The Business Value of XML and the Darwin Information Typing Architecture (DITA)

Executive Summary

Information developers at IBM face increasing pressure both to maintain high levels of customer satisfaction and reduce the Total Cost of Ownership (TCO) and Total Cost of Deployment (TCD) of information supporting IBM products. As a result, IBM Corporate Information Development formed the IBM User Technology Workgroup on XML to look at how to reduce the cost of ownership and deployment. After reexamining IBM's basic approaches to creating and delivering product technical information, the Workgroup recommended that IBM move away from a monolithic document model and toward a topic-based reuse model.

To respond to the demands for more efficient TCO/TCD, as well as provide better access to product content, the Workgroup developed an XML-based information architecture called DITA, which is short for Darwin Information Typing Architecture. Using this architecture uniquely positions IBM to take advantage of XML to deliver content to users. DITA improves the overall flexibility of information designs, facilitates better writing, ensures information completeness, and promotes sharing and reuse, both within and across product development teams. Using DITA, IBM will reap significant long-term TCO and TCD benefits that will better position us for continued growth in increasingly competitive worldwide markets.

Introduction

Those who develop, produce, and deliver technical documentation know that "information is king." The emergence of the Web makes the easy availability of information and content both ubiquitous and expected. At the same time, information developers face increased pressures to reduce both the Total Cost of Ownership (TCO) and the Total Cost of Deployment (TCD) of information. With some external estimates of development, translation, and production costs for a printed book starting at 30 cents per word, IBM also wants to minimize costs related to every word.

IBM Corporate Information Development formed a Workgroup to look at these issues. The Workgroup actively explored ways to leverage the power of XML, not only to address cost concerns but also to provide breakthrough access to product content, with significantly greater ease-of-use and overall customer satisfaction. To accomplish these goals the Workgroup developed an XML architecture. The XML architecture is named Darwin Information Typing Architecture - or DITA - because the architecture uses the principles of specialization and inheritance associated with Charles Darwin's theory of evolution.

The Context

Before delving into what DITA is and what it can offer, here's a little necessary background about the issues that prompted the Workgroup to create DITA -- namely, the business value of electronic content delivery, topic-based authoring, and XML.

Electronic Delivery of Content

IBM knew it had to deliver content electronically to eliminate or reduce costs associated with printed publications where possible. Electronic delivery of content brings dramatic advantages, both for the information developer and for the customer.

Highly Available

For customers who have access to the Internet, information is readily available; information developers do not need to package and then deliver information to customers. Instead, customers with Internet access can go to a Web site and download the latest material when they want. With software products especially, there's often a considerable lag between the

end of the documentation writing cycle and the code freeze and ship date of the product. Making product information available electronically over the Internet provides an opportunity to keep that information always up-to-date. In fact, using the Internet as the primary publication vehicle for a product may completely eliminate the physical delivery cycle.

Minimized Distribution, Warehousing, and Scrapping Costs

Electronically distributing information is significantly cheaper than printing and distributing printed publications. By some estimates, the cost of delivering an entire electronic document is nearly equal to the cost of delivering one printed page.

Electronic delivery also eliminates costs associated with storing many copies of printed stock. These stocking costs include warehouse space, fees, personnel, and management. Maintaining publications that are delivered electronically costs significantly less, because you pay for relatively cheaper hard disk space, not warehouse space. In addition, updating information in an electronic publication avoids the expense of scrapping obsolete copies. Furthermore, at the end of the life-cycle of the product, there are no preprinted copies of publications to scrap.

Incremental Updates and Staged Delivery

With electronic delivery, you can update information as often as the infrastructure for delivering it supports. You can simply provide updated content and correct minor errors directly on your Web site and avoid scrapping the entire stock of printed publications or delivering updated Readme information or technical newsletters. Providing frequent Web updates also has an additional benefit: customers come to expect that information on the web site is up-to-date, and this expectation may drive customers to visit the site often. The result is increased customer confidence and satisfaction -- two tangential benefits that have great value within IBM.

You can augment and improve product information in a continuous process of incremental publishing. For example, field and support engineers can add new material and examples from their experiences with the product. Information about tuning product performance or upgrading to a new release can draw on real-world experiences and be kept up-to-date.

Implementing an incremental publishing strategy facilitates delivering information in stages. For example, an initial product release may include basic information. Then as the

product becomes more widely available, you can provide more complete documentation. A major win for a product team is that the time between the creation and the publication of information no longer depends on print production schedules. Consequently, information developers have the time they need to produce accurate technical information.

Resource Management and Workload Balancing

Schedules for electronically updating publications optimize workload and resource management. With print-based delivery, a significant amount of time and effort go into print and production cycles. In addition, updating existing printed content is often tied to product ship schedules. With electronic delivery, the updating of content is not tied directly to a product ship schedule. Instead, you can balance the workload of a technical writing team by scheduling updates of Web-based content for times when direct product work might be low.

Topic-Based Authoring

To better position IBM to take advantage of electronic delivery, the Workgroup examined the current information architecture of our publications and evaluated the resulting use and reuse of that information.

The Workgroup quickly ascertained that our current model of presenting information in large monolithic documents does not address the needs of different types of users who use information in different ways. In addition, as products become increasingly composed of integrated components, pressure mounts to integrate and reuse information across products. This pressure prompted the Workgroup to reconsider the use of the monolithic document model and evaluate a move to a topic-based reuse model.

Monolithic Document Model - The Problem

In the monolithic document model, a documentation set includes one or more books that contain all content related to a product. These books typically include information about product evaluation, support, use, installation, configuration, troubleshooting, service, maintenance, and so on.

A close look at a traditional documentation set reveals that information is often repeated in one or more books. For example, a description of installing a product may appear in both a planning guide and an installation guide.

Different types of users, each with different information needs, may need content from different books in a documentation set. For example, service and support staff may need information from all of the books; marketing staff may need only information from a planning guide; programmers require just API reference information. Most technical books are written for a specific audience: administrators, programmers, designers, or end-users. Consequently, tailoring the content of a monolithic documentation set to meet the needs of specific groups of users requires substantial rewrites of the same information.

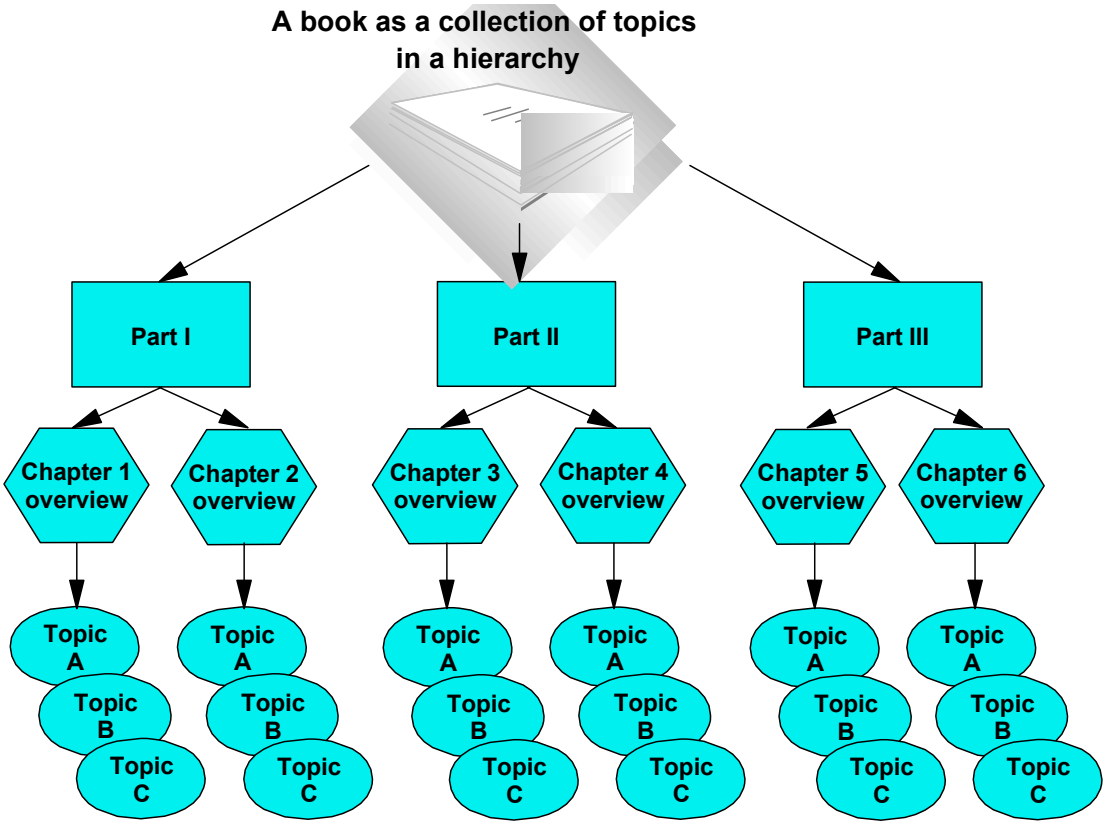
Similarly, because a software component may be included in multiple products, writers must often repeat or rewrite information about the component for each product's documentation.

Within the monolithic document model content reuse is based on how writers cut and paste the information from the same source. This type of content reuse has several major flaws. For example, when a writer updates one copy there is no guarantee that other copies will be updated. Even worse, different writers might update difference copies with different content, resulting in incomplete or incorrect information. This proliferation of dissimilar copies creates problems for the writer who must eventually synchronize the content in the copies.

In contrast to the monolithic document model the topic reuse model plans for and builds reuse directly into the information architecture.

Topic Reuse Model: The Solution

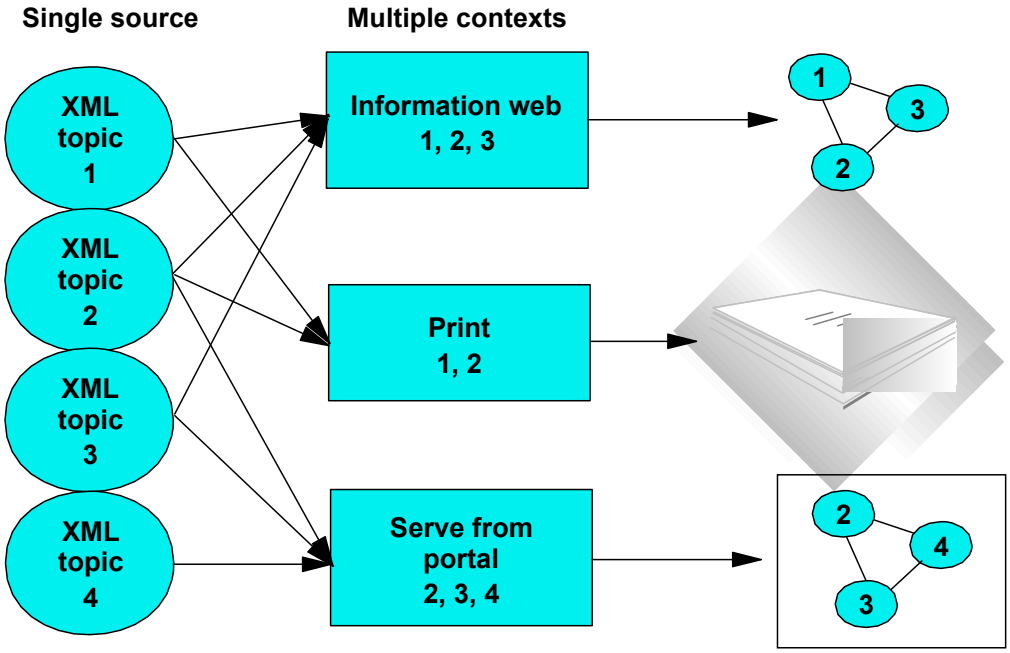
In the topic reuse model, writers create and store information about a specific content area as a set of discrete topics, For example, in this model a book is a collection of topics in a hierarchy, as shown below.



Each topic provides content that relates to a larger concept. In addition, each topic can stand on its own, independent of the topics around it and the structure -- chapters, parts, or books -- that holds it.

Storing information as topics facilitates the creation of various types of outputs. For example, from a single set of topics you can produce a printed book, a collection of help topics, and a server-based information portal, to name just a few.

Single source and multiple contexts



When multiple writing groups use topic-based authoring, you can merge content created by one group with information created by another group. For example, you can combine topics written by a marketing group with topics written by a support group to describe a specific sales scenario. Or you can merge topics originally written for individual publications such as a functional specification, a customer support note, and a marketing white paper into a product documentation set. Then you can include the same topics, untouched, in training materials or a marketing overview.

XML Adds Value To This Solution

Acknowledging the value of moving to topic-based authoring and electronic content delivery was just the first step. To envision a complete solution, the Workgroup investigated the use of XML.

XML

XML is an abbreviation for eXstensible Markup Language, which is a text-based tag language that is similar to HTML. Unlike HTML, however, XML has no predefined tags.

How tags are defined and how information is stored in XML is determined by the authors, information designers, and applications that create and process the XML content.

At IBM, considering a move to XML is based on several unique features of XML:

- **Open standards.** XML is an open industry standard, controlled by the World Wide Web Consortium (W3C). XML provides an application- and system-independent format for sharing and exchanging content, made better when sharing organizations use an agreed upon tagging system, such as defined in a document type definition (DTD) or other schema.
- **Separation of form from content.** XML separates form -- that is, presentation, look, and layout -- from content. This separation makes it possible to present the same source content in different formats -- for example, as Web pages, printed pages, or other delivery media. Consequently, a program can transform the presentation to give an entire Web site a new style without changing the underlying content.
- **Extensible and meaningful tags.** XML is a "meta language" for describing user-defined markup and tags. Tags can be designed to have a specific meaning and label specific content. For example, a zip code in an address might use a tag called “zipcode;” a step in a procedure, “step.” Using the tags, an author marks the content according to its meaning, and then based on the tags, processing systems such as search and personalization software filter and present the content targeted to specific groups of users.
- **Consistent tools.** The reliance on open standards provides the foundation that ensures a wide variety of tools for creating, managing, and deploying XML content will emerge.

Introducing DITA

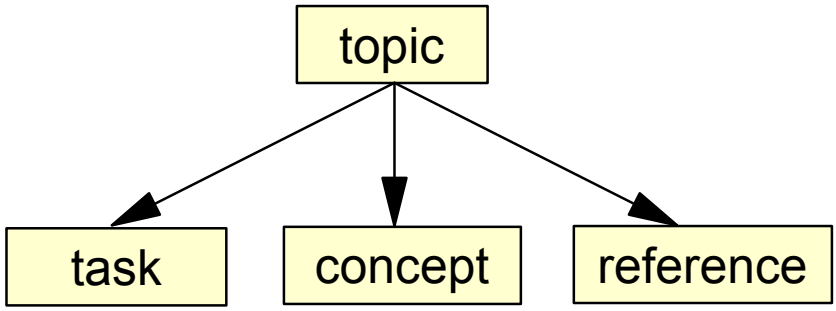
Darwin Information Typing Architecture (DITA) is an XML topic-based information architecture and collection of DTDs for creating and delivering technical information. Because DITA is based on XML it leverages the advantages inherent in XML and extends beyond those advantages.

DITA Is Topic-Based and Modular

DITA is an architecture for creating and delivering topic-based, modular technical information. In DITA, the core information unit is a topic, which describes a single task, concept, or reference item. Because DITA is topic-based, DITA content can be combined, recombined, and reused to create online help, printed books, Web-based information centers, and product support portals, to name a few.

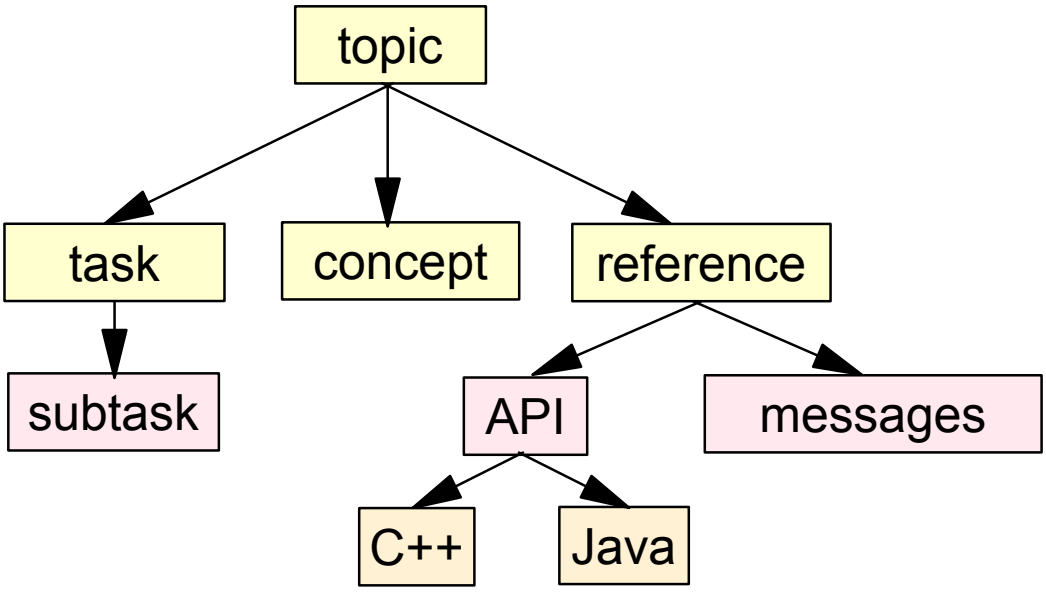
DITA Is Based On Information Types That Can Be Specialized

An *information type* defines the role of a topic. DITA includes three information types derived from the base topic type: concept, task, and reference. A task topic presents the step-by-step procedure for a task. Task topics answer "How do I?" questions for a specific task. Concept topics provide the reason behind the tasks by defining terms and explaining concepts. A reference topic provides information about a specific command, message, program option, API, and so on.



In addition, DITA provides an XML-based architecture for extending these three basic types to specialized topics. For example, in the following figure, the basic reference topic

has two specialized information types -- API and messages -- and API has two additional information types -- C++ and Java.



Using specializations, you can define increasingly specific content structures. The DITA specialization architecture makes it possible always to "fallback," or return back "up the tree" to a more generalized form of content.

DITA Example

The following is a DITA task topic that describes how to change mail preferences.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<task id="H_CHANGE_MAIL_PREFERENCES">
 <title>Change Mail Preferences</title>
 <taskbody>
     <context><p>To change any preference:</p></context>
     <steps>
         <step><cmd>Click Preferences.</cmd></step>
         <step><cmd>Specify mail preferences.</cmd></step>
         <step><cmd>Click Save & Close.</cmd></step>
     </steps>
 </taskbody>
</task>
```

Notice how the tags themselves identify the type of content.

- "task" indicates that this is a procedure. Because all procedure topics use the task tag, a user or a process can choose to present only "tasks."
- "context" describes the purpose of the task.

- "steps" provides the procedure itself, with each step contained in a "step/cmd" tag.

In a DITA "generic" topic format, you might use an ordered list/list item (ol/li tags) structure to code the steps. However, using a "steps" tag more clearly denotes that this is an ordered list of steps that describes how to perform a specific task, not just a numbered list of information.

DITA allows specialized information types to be defined as extensions of an existing "common" type. The specialized information type inherits common structures and maps new, specialized elements back to existing elements in the common, or ancestor, type. DITA maintains information about ancestor types directly in the DTD that defines the specialized type. Therefore, it's always possible to transform a specialized type back to a more generic ancestor type. DITA also makes it possible to follow the reverse path and map a generalized topic back to its specialized descendant.

This "specialization with a fallback" provides flexibility. As product requirements and circumstances change, you can migrate content to new information types or tailor it for specific audiences, still keeping a fallback to the more generic common types for exchange and compatibility with other information sources and providers. For example, suppose you design a specialized reference topic to accommodate a product-specific API. If you ever need to exchange that product-specific content another team, you can use the DITA architecture to ‘fall back’ to the generic reference topic, the format of which is consistently implemented across teams.

Additional Benefits of DITA - The Pieces

DITA makes it possible to realize the benefits of moving to a topic-based reuse model for information development in the following ways.

Single-Source Authoring, With Multiple Outputs and Multiple Reuse

DITA supports single-source authoring and reuse by providing an information architecture and reuse model that describes technical content. Using DITA, you create content once and deliver that content in different layouts, in different media, and for different audiences.

Flexible and Extensible

Because specialization is built into DITA, when you need new topic types, you change the existing DITA DTD, rather than rewrite a new DTD. Associating new specializations, through inheritance, with the in-place specializations, facilitates quick support and development of the information architecture as it evolves.

Specialization of Content and Presentation

Because XML separates form and content, DITA accommodates the specialization of both. Similarly, extensible stylesheet language transformations (XSLT) and other XML processes can work at either the specialized or generic levels of DITA. This means that a common XSLT transform can handle the general case, and more specific specializations can derive from this common presentation model. For example, a generic specialization of the presentation of APIs might be made specific for C++ APIs.

Faster Response Through Specialization

Using DITA you can quickly respond to customer demands for new and updated product information. You can implement new information models quickly through a new DITA specialization, without the need to start all over from scratch.

Enforcing Business Architecture

Through DITA specialization, you can create and enforce a consistent information architecture. For example, a specialized topic used to document a C++ API includes rules that force writers to compose a set of required content. For example, within DITA you can define how to document a return value. Users of this kind of highly structured content become familiar with the consistent structure of the information and find they can almost intuitively locate topics. For example, searching for “an install step for an expert” or searching for “return values in C++.”

Portable Through Standards

Using DITA, product groups and external business partners can easily share and exchange content. Third parties can use "common" transformation and presentation models with DITA, or create specialized processing to offer views and presentation of content that is company- or brand-specific. This content portability is critical for maintaining

arrangements with third-party partners and for ensuring that a writing team remains productive through business reorganizations, mergers, acquisitions, and spin-offs.

Focused Content and Better Writing

Topic-based authoring produces better writing. Categorizing content into concept, task, and reference topics ensures that users can perform tasks faster because the information is focused. In addition, users can search for information based on their company role, their job responsibilities, and their task goals.

Common Practice

DITA implements standard information types, concept, task, and reference, that are taught in technical writing classes. Therefore, technical writers using DITA follow common authoring practices.

Information Completeness

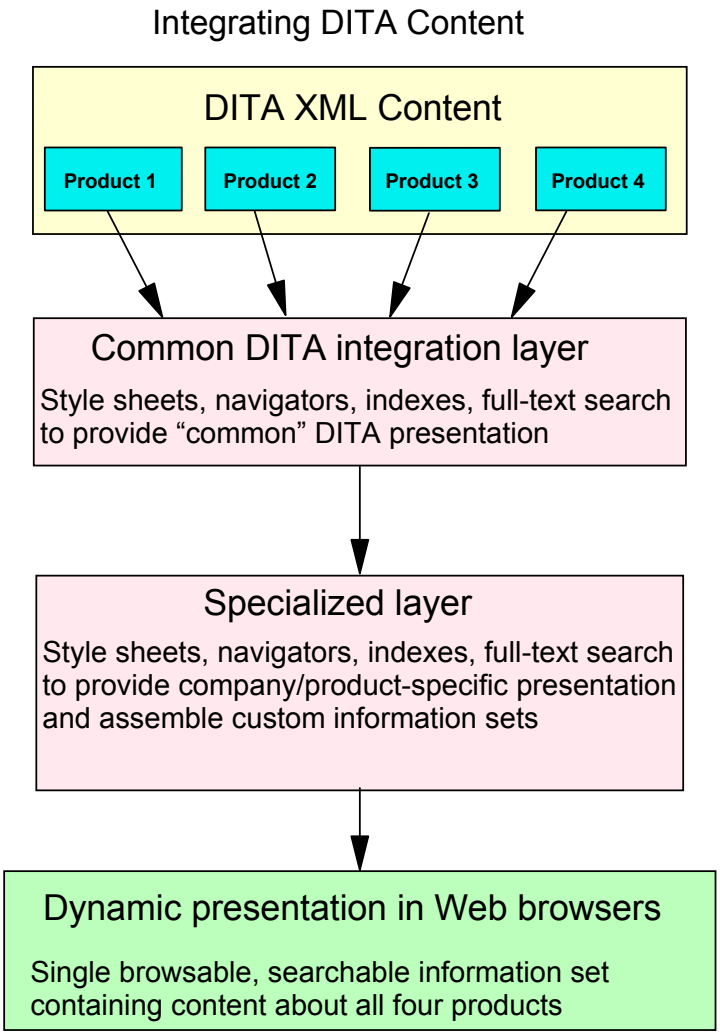
Because DITA supports concept, task, and reference topics, writers and editors can quickly determine if a new function has been completely documented. For example, finding task topics that are not supported by concept topics may indicate that additional writing is required.

Custom and Dynamic Publishing

With DITA, writers can assemble topics about a specific set of issues and publish them as a unique deliverable. For example, a customer support team might compile from diverse sources a set of topics that provide a customized solution to a problem reported by a major customer.

Content Integration - Pulling it Together

The combination of XML and DITA present compelling advantages for sharing and integrating content within and among groups. The following figure illustrates how multiple groups can share DITA content and deliver one integrated information set.

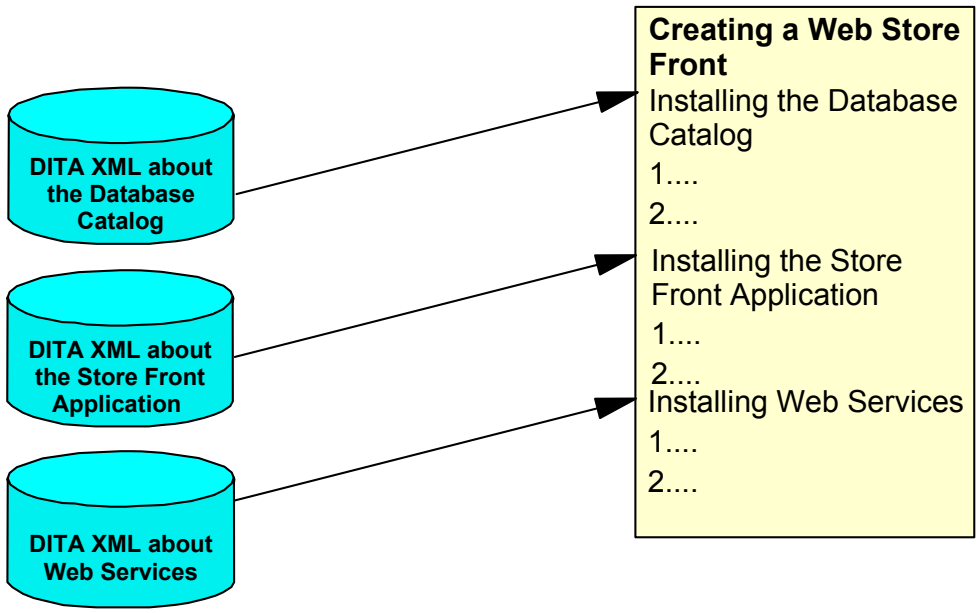


XML itself provides a platform and standard that makes it possible to process, exchange, share, and integrate content from different sources. The common tagging standard defined by the DTD facilitates content exchange and integration.

DITA makes it possible to deliver the same content, without re-editing or re-writing it, to different presentation devices and different formats.

Specifically, DITA provides a "common" integration layer to exchange, share, and integrate XML content. This layer combines different sets of topics and merges them into a common presentation and view. Adding a common set of navigators, searches, and topic styles increases the consistency of the content. In this way, it's possible to provide an information center or other integrated information deliverable that brings together content from a variety of sources and product lines.

Pulling Together Content from Different Sources



Using DITA you can standardize the presentation of information to provide a unified user experience. Contrast this with existing information portals that present information in a variety of formats including HTML pages, navigators, content models, PDF documents, ASCII downloads, and so on. Because each format looks and behaves differently, users become confused and often cannot determine that the content has any overall coherence. On the other hand, an XML-based DITA information system that contains a variety of content sources can provide a more unified user experience.

Using DITA, you can maximize limited resources to implement the presentation. Currently, if a presentation standard can't meet a specific need, a team creates a customized solution. This leads to duplication of effort across teams, with resulting limitations in each presentation framework. With DITA, it's easy to pool efforts across teams to customize the content and presentation for specialized cases.

Tagging content based on information type makes it possible to tailor the presentation for specific groups of users, such as system administrators, programmers, end users, designers, and so on. Each role-specific presentation of content derives from the same underlying topic or collection of topics, or from specialized merges of content from multiple topics. The amount of information presented becomes much less, and thus considerably more

manageable than in the monolithic view, which can only accommodate a single large body of content at once.

Similarly, DITA can provide content at different "levels" of detail, as "layered" information. For example, a brief summary might comprise the top "layer" of a topic. DITA can then reveal increasingly specific details about the topic, as requested by the reader. In this way, DITA content becomes customizable and targeted at specific information needs, for view by individual readers, partners, and customers. Topics can be assembled based on individual user profiles and session histories.

Finally, at all stages, DITA makes it possible for business partners and other third-parties to work with their own content to create custom help systems, generate interactive assistance, pull together comprehensive product information centers, and identify specific subsets of content that address specific customer needs.

Managing the Change

A shift to XML and DITA brings with it significant challenges, both technical and organizational. One big challenge involves identifying where XML and DITA fit within the overall strategy and purpose of the organization. A second major challenge involves managing the change involved in realigning organizational and work patterns to fit the new model.

Migrating Investments In Legacy Content

Most organizations have a huge amount of legacy content that may have been authored in a variety of ways. For example, writers at IBM currently author and store content in SGML, HTML, and other formats. Some IBM writers use a book-oriented publishing tool to author content as chapters. Others use a help authoring tool to create content as help topics. In each case, moving to an XML-based authoring and delivery system must support transforming and migrating legacy data to XML. Rather than having an open-ended policy on migration, it is critical to develop a set of strategic guidelines to determine if, when, and how to migrate legacy content.

Equipping Employees To Deal With The Change

Authors, editors, translators, developers, and other employees affected by the change to an XML-based information system need training and support for using new processes and tools. For example, if you move from a book-oriented authoring model, provide training in topic-based authoring and anticipate the type and amount of cleanup that writers must do after you migrate the legacy content.

Content Integration Issues

While DITA and XML provide advantages for integrating and sharing disparate content, there are also some limitations. By definition XML enforces consistency in content structure and organization. For producing technical content, DITA specializations provide even stronger content consistency; better overall agreement on basic structures and vocabularies in the source XML content makes for better overall consistency.

However, even though DITA's topic-based architecture provides the foundation to structure specialized topics and the information in them, the role of editorial standards and practices remains important. Within the structure of each DITA topic, there's still room for variation and choice. DITA makes it easier to spot and track inconsistencies. Nonetheless, writers need editorial standards and guidance in making these choices.

Assess Organizational Effect

Moving to XML results in changes not only to processes but also to the overall organization. Therefore, you must identify new areas of overlap brought about by the move to XML and reassess old organizational boundaries. For example, a graphic services or publishing department that makes sense in a book-oriented environment may no longer be required in an XML-based publishing environment.

Tools To Support DITA

Tools that fully support XML and DITA are just starting to emerge and are still under development. The Workgroup is investigating specific tools for:

- **Content management.** A system for creating, building, deploying, and managing XML and DITA content is a critical need. Existing solutions are not widely deployed.

- **Authoring.** Tools for editing XML and DITA content, developing styles and formatting, validating, browsing, and building to different outputs are just starting to emerge.
- **Navigation, linking, and search.** XML query, navigation and topic maps, and XML linking are in various stages of maturity and lack implementation support. Similarly, current off-the-shelf full-text-search solutions are not optimized for XML.

DITA Goals and Directions

IBM is interested in making DITA the standard for developing and deploying technical XML content. Working with additional partners is essential to accomplish this goal.

Partners who can provide diverse information sets will help us determine how far the DITA information model can stretch to incorporate additional content. Can DITA be used for policy manuals, marketing collateral, technical articles, support bulletins, and so on?

Conclusions

This paper provides some of the details and background about DITA, an XML-based "information development" architecture under development at IBM. DITA, by its reliance on XML, provides the ability to deliver content in a variety of outputs with minimal "touch" and "re-touch" costs. Minimizing content change, increasing content reuse, using specialization to support changes in information architectures -- all aim directly at reducing the Total Cost of Ownership (TCO) for technical product content and documentation.

IBM is implementing DITA as an internal standard for technical product content. The Workgroup designed DITA to achieve significant long-term TCO and TCD benefits for IBM. These benefits will position IBM for continued growth in increasingly competitive economic markets. The Workgroup invites your comment and collaboration in working to help further refine, extend, and leverage DITA.

Resources

The following links provide additional information about XML and DITA.

- **Introduction to DITA**

  http://www.ibm.com/developerworks/library/x-dita1

- **Specialization in the DITA**

  http://www.ibm.com/developerworks/library/x-dita2

- **DITA FAQ**

  http://www.ibm.com/developerworks/library/x-dita3

- **Make IT Easy paper on DITA**

  http://www.ibm.com/ibm.easy/eou_ext.nsf/Publish/1819

- **Sample DITA content**

  Lotus Notes Help: http://notes.net/notesua.nsf/find/notes503xml
  Lotus XML InfoCenter: http://notes.net/notesua.nsf/find/xml10info

- **XML at IBM developerWorks**

  http://www.ibm.com/developerworks/xml/

- **XML at Lotus**

  http://www.lotus.com/xml

- **XML authoring and publishing**

  Priestley, M., Hargis, G., Carpenter, S. (2001). “DITA: An XML-based Technical Documentation Authoring and Publishing Architecture.” *Technical Communication*, Society for Technical Communication, 352-367.

Authors

This paper is a product of the IBM User Technology Workgroup on XML. Responsible for the development of DITA, the Workgroup consists of 30 technical professionals who have been working on DITA since 1999.

Dave A. Schell and John P. Hunt are the primary authors of this paper, and Phyllis A. Sharon is the editor.

- Dave A. Schell is the User Technology Corporate Sponsor and management representative for the Workgroup. He can be contacted at: dschell@us.ibm.com
- John P. Hunt is a Senior Software Engineer at IBM and member of the Workgroup. He can be contacted at: john_hunt@us.ibm.com
- Phyllis A. Sharon is a Principal Editor at IBM, working on Lotus Software. She can be contacted at: phyllis_sharon@us.ibm.com

The ideas developed in this paper are heavily based on the ongoing Workgroup discussions and the published work of other Workgroup members; specifically we reused major portions of the article by Priestley, Hargis, and Carpenter cited above.