic" topics act as a starting point for all topics.

For example, here's the DITAbase markup for a simple "generic" topic:

```
<?xml version="1.0" encoding="UTF-8"?>
<topic id="H_CHOOSE_LOCATION">
  <title>To choose a location</title>
  <prolog>
    <relgroup role="friend" dupes="nodupes">
      <link url="H_CREATE.xml">Creating locations </link>
      <link url="H_SPECIFY.xml">To switch a User ID</link>
    </relgroup>
  </prolog>
<body>
  <ol>
    <li>Choose File - Mobile - Choose Current Location.</li>
    <li>Select a location.</li>
    <li>If necessary, click OK.</li>
  </ol>
</body>
</topic>
```

The structure here is pretty straightforward. A <topic> element encloses the topic, there's a title, a prolog, which contains information about metadata and topic relationships, and a body. For the body content, DITA supports most "standard" HTML tags, which not only makes it easier on authors and but also makes it easier to move existing HTML-tagged content into DITA body sections.

Finally, a topic can also contain other topics, or subtopics. And, you might also notice that a topic does not contain any heading tags. The <topic> tag itself sets the boundaries for headings, and the <title> provides the heading text.

Specialization: Morphing a "generic" topic to a DITA "task" topic

Even though a DITA "generic" topic can hold this content, the above topic is really a task. That is, the ordered list in the body provides a numbered series of steps for completing a specific task, in this case, choosing a mobile location. In this case, a DITA task topic provides a more specialized information structure, with a more specific markup.

Here's what the above "generic" topic looks like when authored as a DITA task topic:

```
<?xml version="1.0" encoding="UTF-8"?>
<task id="H_CHOOSE_LOCATION">
  <title>To choose a location</title>
  <prolog>
    <relgroup role="friend" dupes="nodupes">
      <link url="H_CREATE.xml">Creating locations </link>
      <link url="H_SPECIFY.xml">To switch to a User ID</link>
    </relgroup>
  </prolog>
  <taskbody>
    <steps>
      <step><cmd>Choose File - Mobile....</cmd></step>
      <step><cmd>Select a location.</cmd></step>
      <step><cmd>If necessary, click OK.</cmd></step>
    </steps>
  </taskbody>
</task>
```

As you can see, most of the specialization in the task topic takes place in the body section. Based on our information analysis, task topics present the step-by-step procedure for completing a task. They answer "How do I?" questions about a specific task. Task topics always contain at least one sequence of steps and commands, and may also contain additional supporting information, including an introductory paragraph, notes, tips, pictures, and bulleted lists. DITA provides for this task content in a taskbody, using the following element names: prereq, context, steps, result, taskxmp, and postreq. The steps element, of course, is probably most important -- it provides the numbered procedure for the task. A Steps section contains individual step elements, which include the cmd and optional additional information about that step.

We've completed a DITAbase XML version of all 768 topics in the Notes 5.0.3 end-user help system, and it's available for download in the Domino/Notes Documentation library on notes.net -- http://notes.net/notesua.nsf/find/notes503xml.

General Patterns

Several general patterns emerge from the process of creating this DITAbase version of Notes task help. First, there's lots of "indented" content -- the additional paragraphs, notes, or tables that might appear between steps, which in the DITAbase version appears inside an XML <info> element. Second, all subheadings in the original source start a new sub-topic in the DITA version. Third, in a subtopic, any content preceding the steps becomes a <context> section. And, finally, all content following the steps becomes a <result> section.

And, last but not least, tables. Technical documentation uses lots of tables, and the Notes help topics are no exception. A table, by definition, lays out information according to a certain structure. In Notes client help, there's lots of two-column tables, with headings like "Option/Description", "Click/To", and even "Name/Mood Stamp", which depicts the graphical mood stamps you can add to an e-mail address. In the DITA markup, we decided to represent these kinds of two-column tables as a definition list. This makes the content more accessible through XML processing, doesn't preclude presenting it as a table to the user, but also allows us to extract and present the information in other ways, as well.

Proof-of-concept

The DITA team has run several activities to demonstrate the viability of DITA's design.

The team did an analysis of existing practice within IBM on the structure and use of FAQs--Frequently Asked Questions. The exercise revealed that some presumed information types actually represent the sharing of information and publishing responsibilities between more than one community, as FAQs typically represent the gathering of input from customer service centers, and the republishing of that data through product support sites maintained by Information Development teams. As a result, DITA was provided with a structure for question and answer pairs that can be associated with standalone topics that may be part of a larger collaborative design between the two communities.

A separate workgroup evaluated and documented the essential information model underlying the most current use of messages within IBM, Tivoli, and Lotus. Michael Priestley demonstrated the use of DITA specialization to represent that data model of messages as a specialization of a reference topic.

John Hunt of Lotus demonstrated moving a large volume of Notes and Domino help content to DITA format. A key goal of the exercise was to preserve the huge investment in that content, including the topic text itself, the index keywords, and the ability to produce multiple outputs to HTML, printed books, Notes .nsf help, PDF, and other formats. In a further exercise to identify additional content and combine it with the core Notes help content, John carried out a conversion of an IBM Redbook on the Domino XML Web site into DITA format, and combined this with an XML version of the Domino XML Toolkit reference help. By pulling together disparate information in this way and tying it together with DITAbase XML as the common denominator, he demonstrated the ability of DITA to handle multiple types of information.

These activities have demonstrated the viability of DITA's design, and lead to improvements throughout the process.

The value of analysis

In doing his migrations of various Lotus content, John provided an analysis of the cues in the legacy markup that he used to guide his transformations. For example, he observed that:

- Any topic or subtopic that contains an <ol> tag is probably a task topic.
- If a main topic contains several subtopics that are tasks, then the topic is either a Collection or Super steps topic.
- If no tags other than <li> appear inside the <ol>, then it's a "Basic" task topic.
- <p>, <ul>, <note>, and <note caption="Tip"> tags that appear in an <ol> (these are what we sometimes call "indented" paragraphs in the SAM files) can fit as part of the DITA task content in an <info> section.

This analysis forced us to rethink the information we had written. We can use the nuances implied by our authoring conventions to interpret the intent of what we had actually written. This analysis exposed more ideas for the design of our content models, it showed that we needed greater specificity to avoid making these mistakes in the future, and it highlights a need for greater consistency within more specific infotypes - more rigorous content writing equates to better semantic specificity. The exercise compelled us as designers (and as current or former technical authors) to relate past guidelines and practices to the intent in the eventual DITA design. Now, we can tell writers, "If you did so and so in the past, we've prepared a DITA structure for which you would now do this." This is still doing task analysis as we've requested writers to do all along, but with explicit content models now for the actual writing. This replaces a common earlier practice of providing "cookbook" examples of markup that tended to represent the desired visual structure of various writing styles.

We now have Concept, Task, and Reference at the high level, and specialization allows the creation of more specific vocabularies within those categories of information types. Determining the information type for a topic begins at the point of selecting an appropriate topic container for the content at hand. We take away a set of choices by not allowing a task within a concept, for example, but this provides better enforcement and consistency within the information, and reduces the opportunities for mistakes. We will have to continue the process of "uneducating" certain past practices; if you see you can use an OL (ordered list) within a concept, that doesn't mean you can develop a task there! In contrast to actual task markup, an OL within a concept topic has ambiguous meaning.

We wouldn't have understood this as well if we had not done the various analyses to see how even well-skilled authors use different styles and multiple markups, to represent the same thing. By mapping our understanding of topic types into the DITA structure, we can improve how authors create appropriate information. And in explaining how to use the DITA structures, we will be able to leverage both the technical writer's understanding of information types and reinforce its use through the structures in DITA.

There's more than one way to present a task, concept, or reference topic! While DITA doesn't straight-jacket all content into a single model for these topics, it does reduce the choices and provide for a higher level of consistency and format within these topic types. In addition, the DITA specialization architecture provides a consistent method for identifying more constrained information models, and applying those models to more specific DITA topic types. For example, differences among reference topics illuminate the differences in writing an API for C++ versus an API for Java, and DITA provides a way to accommodate those differences within a consistent overall information architecture.

Status

We are in the phase where we are using a more experience-based approach in validating our assumptions.

We recognize that the DITA design is a prototype; its design -- including DTDs and style sheets -- will continue to evolve through feedback from the interested user community -- you!

Use the DITA newsgroup (ibm.software.developerworks.xml.dita) to discuss the use of the DITA DTDs and style sheets. Through discussion in the DITA newsgroup, we'll apply the significant ideas to subsequent refreshes of the package. The forum will be actively monitored by the DITA project's architects, Don Day and Michael Priestley, among others.

DITA has already generated some active interest among other information specialists. Evidence from several newsgroups suggests that people are already testing DITA principles at their own companies. The feedback and the interchange with the public community has been invaluable for everyone involved. If you believe that DITA principles may be of value to your company, we certainly encourage you to investigate it and apply it as appropriate.

Future Directions

As DITA stands now, we have:

- A core set of DTD modules, to allow authoring in the core topic types and to provide a base for creating more specialized topic types
- A set of guidelines and samples for creating content, transforms, stylesheets, and new topic types in the DITA architecture

There is plenty of work left to be done:

- Navigation maps, for defining how topics get collected or related for particular delivery contexts
- Common specializations, to allow for interchange at more specific levels in common content areas such as contextual help and API documentation
- Content management, to provide for centralized tracking and versioning of topics in source repositories
- Additional tooling support, to allow custom authoring environments to be reused with specialized content in the same way that stylesheets and transforms can be reused
- Personalization processes, to give the user more direct control over the use of specialized markup and content

Hopefully over the next year we will see the discussion and experimentation that DITA needs to continue growing and evolving. Our existing architecture provides a base, we are looking to a wider community to help find out what happens next.

Summary Conclusions

- IBM XML DITA leads the way in topic oriented support.

- DITA enables more rigorous semantic specification of topic types based on audience and author requirements.

- DITA goes beyond the current accepted DTDs by allowing information to be organized and delivered via multiple means, (i.e., helpsets, manuals, etc.).

- IBM's development of DITA demonstrates our commitment to improving technologies and keeping the improvements open to the public.