Scenario-Based and Model-Driven Information Development with XML DITA

Michael Priestley
IBM Toronto Lab
8200 Warden Ave.
Markham, Ontario, Canada
mpriestl@ca.ibm.com

ABSTRACT

In this paper, I describe how I followed an end-to-end development process in the development of the user's guide and help information for XML DITA, using scenarios to define my information needs and maps to describe my information model. By using technology driven by maps and scenarios, I was able to keep the information focused on user goals and requirements from its inception through to its final form. The paper will also look at how an integrated end-to-end process can help keep information on track through staged delivery of content to ensure early and ongoing feedback, and will look at some future opportunities for further integration in the stages of the information development process.

Categories and Subject Descriptors
I.7.2 [Document Preparation]

General Terms: Documentation

Keywords
Model-driven development, DITA, XML, scenarios, maps, information development process, scenario-driven development.

1. BACKGROUND

The Darwin Information Typing Architecture (DITA) is an XML architecture for creating and publishing information, especially technical information. It is based on several key principles:

- Topic-based information
- Information types and domains
- The separation of content (topics) from context (maps)
- Extensibility and customizability
- Reuse

DITA was created based on best practices in technical authoring such as task orientation, chunking, information typing, and minimalism. It was also shaped by developments in software architecture, including object-oriented and component-oriented software. Because software and technical information share similar challenges, it is not surprising that similar approaches have emerged in both arenas. However, the similarities are not based on simple application of existing techniques to new areas: the similarities run deep, but are not equivalencies.

1.1 Topic-based information

The topic is the smallest independently maintainable unit of content. Topics must be able to stand alone so that they can be understood when they are encountered out-of-context, for example when a user finds the topic through search, an index, or by following a link. Even books can benefit from being written as collections of topics, since a book's logical organization should not preclude its use by people who scan it, flip through it, or use its index.

1.2 Information types and domains

Information types are kinds of topics: for example, tasks, or concepts, or reference information for a particular product or subject element. Industry best practices have identified a large number of different information types, which DITA reduces to three (concept, task, and reference), all derived from a common base type, the generic topic. This simple hierarchy can be extended as required to cover various possible specialized kinds of information type.

Domains are kinds of content within topics: for example, one task might be concerned with user interfaces, and include content like window names and menu choices; another might be concerned with programming, and include content like method names and code examples; a third task (for generating code based on GUI selections, for example) might include both kinds of content. Domains, like information types, are organized into a hierarchy, with more specific domains specializing more generic domains. For example, a menu choice could be a specialization of the more general class of user interface controls, which in turn could be a specialization of the most general class, keywords.

1.3 Content and context

Because topics are written to be read in a variety of different contexts, they are most useful if we can factor out any context-specific elements (such as links to other topics) that might not be appropriate for all contexts. This separation lets us quickly update elements such as links and navigation without having to edit the topics involved.

The separation of content from context becomes particularly important in a component-oriented environment, where different products share common components and the component's information needs to be integrated into each new product set without interfering with the other product sets.

Context may vary depending not only on the product (for component-oriented information), but also on delivery format (PDF versus Web versus embedded help), audience (programmer versus administrator versus tester), or learning activity (background learning versus task failure recovery versus troubleshooting). As a result, our context format needs to be flexible enough to cover a variety of organizational, navigational, and metadata elements, which may need to vary based on these different dimensions of context.

1.4 Extensibility and customizability

DITA can be extended through the definition of new information types, domains, or kinds of maps, as well as through the addition of new output processes or extensions to processes. When you add new information types or domains, through a process called specialization, you do so by extending existing ones, and adding to a hierarchy of types or domains that allows inheritance of design of elements and inheritance of processing rules. For example a new type of task might have most of the same elements as a regular task, so you can use the existing elements without redeclaring them. And because the new type of task inherits from the regular task, output transforms that work for tasks will automatically work with the new type of task as well. This means that you can design new kinds of content, and immediately start producing output that includes a mix of new and old designs. Specialization mechanisms are discussed in more detail in [1].

1.5 Reuse

DITA is in many ways an architecture of reuse: the XML Cover Pages called it the "Holy Grail of content reuse" [2]. Some of these ways are covered in [3]. The capabilities include maps for managing collections of topics, conditional processing (filtering and flagging) for managing differences within a document (either a topic or a map), and content referencing for any valid reuse of information between topics or between maps. The important thing to remember is that reuse is not an end in itself: it is the enabler for creating user-focused content consistently, quickly, and accurately.

2. THE END-TO-END PROCESS

DITA is based on best practices in information development and was developed to support those best practices both in the authoring and in the delivery of technical information.

In the following sections, I will describe how I developed the content for the DITA user's guide and help set, following what I perceive as the best practices in the industry, and for the most part helped and supported by the DITA architecture. This experience validated some important features of the architecture and also pointed to some opportunities for further enhancements.

I started with the basic assumptions that my information needed to be:

- Audience-focused
- Scenario-driven
- Task-oriented
- Accountable to the user

In order to achieve this, I developed the content through a number of key stages leading up to the full internal release of the information and through follow-on education.

The key stages were:

1. Identifying the audience and defining roles
2. Identifying scenarios: descriptions of situations that exemplify the problems our users are trying to solve, and how we expect them to use the product to solve them
3. Developing scenario materials, including tutorials and samples, that leverage the background work done in the scenario documentation
4. Identifying and organizing tasks into a task flow, starting with the scenarios as a resource for identifying potential tasks and sequences
5. Identifying and organizing concepts into a concept hierarchy
6. Writing tasks and concepts
7. Relating tasks, concepts, and reference information to each other
8. Developing an integrated reading hierarchy
9. Indexing
10. Producing PDFs, help sets, and Web organizations of the information

At each stage, I reviewed my progress with other members of the development team, usability experts, and, most importantly, early users of DITA within IBM.

2.1 Identifying audience and roles

We used surveys to establish the general level of skill and educational requirements for new DITA users within IBM. This helped us identify three main audiences:

- Experienced SGML authors, who were familiar with tag-based authoring and the separation of content from presentation, but would need some general education on DITA tags and might require help in changing over to a topic-oriented architecture
- Intermediate SGML authors, who would need reinforcement on the XML mindset (separation of content from presentation), as well as education on the specific tag set and, potentially, education on topic-oriented architectures
- HTML authors, who would need education on XML and on the specific DITA tag set, but typically needed less education on topic-oriented authoring

We also identified several roles related to DITA, as described in [4]:

- **Type architect.** Analyzes topic types needed to accommodate content being produced and defines new topic types if needed.

- **Information architect.** Analyzes the overall structure of the content, groups it into topic collections, and defines maps that describe the relationship of topics to each other.
- **Information developer.** Writes and edits topics, according to the topic-type standards established for the project by the type architect.
- **Build developer**. Processes the DITA source topics into various formats, as needed for product deliverables.
- **Information designer.** Establishes the "look and feel" of the output presentation.

The DITA information needed to initially support the information architect and information developer roles, both of which would be necessary in every team. The information architect role would be new to many SGML authors and to some HTML authors, and would be using unfamiliar methodologies (DITA mapping), so it would require special support.

2.2 Identifying scenarios

Scenarios are descriptions of how a product might be used in the context of solving a particular problem or achieving a particular user goal. In the context of product development processes such as the Rational Unified Process [5], these would be something like use case instances. Typically these are used to identify requirements for the product being developed. Scenarios are also used for the design of human-computer interaction, including interaction with information systems [6].

In this paper, however, scenario-based information development goes beyond just describing ways in which users might access the information (although that is certainly useful in its own right). Rather than focusing on information interaction scenarios, this paper is instead focusing on the use of product scenarios for the design of an information model. In other words, stories about how the product should be used can form the basis for descriptions of how the product should be used. While this point may seem obvious, it is important to distinguish between the two possible applications of scenario-based information development: on the one hand, using information interaction scenarios to inform information system design, and on the other hand, using product interaction scenarios to inform information system content. Both applications are appropriate and useful, but this paper concentrates on the second case, building on connections between product and information development processes already described in some detail elsewhere [7].

I looked at the list of scenarios that had been identified during development planning for the internal DITA toolkit. These included scenarios like “Creating navigation”, “Creating tasks”, and so on. However, several of the scenarios needed to be updated to focus on the user perspective rather than just capturing development requirements, and an overall end-to-end perspective was missing. In the end I identified several key scenarios that I wanted to specifically support, with an overall flow as follows:

1. Creating information models
    a. Creating task hierarchies
    b. Creating concept hierarchies
    c. Creating reference hierarchies
    d. Creating cross-type mappings
    e. Creating reading-oriented hierarchies
2. Creating content
    a. Creating topics
    b. Creating tasks
    c. Creating concepts
    d. Creating reference topics
3. Conditionally processing content
4. Reusing content
5. Producing output

2.3 Developing scenario materials

I prioritized the scenarios and developed samples and tutorials for the first main scenario: creating information models.

I gave the tutorials and samples to early users for review and testing, and modified my content based on their feedback.

Next I developed tutorials and samples for the second main scenario: creating content. These also went through a cycle of review and testing with early users.

I did not have time to develop scenarios for the more advanced processes (conditional processing and reuse), nor for the (relatively) straightforward output activities. However, I now had tutorials and samples for the main user tasks for information architects (creating an information model) and for information developers (creating content).

By the end I had eleven tutorials, taking up approximately 70 printed pages, as well as accompanying samples.

2.4 Identifying and organizing tasks

Based on the scenarios and tutorials, I developed a list of tasks and organized them into a hierarchy. Where possible, I followed the same order and organization as the scenarios, including tasks for linking, conditional processing, content reuse, and generating output, which were not directly covered by the tutorials.

I authored the hierarchy initially in a plain text document, using indenting to indicate hierarchy, and I reviewed it with other developers and architects, as well as with users and a usability expert.

I then re-created the hierarchy as a DITA map, using attributes to indicate which parts of the hierarchy were sequences:

- Creating maps
    1. Adding topic references
    2. Defining collections
        - Defining sequences
        - Defining families
    3. Defining hierarchies
        - Defining basic hierarchies

        - Defining task hierarchies
        - Defining concept hierarchies
        - Defining reference hierarchies
        - Defining reading hierarchies

4. Creating relationship tables

The resulting map was used as a starting point for authoring tasks, as a definition for online navigation, and as a model of relationships I could derive links from, including parent/child links for all topics and next/previous links for topics in a sequence.

2.5 Identifying and organizing concepts

Based on the scenarios and tutorials, as well as the audience requirements, I developed a list of concepts. The concepts addressed not only support for specific tasks, but also support for DITA authoring in general, including descriptions of DITA roles and development processes, comparisons to other markup languages such as IBMIDDoc (our existing internal SGML language) and HTML, and answers to "why" questions such as "Why use maps" and "Why use topics".

For example:

- Introducing DITA
  - Topics and information types
  - Maps and relationships
  - DITA and IBMIDDoc
  - DITA and HTML
  - DITA and minimalism
- The information development process
  - Design phase
  - Development phase
  - Delivery phase
- Roles in the writing process
  - Type architect
  - Information architect
  - Information developer
  - Build developer
  - Information designer
- Topic-based information
  - Why use topics?
  - …
- Maps
  - Why use maps?
  - …
- Information types
  - Scenarios
  - Tutorials
  - Samples
  - Examples
  - Task topics
  - Concept topics
  - Reference topics
  - ….

Within the concept hierarchy, I tried to introduce new concepts in a spiral pattern, first with an overview (like "Maps and relationships" in the introduction) and then with more substantial treatment (an entire branch of concepts under "Maps").

As with the task hierarchy, I reviewed the concept hierarchy with developers, architects, users, and usability experts.

Again, the map was used as a starting point for authoring, and also for generating navigation and links.

2.6 Identifying and organizing reference information

The reference content was developed by another writer, using a set of transforms that extracted information from the DTDs and a specialized XML reference DTD to manage descriptions of elements and attributes. Maps were used to provide two separate organizations of the content (alphabetical and by semantic category), and the scenarios helped prioritize which elements would require the most focus and support. However, the scenarios did not tell us what reference subjects needed documentation: the actual structure of the subject (in this case a set of DTDs) defined what needed documentation. The language reference information went through its own review cycle, administered by its author.

2.7 Writing concepts and tasks

Because I wanted to validate my understanding of roles and development processes and test my justifications for using maps, topics, and DITA with their intended audience, I started by developing the concepts, and I sent them out for review even before the tasks were developed. I also had a number of existing concepts from a previous version of the DITA User's Guide (which was more concept-oriented and less task-oriented), that I wanted to reuse and revalidate as early as possible.

After the concepts were written, I developed a minimal set of tasks. Although normally tasks would be the focus of the information, since this was a first release of something very new to our users, I felt that tutorials and concepts needed a high priority, although not at the cost of task orientation.

2.8 Relating tasks, concepts, and reference information

After the concept, task, and reference content was defined, at least to the extent of having shell topic files that each had a title and short description, I mapped the information types together using a DITA relationship table.

For example:

Maps	Creating maps	<map> element

This row in the table asserts a relationship between a concept, task, and reference topic, which will result in links being added among the three.

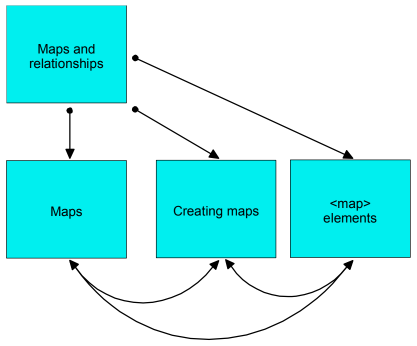
I combined topics in a single table cell when they shared the same relationship to other cells, and I used a linking attribute to customize how the links would be generated.

For example, I wanted the introductory maps topic ("Introducing maps") to have the same set of supporting links as the main map topic ("Maps"). I also wanted the introductory topic to link to the main topic. But I didn't want any other topics linking to the introductory one, since it was not the main topic on maps and was intended to be read only once as part of an introduction, not referred to repeatedly as a point of reference.

To accomplish this pattern (an introductory and main concept linking to each other, sharing the same pattern of links to other topics, but without anything linking to the introductory topic), I modified the row as follows:

[*Maps and relationships* Maps]	Creating maps	<map> element

The square brackets [ ] indicate that "Maps and relationships" (the introductory topic) and "Maps" are part of a family grouping, which means they link to each other. But the *italics* means that "Maps and relationships" is a "source-only" participant in its relationships in terms of linking. So "Maps and relationships" will get links to other topics in the relationship ("Maps" because it is in the same *family*, and "Creating maps" and "<map> element" because they are in the same row) but not vice versa. The other participants in the relationship behave normally but ignore "Maps and relationships" – so the task and reference topic will link to the main concept ("Maps") and to each other, but not to the introductory concept ("Maps and relationships").



A single row in the table expresses a fairly complex linking pattern, with four topics and nine hyperlinks, clearly and compactly.

2.9 A note on maps and linking

As you've seen, you can use maps to describe different kinds of relationships among your topics. Hierarchical relationships, like parent and child, can be drawn from the same map you use to generate navigation (like an Eclipse help or Microsoft HTML Help table of contents). You can supplement these with sequences and families within the hierarchy. And you can go beyond the hierarchy, using tables to organize your linking dependencies and creating additional families and sequences outside the hierarchy.

There are several advantages to maintaining links in your map, rather than in the topics:

- You can change links more efficiently. Editing groups, hierarchies, and tables is more efficient than managing all the individual links they imply.
- You can review consistency and usability in one place. You can see patterns and identify holes in your Web more easily looking at a map than you can by reviewing each topic individually.
- You can apply different links for different contexts. If you reuse information in another Web, where it needs different links, you can just apply a different map, without editing the topic content.
- You can continue editing links even after the topic content has shipped to translation. If you add or delete a set of topics at the last minute, you can incorporate links to them throughout your Web without editing the topic content.

2.10 Developing an integrated reading hierarchy

Although the concept, task, and reference hierarchies were suitable for producing online navigation, they were less suitable for an end-to-end reading flow through all the information, such as a user's guide would typically provide. After developing all the content and mappings, I created an additional map specifically geared to provide a readable sequence of concepts and tasks. I followed a few basic guidelines to assemble the map and was producing a publishable PDF within a day.

The guidelines were:

1. Create a DITA map file.
2. Copy in the task hierarchy.
3. Add concepts as children of high-level tasks, as siblings of tasks, or as parents of low-level tasks. Make sure the concept precedes the actual instructions to the user that the concept supports.
4. Adjust the nesting level to make sure nothing is nested more than three levels deep.

2.11 Indexing

Indexing was almost the last step. I could have added index terms to either the map or the individual topics to create either context-specific indexes (for help vs. PDF) or general-purpose indexes. In this case, because I was integrating the online tasks and concepts with the language reference, which had been indexed at the topic level, I followed suit and defined a single set of index entries for use both in help and in PDFs.

2.12 Producing output

I used the concept, task, and reference maps, along with tutorial and sample maps that organized the scenario background materials, to create the HTML Help deliverable that shipped with the internal DITA toolkit. The help file included all the information discussed above, including the language reference material developed by the other writer. Each page included a feedback link.

I used the reading hierarchy to produce a PDF available from the toolkit's documentation intranet site. This PDF did not include the language reference, which was made available in a separate PDF. The PDF included a feedback email address in the introduction.

I used an additional map that integrated all of the other maps to create a multiple-view navigation for use with an Eclipse help plug-in version of the complete information set. This additional step, including defining the integrating map and producing and testing output in the additional format, took half a day for approximately 500 printed pages.

3. INFORMATION ACCOUNTABILITY

There were several features of the process described here that helped to validate the information and keep its development on track:

- Scenarios, which informed the rest of the information architecture, were directly validated with users through the review and testing of tutorials and samples before any other information was developed.
- The information model based on the scenarios, specifically task and concept hierarchies, were reviewed before content was developed.
- Roles, which described expected usage patterns and responsibilities, were directly exposed to users as concepts and reviewed early in the information development cycle.
- The information output included pervasive feedback mechanisms.

The information was based on scenarios throughout, and was task-oriented from beginning to end. The same scenarios that shaped tutorials and samples at the start helped shape the separate help and PDF tables of contents at the end. When scenarios changed, the change could be tracked throughout the information, and the output could be quickly updated as the information model evolved.

The information went through many stages of review, from initial tutorials and samples through to the final help and PDF. Because content was developed in the order that users required it, each stage received a thorough review by users who actually needed the information, instead of a more artificial reading-oriented review by technically proficient but differently motivated developers.

4. PROCESS INTEGRATION

Looking at the stages in the process described, there are some points of integration that can help keep the content synchronized with the requirements expressed by the original scenarios. Other tools [9] have also explored the integration opportunities between tasks and scenarios, and have also explored automation possibilities. The DITA architecture to some degree may provide not only the basis for some simple integration points as shown here, but may also serve as an interchange point for any tool collecting information that would feed into this process at any point, allowing different tools targeting the same information lifecycle to work together.

In our own existing process, the following integration points are supported:

- **Roles and information model**. Within DITA maps, you can use an audience attribute to identify branches that apply to a particular audience or role. For example, the "Maps" branch of the concept hierarchy, and the "Creating maps" branch of the task hierarchy, applied specifically to the "information architect" role.
- **Roles and concepts**. You can create concepts for each of the roles to directly expose the role analysis to your users.
- **Scenarios and concepts**. Concepts explored in initial tutorials were migrated into the concept hierarchy and then reused in the tutorials. This meant that key concepts were reviewed as part of tutorial testing, and although also available in the concept hierarchy, they were maintained in only one place.
- **Information model and links**. As the information model evolved, it applied different links among topics on output. This meant that when task flows changed, new next and previous links were automatically applied; and when branches were split up or combined, the parent topics automatically received the appropriate child links.

In future, additional integration points can be considered, both within the process and across processes.

4.1 Integration within the process

Future integration points within the process could include:

- Automatic generation of task flows from scenarios, or automatic tracking of the effect of scenario changes on task flows.
- Mapping of audiences and roles to a hierarchy of role definitions, to allow easier analysis of roles across information sets.
- Integration of scenarios with tasks, to allow scenario-specific versions of tasks to be single-sourced with the generic versions in the task hierarchy.
- Integration of the information model with topic metadata, allowing an evolving information model to apply different metadata as needed for new situations without having to directly edit the content. This integration of model and metadata would also allow more flexibility in the reuse of topics across scenarios and across information models.
- Integration of the information model with topic titles and short descriptions to allow these text elements to be adjusted as needed for different contexts without direct editing of shared topics.

4.2 Integration across processes

Other processes could also benefit from integration. While some possibilities were discussed in [7], these opportunities are much more realizable within an XML framework that accommodates multiple dimensions of reuse, transformation, and specialization. In brief, these opportunities could include:

- **Integration with development use cases**. When created with information requirements in mind as well as software requirements, a single set of use cases could inform both software and its documentation and user assistance.
- **Integration with testcases**. Tutorials based on these scenarios could be automatically executed to create their accompanying samples, which simultaneously tests the tutorials, tests any concepts and tasks they include, and makes sure that the samples are current with the version of the toolkit being tested.
- **Integration with the user interface**. Translatable strings for user interface elements such as button names and window titles could be authored as DITA resources and then reused by DITA content as well as transformed to software resource files. This would mean that references to user interface labels are always accurate and up-to-date. Hover help could be single-sourced with the short descriptions of help topics, providing progressive disclosure from the user interface label to its hover help, to its F1 help, to complete reference documentation.
- **Integration with user feedback**. As discussed to some degree in [8], a flexible XML framework could allow the validated incorporation of user feedback through the form of self-directed FAQs or through more formal channels from a technical support database of problem reports and solutions.
- **Integration with education**. Educational materials could be sourced from the same scenarios and tutorials as the documentation and help, automatically abstracting tasks into exercises and concepts into bulleted summaries with speaker notes.

5. GOING FORWARD

DITA is an exercise in using technology to support best practices. It is also an opportunity to promote these best practices as part of the education and documentation efforts associated with the architecture's internal release. First we used best practices to define the technology we wanted. Now we are using the technology to help drive adoption of the best practices.

Going forward, there are even more opportunities presenting themselves, both for a more completely supportive process within the development of an information set, and across all the information processes associated with software development and education.

Ultimately, DITA can be a tool for capturing and driving best practices, not just within information development but across the entire information lifecycle.

The challenge will be to realize these opportunities while keeping the architecture flexible enough to allow evolution in the best practices it supports, and open enough to keep the architecture accountable to its users.

6. REFERENCES

[1] Priestley, M., Schell, D. Specialization in DITA: technology, process, and policy. ACM SIGDOC 2002 Conference Proceedings

[2] http://xml.coverpages.org/ni2003-04-25-a.html

[3] Priestley, M. DITA XML: a reuse by reference architecture for technical documentation. ACM SIGDOC 2001 Conference Proceedings

[4] Day, D., Hennum, E., Hunt, J., Priestley, M., Schell, D. An XML Architecture for Technical Documentation: The Darwin Information Typing Architecture. STC 2003 Conference Proceedings

[5] Rational Unified Process. http://www.rational.com/products/rup/index.jsp

[6] Rosson, M. B., Carroll, J. M. Usability Engineering: Scenario-based development of human-computer interaction. Academic Press 2002

[7] Priestley, M., Utt, M. H. A unified process for software and information development. ACM/IEEE IPCC/SIGDOC 2000 Conference Proceedings

[8] Priestley, M., Berglund, E. Open-Source Documentation: in search of user-driven, just-in-time documentation. ACM SIGDOC 2001 Conference Proceedings

[9] Paris, C., Vander Linden, K., Lu, S. Automated knowledge acquisition for instructional text generation. ACM SIGDOC 2002 Conference Proceedings

Trademarks