XML Tips and Techniques
in Adobe FrameMaker 7.2

XML Overview
Extensible Markup Language (XML) is riding a wave of publicity. Business people around
the world are trying to figure out how this new technology relates to important concepts
such as structured authoring, network publishing, and web services.

Structuring information with XML provides many benefits:
• Dramatic efficiency improvements
• Better customer service
• Faster cycle times for projects
• New possibilities for the exchange and presentation of information

While the migration to XML offers great benefits, this transformation can often require
significant modifications to existing workflows. Information workers may have to learn
additional skills, master new technology, and learn to collaborate in new ways.

Manage this change carefully. Take time to ensure that the new XML-based system leverages
the capabilities of XML in harmony with other systems, and provides continued ease of
use for those who author, manage, review, approve, and deliver content. Select appropriate
tools to ensure a smooth transition to the new, more powerful ways of working with and
presenting information that XML makes possible.

This whitepaper explores XML from several perspectives:
• If you are completely new to XML, continue on with this section, titled "XML Overview,"
  which provides a high-level primer on XML and an overview of the XML features in
  Adobe® FrameMaker®.
• If you already know the basics of XML, but need to know more about how it can be
  applied to solving publishing problems, check out the section titled "Single-Source
  Publishing with XML " on page 4.
• For a detailed review of XML from a technical perspective, read "XML in Detail" on
  page 9.
• "Working with XML in FrameMaker” on page 14 offers specific information on
  FrameMaker and its support for XML.
• If you want to learn about the ways you can implement a structured authoring solution
  built around FrameMaker and XML, "Implementing a FrameMaker XML Solution" on
  page 18 details the role of FrameMaker as the core of an enterprise-wide solution.
• To decide whether structured authoring will benefit your organization, the section titled
  "Next Steps" on page 24 will help you evaluate your situation and plan for the future.

Why XML?
XML is a metalanguage, or a language for creating languages. XML lets you define the
structure and syntax of your own markup languages, or document types, with which you
can manage your information.

XML provides the capability to structure information, letting you work with content
independently of the way it is presented. You can, for example, maintain a single XML
document and publish its content to many output formats: print, Adobe PDF, HTML, and
interactive XML applications, such as VoiceXML.

XML also enables customization and personalization in the course of delivery to such formats.
The benefits of structure
Organizations reap rewards from having their documents conform to a consistent, defined structure:

- The ability to reuse design templates
- The capacity to efficiently exchange information internally and externally
- The ability to work with document components effectively

XML provides methods for defining and enforcing a consistent structure used throughout the authoring process. For example, you can use an XML Document Type Definition (DTD) or an XML Schema to define your document structure; with authoring applications that support XML, you can author in an environment that guides conformance to this structure.

Separation of content from presentation
By separating a document's content from its formatting, the content can be rendered and delivered to multiple output devices by changing only the formatting rules for each device. XML provides a means of achieving this distinction.

XML allows content and presentation to be maintained independently.

Separating content from presentation allows different people and departments to work on different aspects of publishing projects independently, and provides maximum flexibility in planning for the future. Content that is structured and maintained as XML is ready to be styled for any future XML-based form of delivery.

Data and document exchange
Because XML lets you define your own definitions of document or data structure, it can serve as a powerful format for the exchange of information—between people or applications within a single organization, between distinct organizations, or between an organization and the public at large.

Industry-standard XML document types provide common data and document structures, giving organizations common languages for information exchange. Although XML is relatively new, there are already XML document types publicly available for almost every kind of information—from purchase orders to publications, to messages between computer programs.

XML features of Adobe FrameMaker
FrameMaker provides a powerful, scalable solution for authoring and publishing information:

- It offers a full-featured, structured authoring environment with support for XML.
- It lets you define the structure of your documents with XML DTDs or Schema and allows you to author with real-time validation of document structure.
- It provides formatting capability of XML for high-quality print and PDF output.
- It can interface with other systems that support XML, letting you take full advantage of continued advances in content management and publishing technology.
Integrated XML authoring and composition
FrameMaker 7.0 and later offer a single environment capable of both structured authoring and simultaneous "What You See Is What You Get" (WYSIWYG) editing of the same content. With FrameMaker, you can edit your document from the Structure View of the underlying XML, and simultaneously see and control precisely how your XML will look when printed.

For authors moving to XML authoring from an unstructured, word processing environment, migration to working with structured documents is made easier by the availability of the Document View feature in FrameMaker, representing the precise appearance of resulting PDF or print output. This intuitive mode of editing lets authors see how their content will appear in print, while being guided to edit in accordance with the structure of the associated XML DTD or Schema.

XML formatting capabilities
FrameMaker offers a robust document object model that lets you compose print-ready documents from source XML, for a wide range of document types—from simple invoices to complex technical documentation. FrameMaker provides the same powerful formatting capability to structured documents, with the addition of context-dependent formatting rules. FrameMaker styles are defined and applied in a user-friendly, visual environment that doesn't require knowledge of a complex coding mechanism.

XML mapping to common document objects
FrameMaker lets you format your XML documents by associating style with the context of document elements. The Element Definition Document (EDD) defines the structure of the document as specified by an XML DTD or Schema and maps of document structure to formatting.

With EDDs, you specify general formatting rules for each element in your document, as well as rules for changing this formatting when the element appears in specific contexts (for example, the same element may be styled one way whenever it occurs in an introductory section, but differently when it occurs in other sections). Such formatting rules map XML context to familiar document objects such as tables, graphics, and footnotes. This method of mapping presentation to structure in FrameMaker provides complete flexibility and formatting power without sacrificing ease of use.

Support for conversion of unstructured documents to XML
FrameMaker provides a means of importing and applying structure to unstructured content such as word processing files. If your word processing documents conform to a consistent use of style tags, you can automate the conversion process using conversion tables.

Conversion tables let you define a mapping between the paragraph and character styles of imported documents to the structured elements of FrameMaker. Once you have set up a conversion table, applying structure to a word processing document can be automated, allowing such content to be fully integrated into your XML workflow.
Support for XSL transformations

The Extensible Stylesheet Language Transformations (XSLT) is a W3C-standard language that provides the capability to process, manipulate, and transform XML documents. XSLT is often used to select, sort, or filter XML content, or to transform XML content to another markup language for publishing. FrameMaker can automatically apply an XSLT transformation when importing or exporting an XML document.

Support for XML-based knowledge management

Because FrameMaker supports XML, it can interface with and leverage an expanding range of XML-based tools and technologies. FrameMaker can format complete XML documents or import XML fragments into your documents from databases and XML data sources, exchange XML with document repositories, and create XML in addition to print, PDF, and HTML output.

Automated and server-side XML publishing

While the standard (desktop) version of FrameMaker offers complete desktop authoring and publishing capabilities, server-side publishing is possible with the FrameMaker Server product. The two products use the same publishing engine, allowing for a workflow in which templates and formatting rules created in the user-friendly desktop authoring environment can be applied directly to XML on the server. Such content can come from a wide variety of sources (including content authored in FrameMaker) and can be formatted for output in an automated, high-volume, high-throughput, server-based fashion. In this way, FrameMaker Server is a valuable tool in producing personalized print and PDF documents from content managed in XML.

Single-Source Publishing with XML

XML can be incorporated into enterprise workflows in many ways—from simple changes in document export formats to complete transformations of business processes. The move to single-source publishing often requires changing the way information is managed within an organization.

There are significant reasons why a company decides to adopt single-source publishing methods, even though such a migration may include an involved implementation process with changes to established workflows. Companies often adopt single-source publishing as a way to increase publishing efficiency:

• Standardize information formats
• Write once, publish everywhere
• Save time
• Reduce cost
• Achieve consistency
• Repurpose information
• Personalize content

Single-source publishing is indeed a big step; however, the productivity gains and additional benefits can more than pay for the cost of implementation. Benefits may include faster time to market, more effective cross-enterprise collaboration, and increased accuracy and availability of published materials.

The challenge: information across the enterprise

Success or failure in business is often determined by the effectiveness of information management. The challenges associated with information management get more intense as organizations grow in size (for example, a large organization typically houses its information across different departments, different divisions, and sometimes across many countries).

Multiple information sources

Content aggregated to produce a given publication may come from several departments, from a combination of internal and external sources, or both. Content from different sources may also be updated at different frequencies. For example, financial documents often include relatively static text merged with dynamically-generated numbers. Organizations face the challenge of effectively working with such an array of information sources across the enterprise or in collaboration with clients, vendors, and partners.
Diverse document and data formats
Information that is aggregated into published content may come from a wide range of document and data formats. As different departments or companies may use different applications to create content, a diverse assortment of file types must often be aggregated to produce a single document. Information sources may also differ in structure. Great disparities may exist between the way two documents of the same format are organized. Merging or exchanging content between data or documents of different structures can be a daunting task.

Demand for multichannel publishing
Beyond the potential challenges with multiple origin and formats of source content, there is an ever-expanding range of ways that content can be published. There are a growing number of forms of content delivery—from new Print-On-Demand technologies, to increasing capabilities of websites, to new XML-based publishing formats such as Wireless Markup Language (WML) for phones and VoiceXML for voice browsers.

Organizations face an increased demand for multichannel publishing.

In addition to the demand for content delivery to an expanding range of formats and media, there is also increased need for such delivery to take place as quickly as possible, often in a customized or personalized manner. Since the advent of the World Wide Web, consumer demand for real-time, custom information has steadily increased. While the efficiency that can be gained from such targeted communications is desirable, customization and personalization introduce another layer of complexity into the publishing process.

The single-source solution
By implementing a single-source publishing solution, organizations can effectively face the increasing demands on content development, management, and distribution. Rewards of effective single-source implementation can include productivity gains that translate directly to the bottom line.

Standardize information formats
With single-source publishing, information is housed in a consistent structure across common document and data types. Using a defined, agreed-to structure to manage content makes working with multiple information sources far easier.

Such standardization can benefit information exchange from both within the organization and between the organization and the outside world. Within the organization, consistent document structure facilitates communication and collaboration. Information exchange between companies can also benefit from the use of a common standard for document structure, such as that defined by industry-standard XML DTDs or Schema.
Write once, publish everywhere
Standardized document structure can provide powerful capabilities for publishing. By maintaining content with a consistent structure, formatting and processing tools used to present that content know what to expect. Because styles are associated with a document structure, rather than with a specific document instance, content can be kept distinct from presentation. This can allow a template-driven workflow in which the same template or stylesheet can be reused for multiple documents (as long as they are of the same type).

Save time
When information is well structured, it is easier to exchange, publish, and access. By adopting the methods of single-source information management, organizations can spend less time in their publishing efforts, as content with predictable structure can be deployed to multiple outputs using automated processes. Less time can also be spent in accessing information, as structured content lends itself to indexing, searching and navigation. By housing content used in multiple output streams in a single source document and automating publication, there is no need to reconcile differences between multiple versions of the same document.

Reduce cost
In addition to saving time, single-source publishing can save money. With the increased level of automation made possible by structured content, reduced resources can accomplish similar or better results in content distribution than with former unstructured methods. By maintaining content in a single source document, resources formerly devoted to managing multiple versions of the same document can be reallocated.

Even though cost reduction may not be apparent at the beginning of a single-source publishing project, the benefits of single-source publishing are attained through significant work in defining document architecture at the outset. It is this proactive setup that makes subsequent work in content creation, management, and delivery more efficient.

Resource Units

![Resource Units Graph](image)

Single-source publishing can dramatically decrease the long-term cost of documentation projects.

Resource allocation for single-source structured authoring is different from that of unstructured authoring. While a structured authoring project typically starts with substantial effort devoted to setting up document and system architectures, unstructured authoring projects do not necessarily require such extensive startup costs. However, the benefits of a structured system become apparent over time.

By the content delivery phase, the differences can be stark: With unstructured content, this phase can be a mad scramble to reconcile inconsistencies between multiple versions of the same content; with structured content, any last-minute changes to content can radiate to all forms of published output.
Achieve consistency
In addition to time and cost savings, working in a single-source environment generally provides far greater consistency between documents than is offered by an unstructured environment. When documents conform to a consistent structure, document processing can be more efficient. With unstructured documents, document processing functions have to be ready to adapt to a wider range of possibilities, so automation is more difficult. Consistency can also aid in maintaining the same corporate image across publications, and can increase the accessibility and reusability of published information.

Repurpose information
The structured nature of single-source content can facilitate repurposing of the same core content to multiple delivery formats, and enable customization for delivery to different audiences. Structured formats such as XML can be processed by applications that filter, transform, and style the same content to produce multiple versions of a document from a single source (for example, a document may have one version for internal distribution and another for external publication). With a clearly defined structure, sections can be included or excluded as needed, formatted in different ways, or merged with external data based on business rules defining the differences between the two versions. Such repurposing is possible because processing applications can rely on consistent document structure and, in some cases, metadata (such as XML attributes).

Personalize content
Single-source publishing can also make personalization of content delivery a reality. As the capabilities of electronic information delivery expand, the many ways that data and documents can be personalized increase. Personalization can dramatically improve the efficiency of content delivery, as the end users of personalized content receive only the specific information they need.

Personalization requires the interaction of two information sources: the content being published and a description of how that content should be personalized for a given user. The description of personalization needs can come from direct selection by the end user (for example, selection of options on a website) or from data about the user that has already been collected (for example, user profiles). Applications delivering personalized content generate documents reflecting a mapping between these two information sources according to specific business rules.

Structured source content expands the possibilities for personalized content delivery, as it allows the systems that manage personalization to reliably identify document components for inclusion, exclusion, or modification. With structured content, a fine granularity of personalization can be maintained (for example, changing the formatting of a single word based on a user profile). On the other hand, it is more common with unstructured documents for personalization to occur on the page or at the document level (for example, specifying entire sections of documents for inclusion or exclusion); without structure, there is often no reliable way to access document components in a fine-grained manner.

Implementation of a single-source solution
Migration to structured authoring takes planning, dedicated work and, at times, changes to workflow. Once such a transformation is complete, however, the benefits of single-source information management can be substantial. Payoffs such as reduced time to market, increased efficiency, and ultimate cost savings and competitive advantage can be well worth the effort of implementing a single-source solution.

Working in a structured authoring environment
Although such a migration can have impact across the enterprise, the authors who initially create and maintain content may be the most impacted by the change to single-source information management. Authors have to adapt to the demands of a structured environment (including mastering new tools and collaboration methods) and to the increased responsibility of maintaining a single source document from which all published materials flow.

All documents have some degree of structure; even authors with a background exclusively in unstructured authoring have at least some familiarity with document structure. There are usually defined standards for document types within an organization even when these are only guidelines rather than rigorous definitions. New challenges arise for authors because structured authoring environments provide more explicit rules for document structure. Structured authoring tools will often enforce these rules in real time.
XML Tips and Techniques in Adobe FrameMaker 7.2

With word processing programs, it is possible to bend the rules and introduce inconsistencies of document structure. XML authoring applications, on the other hand, define rules of structure with an XML DTD and will typically either prevent or warn of violations of those rules, making it impossible to make changes that would break document structure. Authors new to XML may need time to adjust to this stricter environment.

While all structured authoring systems maintain content independently of presentation formatting, different systems offer different methods of applying formats to the core content and different degrees of editing within a “formatted” context. Transition to structured authoring can be easier or more difficult, depending on how directly the appearance of a document in the authoring application maps to the way it will look when formatted for print. This is especially true for users moving from a WYSIWYG environment such as a word processing application.

Addressing the challenge of multiple systems
In single-source implementations, there is often a requirement to aggregate data or document content from a variety of systems and data sources. By maintaining core document content in XML, such content can, in many cases, be updated easily from other XML sources. XML authoring systems generally facilitate the import and export of XML fragments, so those systems and data sources capable of exchanging XML can often be better integrated in a structured authoring system.

While there are integration advantages associated with basic support for XML, publishing needs may also exist for importing content from unstructured documents (for example, word processing or spreadsheet files), delivering documents in non-XML formats, and referencing non-XML document objects from the core XML content. Binary graphic file formats, such as EPS or TIFF, are typically referenced in XML documents and must often be processed by publishing systems.

In selecting an XML authoring application, it is important to consider its capability for authoring and processing XML as well as its ability to work with such formats. There are often needs, for example, to translate from one graphic format to another in the course of publishing to multiple formats: A graphic object maintained as an EPS file referenced in the source XML document may be translated to Portable Network Graphics (PNG) for delivery to a web page in one output stream, while being integrated into PostScript® output in another.

XML authoring applications have a range of support for integration with non-XML formats—from limited XML editors that only process XML, to systems that integrate XML authoring with advanced document import, graphic processing, and page layout capabilities. Depending on the particular tools selected, single-source solutions can have a varying number of components.

Staffing for XML systems
Depending on the chosen application architecture and tools, different degrees of expertise will be required by those playing the various roles in the XML publishing process. Single-source solutions may introduce a greater degree of specialization within authoring groups, such as a defined distinction between authors, document architects, and template designers. Depending on the backgrounds of existing staff, there is often a need for training; in some cases, hiring consultants or staff for some of the more highly specialized functions may be desirable.

Managing workflow for reuse
Structured authoring enables new forms of collaboration and document management. While providing many benefits, this may also present adaptation challenges. With XML, it is possible to reliably identify document components, which makes it easier to work independently with document sections or components.

There are an increasing number of ways XML information can be stored, such as XML databases and repositories. With some document management applications, users can check in or check out XML documents or document fragments, and the system provides features such as version control and document distribution in a review and approval process.

When content needs to be distributed to multiple channels, it can be helpful in the authoring process to plan ahead in anticipation of the forms in which the content will be published. Certain output formats may have unique requirements. For example, a single source document may be published to PDF, HTML, and Braille or VoiceXML (through transformation of the XML output). In such cases, authors can use XML metadata to provide information that aids in specific presentation methods (for example, spelling out acronyms for use in Braille presentation and indication of voice styles for VoiceXML delivery).
Planning for change
By implementing an XML-based, single-source information management system, organizations are taking a prudent step in preparing for the future. While XML document types continue to proliferate, those documents that have been maintained in XML are usually ready for transformation to new document types, as it is generally possible to map one XML structure to another and process transformation between structures with a method such as Extensible Stylesheet Language Transformations (XSLT). FrameMaker can automatically apply XSL transformations when importing or exporting XML.

XML in Detail
XML is at the core of most single-source publishing solutions currently implemented. While XML was created in response to the needs of the publishing community, it has since become associated with information in a more general sense. Beyond its expanding role in publishing-related applications, it is also being used to manage data in forms unrelated to publishing (for example, by web services as a format for low-level XML messages between applications across the web). Learning the fundamentals of XML can lead to an understanding of its ever-expanding applications.

Some authoring systems provide user-friendly interfaces that make it easy to author XML without learning the details of XML syntax or mastering XML hand coding. In such cases, a knowledge of XML basics can still be beneficial—to learn behind-the-scenes activities and to leverage the power of the various forms of available processing once you author your content with an XML-supported application.

The basics of XML
XML 1.0 was approved as a standard by the World Wide Web Consortium (W3C) in 1998. XML is a subset of Standard Generalized Markup Language (SGML), which was invented by Charles Goldfarb and his associates more than 20 years earlier and which became an International Organization for Standardization (ISO) standard in 1986.

The creators of XML were SGML experts who had recently seen the widespread acceptance of HTML with the advent of the World Wide Web. While HTML afforded a simple means of defining web pages, it was very limited and provided only a single set of specific tags with which to mark up content. SGML, on the other hand, is more complex, but lets users define their own tags and structural constraints on documents. The creators of XML strived to define a language reflecting the best of both worlds—combining the power of SGML for defining arbitrary document types with as much of the simplicity of HTML as possible.

When XML became a W3C standard in 1998, it was referred to as "SGML for the web"—a lighter version of SGML, which was easier to understand and implement. The specification for XML is defined at www.w3.org/TR/REC-xml.

XML provides rules for marking up documents and defining their structure. Consider the following simple XML document:

```xml
<?xml version="1.0"?>
<!DOCTYPE message SYSTEM "message.dtd">
<message importance="high">
  <to>Jane Doe</to>
  <from>Cynthia Smith</from>
  <body>Don't forget to bring the year-end report to our meeting tomorrow! See you then...
  </body>
</message>
```

The above example starts with the XML declaration (that is, it tells the application that will process the document that it is an XML file) and the version of XML used. The next line is the Document Type Declaration, which indicates that this XML file conforms to the structure defined in a specified external DTD.
Elements and attributes
The most commonly used XML objects are elements. Elements are typically delimited by start tags and end tags, and consist of these tags as well as the content that they enclose. All elements have an element name, which is the name in the element's tag. For example, the line:

<to>Jane Doe</to>

represents a single element of name to with start tag <to> and end tag </to>. Elements can also be empty. If this element were empty, it could be represented as <to></to> or use the shortcut nomenclature <to/>.

Elements can contain additional information about their content by means of attributes, or name-value pairs that occur within an element's start tag. For example, the line:

<message importance="high">

represents the start tag of the message element, which has an importance attribute with a value of "high." A given element may or may not be given attributes.

Attributes are critical to XML as they represent the primary means of including metadata, or information about element content (beyond the tag name). Often, such metadata is not presented directly when the XML document is published, but rather serves either to stylize the presentation or provide control over how processing of the document takes place. In the example above, the fact that the “importance” of the message is “high” might cause the message to appear in a different font when published or might trigger the system processing the message to expedite delivery.

While elements and attributes represent the most basic components of XML, there are other sorts of objects that can occur in an XML file, including entity references, comments, processing instructions, CDATA sections, and DTDs.

Entity references provide a means of including a special character, binary object, or section of text that is repeated in a document: Once you declare an entity, you can include its contents with an entity reference. Entities have many uses:

- Text replacement: Entities can serve as typing shortcuts or macros for text that is repeated often.
- File modularization: Entities provide an include mechanism to allow construction of a single document from multiple physical files.
- Graphics: Entities allow inclusion of binary objects such as graphics.

In a broad sense, entities can be categorized as either internal entities (that is, they are defined completely in the document such as an entity for a special character or acronym) or external entities (that is, they are defined by reference to an external object such as an XML file). While entities allow for a flexible and modular document architecture, extensive use of external entities can result in a large number of files for a single XML project.

Processing instructions (such as the XML declaration above) are used to provide instruction to the application processing the XML. CDATA sections provide a means of containing stretches of text, which may include markup characters, while allowing such text to be passed through, rather than interpreted as markup. DTDs are typically kept in files external to the XML, but may alternatively be included inside the XML file itself.

XML’s tree structure
While data can be structured in different ways, XML documents are structured hierarchically in a tree structure. Elements may contain other elements but, for a particular document, there must be only one document element that contains all others (for example, the message element in the complete example above is the document element for that XML document).

The relationship of elements in an XML document is often illustrated in diagrams as an inverted tree with the “root” at the top and the “branches” spanning downward. Elements represent nodes in such a tree structure.
XML documents are often interpreted in terms of a hierarchical, tree-like structure.

**Rules of well-formed XML**

Certain constraints form the criteria for a document to be considered an XML document. Not all XML documents need to conform to a DTD or Schema, but all XML documents must, at a minimum, conform to what are known as well-formedness constraints in order to be processed by an XML processor:

- The XML document must contain one or more elements: An XML document must have a single document element (or root) that contains all others.
- All elements must be properly nested: If the start tag is in the content of another element, the end tag must also be in the content of the same element.
- XML must be written in a case-sensitive manner: `<message></message>` is acceptable; `<message></Message>` is not.
- Every start tag must have an end tag: An element that starts with `<to>` must end with a closing `</to>`.
- Attributes must be quoted: `<message importance="high">` is acceptable; `<message importance=high>` is not.

XML authoring tools enforce well-formedness constraints in different ways—from providing warnings of errors to automatically making XML well formed behind the scenes in ways transparent to the author.

**Unicode**

Computers can represent individual characters of XML text internally in different ways. This is usually transparent to end users who see the same text on-screen regardless of how it is represented. However, behind-the-scenes computers map byte codes to character codes. A set of such mappings (for example, a list of the numerical codes associated with the letters on a keyboard) is known as a character set.

Early attempts to standardize such character sets include American Standard Code for Information Interchange (ASCII). However, such early standards allowed for only a small number of possible options (128 with ASCII and 256 with extended ASCII) for characters—enough to handle the English language, but not enough to handle languages such as Greek or Japanese.

The Unicode standard aims to provide a unique mapping for every character in every human language. It began with two bytes per character, raising the number of possible characters to 65,536. This number eventually became insufficient, so Unicode now supports methods of encoding a virtually unlimited number of characters. XML supports the ISO/International Electrotechnical Commission (IEC) 10646 standard, which can be considered an implementation of Unicode. (The two specifications are maintained with complete synchronization.)
XML document structure

Central to XML is its notion of a precise, well-defined document structure. Such structure can be defined by several forms of document schema. The primary form of schema for document-centric applications of XML is the DTD.

Learning to work with DTDs is important for structured authoring, although there is often a distinction in structured authoring workflows between the roles of document architects (who create DTDs) and authors (who work with documents that conform to DTDs). While document architects must have a detailed understanding of DTDs, including the ability to write them, authors can often leverage tools that guide conformance to a DTD; in such cases, it is not always a requirement for authors to master the nuances of DTD syntax.

DTDs

DTDs provide a set of rules indicating what elements and entities are allowed in the XML document, how often each element may occur, and what contents and attributes each element may contain.

Consider the following DTD:

```xml
<!ELEMENT message (to, from, body)>  
<!ATTLIST message importance CDATA #REQUIRED>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
```

This DTD indicates that the `message` element contains a required `importance` attribute, as well as `to`, `from` and `body` elements. By including such a DTD in the body of the XML document itself, or by referencing such a DTD in an external file, an XML file can be validated against the DTD.

Occurrence indicators can be used to define how many times elements are allowed or are required to occur within another element. There are four types of occurrence indicators:

- (no indicator) an element must occur exactly once
- * an element can occur zero or more times
- ? an element can occur once or not at all
- + an element can occur one or more times

For example, consider the following element declaration:

```xml
<!ELEMENT fish (halibut, carp*, goldfish?, trout+)>
```

In this example, the `fish` element must contain exactly one `halibut` element, zero or more `carp` elements, zero or one `goldfish` element, and one or more `trout` elements.

Beyond such basic definitions of structure, DTDs provide additional capabilities:

- Define different sorts of entities.
- Specify default values of attributes.
- Define special types of attributes such as identifiers whose values must be unique.

XML DTDs provide a simple and powerful means of defining document architecture.

Namespaces

If you work with XML long enough, you will eventually need to work with XML from multiple documents or data sources that don't conform to the same structure.

XML uses namespaces to differentiate between XML of different document types. A namespace qualifies element names with a namespace prefix. For example, a `message` element in the `ns` namespace would have the start tag `<ns:message>` and the end tag `</ns:message>`.

In this way, namespaces prevent names from "colliding." For example, if you have a table element in both a furniture catalog and an XHTML file, you could use the qualified elements `furniture:table` and `xhtml:table` in your document, so you would know from which DTD or Schema each arose.

Namespaces are defined by the W3C Namespaces specification at [www3.org/TR/REC-xmll-names](http://www3.org/TR/REC-xmll-names).
XML Schema

DTDs are defined in the original W3C XML Recommendation of February 1998. However, DTDs suffer from several shortcomings:

- DTDs are written in a different language than XML and therefore require a different parsing method in order to be processed.
- DTDs do not provide a method for defining strong data types for document objects.
- DTDs do not offer an easy mechanism for working with multiple namespaces.

In May 2001, the W3C approved the XML Schema Recommendation, which provides an alternative to XML DTDs for defining the structure of XML documents. For more information about the XML Schema language, see www.w3.org/XML/Schem.

XML and Graphics

Most graphics are expressed as binary file formats, and must be referenced in an XML document as external unparsed entities. However, the W3C has developed a specification for Scalable Vector Graphics (SVG), which expresses two-dimensional graphics as an XML document. Because SVG is a vector-based graphics format, SVG graphics retain their original resolution at any zoom factor. Because SVG is an XML-based format, it provides the capability for supporting data-driven and interactive graphics applications.

XML processing

Because of its conformance to strict markup rules and well-defined structure, XML can be easily processed by computer programs. There is an expanding range of processing methods for XML spanning virtually every platform and computing language. Many forms of processing allow for XML input to be transformed to XML output; therefore, it is possible to combine multiple forms of XML processing. The availability of such a vast array of XML processing tools provides another reason for organizations to integrate XML into their publishing workflow.

XSLT

XSLT is a transformation language for processing XML. XSLT processors take in source XML documents and XSLT stylesheets, and generate output files, which can be almost any structured, text-based format including HTML, XML and Maker Interchange Format (MIF). The XSLT stylesheets used in this process are themselves XML documents.

XSLT processing can occur in one of three contexts:

- As a batch process
- On a web server
- In a web browser

When XSLT processing occurs as a batch process, an XSLT processor takes in source XML files and corresponding XSLT stylesheets, and generates files to a specified file folder, which represents the results of XSLT transformation. When XSLT processing occurs on a web server, the process also involves source XML and XSLT stylesheets; however, rather than writing resulting files to disk, the results of the transformation are typically streamed to the user's web browser. Transformation in the web browser itself is possible for those browsers that support XSLT processing. In this case, an XML file and an XSLT stylesheet are sent to the browser, which presents the results of the transformation directly to the user.

The tree nature of XML comes into play with XSLT processing. XSLT works with a source tree (the XML source) to produce a result tree (hierarchically structured output such as an XML or HTML file) by providing rules for how the different nodes of the source tree are processed. Such rules for transformation are known as template rules and an XSLT stylesheet generally contains many such rules.
XSLT provides a powerful transformation language. You can use multiple XSLT stylesheets to publish the same source XML to different forms of output, such as HTML for the web, WML for cell phones, or VoiceXML for interactive voice applications. You can also use XSLT to transform XML to XML of a different structure, conforming to a different DTD. Such XML-to-XML translation can be useful when XML, which conforms to a proprietary DTD, is maintained in house, but XML exchange with external organizations must conform to an industry-standard DTD. In such a situation, XSLT stylesheets can be written to translate between the two structures.

XSLT is defined by the XSLT specification at www.w3.org/TR/xslt.

XSLT processors often play a critical role in XML publishing systems.

**Working with XML in FrameMaker**
FrameMaker combines sophisticated XML authoring capabilities with the power of multichannel publishing. While FrameMaker can serve as the structured authoring tool and core publishing engine for large organizations, it can also function as a standalone application for an individual user on a single desktop, providing a full feature set without any required add-ons.

FrameMaker includes many features that make it a natural choice for collaborative authoring environments:

- The capability to edit in parallel WYSIWYG and Structure View modes, with real-time validation and context-aware guides to facilitate conformance to an XML DTD or Schema.

- An intuitive, user-friendly interface for defining style mappings between source XML and formatted print and PDF output.

- “Out-of-the-box” printing and PDF generation for all users (that is, there is no need to access a server-side system for basic printing needs).
• Scalable server-side capabilities of the FrameMaker Server product for batch or server-side
generation of formatted print, PDF and other forms of output.

• A template-based model of formatting and styling content for delivery, consistent with the single-
source principle of keeping content independent from presentation.

• Support for scalable vector graphics (SVG), a vector-based graphics format specified as an XML
document. FrameMaker retains the original vector-based quality of SVG in print and PDF, and
can rasterize the SVG or write the original SVG code when saving as HTML or XML.

• Intuitive interface for creating and managing internal and external cross-references and exporting
those cross-references as XML.

• Support for expressing FrameMaker conditional text settings in XML documents and for retaining
those settings on import of XML to FrameMaker.

• Support for automatically applying XSL transformations on import or export of XML, for sorting,
filtering, or manipulating XML content.

Working with XML in FrameMaker includes authoring in conformance to an XML DTD or Schema,
formatting XML for print formats, and delivering content to a range of media and formats.
FrameMaker makes the XML publishing process friendly and accessible to information workers
without requiring extensive XML coding or detailed knowledge of XML syntax.

Authoring XML
Because FrameMaker combines structured authoring with robust formatting and page layout capa-
bility, authors who work in FrameMaker can work with XML in a WYSIWYG environment similar
to word processing. FrameMaker combines this with other views of the content and additional
features that guide conformance to a defined XML structure. With this guided editing feature,
Structure View and continuous real-time validation against an XML DTD or Schema, the intuitive
model of the printed page serves as a visual aid for working with highly structured content.

Guided editing
FrameMaker guides conformance to structure defined by an XML DTD or Schema. At any given
point in the XML authoring process, the Element Catalog lists those elements that are valid in the
current document context. It also provides additional indications about the current context, such as
whether an element could be added at the current location but would render following elements as
invalid, or whether text is allowed at the current insertion point.

FrameMaker also provides the option of showing or hiding element boundaries with tags that dis-
play the XML element names themselves, or with brackets that show the start and end of documents
and provide a quick indication of the current level of nesting. Such elements are integrated into the
WYSIWYG-formatted view of the document.

Structure View in FrameMaker
The Structure View provided by FrameMaker enables navigation of the document through an
expandable/collapsible hierarchical tree view of the document structure. The behavior of the
Structure View mirrors that of the document window (that is, navigation or manipulation of objects
in one view is simultaneously reflected in the other). For example, when you select an element in
the Structure View, the corresponding element in the document window are selected; edits to the
document in one view are instantly reflected in the other.
The Structure View in FrameMaker offers a powerful means of navigating and manipulating XML.

While the Structure View provides powerful navigation features, it also provides both special forms of editing and assistance in maintaining proper structure in conformance to the XML DTD or Schema. In the Structure View, you can edit by selecting elements, and dragging and dropping them to new locations. In the course of such an edit, visual cues indicate whether prospective destination locations for the selected element would keep or break valid document structure.

The Structure View provides similar visual indications of any violations of the defined document structure:

- Missing elements
- Elements at an invalid location
- Undefined elements
- Invalid attributes
- Attributes missing a required value

**Continuous real-time validation**

FrameMaker lets you know immediately when structure is broken. Using tools such as the Element Catalog, you can easily see what editing options would maintain validity in the current document context. FrameMaker also provides methods with which to validate an entire document, or a specific flow or element within a document, facilitating an interactive process of fixing a document by resolving validation errors one by one.

FrameMaker extends complete validation functionality with real-world flexibility, allowing users to save "broken" documents (that is, documents no longer conforming to the structure defined by the DTD or Schema) and make corrections at a later time.

**Generating print and Adobe PDF output**

While the number of possible forms of content delivery continues to increase, delivery to print and PDF remains a core publishing requirement for most organizations. FrameMaker provides robust composition features that let you precisely format XML content for delivery to print and PDF.

**Using the advanced composition features of FrameMaker**

FrameMaker provides a fully featured, industrial-strength page layout and formatting engine that has been proven across a vast range of document types and publishing environments. From high-level document objects (such as books containing multiple documents) to fine-grained objects (for example, character formats), the FrameMaker document object model encompasses the entire range of formatting objects required for document composition. These powerful capabilities can be entirely controlled through an intuitive, user-friendly interface.
FrameMaker offers many page layout and formatting features:

- Multiple layouts within a single document through the definition and application of master pages
- Complete text formatting using paragraph and character styles, which persist in format catalogs for easy reuse
- Strong table-handling capability, which enables tables to span many pages and include complex structure with detailed formatting
- System and user-defined variables, which allow for automation of features such as running headers and footers, page numbering, and page counts
- Conditional text features, allowing for inclusion or exclusion of document sections based on conditions
- Manually or automatically defined markers, which can be used for indexing, cross-referencing, and hyperlinks
- Autonumbering features of individual document objects such as paragraphs (for example, numbered lists and lists starting with bullets or other special characters), headers, page numbers (including static section prefixes or suffixes), and figures
- Powerful book-handling features, allowing for aggregation of multiple documents into a single book, which can be printed as a single document (that is, files such as tables of contents [TOCs] and indexes can be automatically generated to treat the documents contained in a book as a single entity)

Layout and formatting can be mapped to XML document structure in a context-aware fashion. Elements can be associated with master pages, paragraph styles, or character styles, and with different associations dependent on the structural context in which elements occur. A `<para>` element, for example, may occur in one font when contained within an `<intro>` element and in another font when contained within a `<sect1>` element. Formatting can be conditional based on the values of XML attributes.

FrameMaker provides tight integration with PDF, allowing for automatically hyperlinked TOCs and indexes, automatically generated bookmarks, and intradocument hyperlinks. FrameMaker can also automatically generate tagged PDF, which can increase the accessibility and reusability of PDF output.

**Tailoring content with FrameMaker templates**

FrameMaker templates provide a means of maintaining the styles used for print and PDF delivery independent of XML content. Templates are documents that store a complete range of properties:

- Master pages that determine the number of columns, placement of text flows, and background objects such as running headers and footers
- Reference pages that can be used to house reusable document objects and formats
- Element definitions and formatting specifications for elements based on structural context
- Table, paragraph, and character styles that can be referenced by the formatting specifications of element definitions
- Various document-wide settings such as footnote properties and custom marker types
- Color catalogs

Templates are used in the authoring process as a starting point for new documents, and as a means of maintaining multiple styles that can be applied to a single XML source as needed. Templates can allow, for example, delivery of multiple versions of a single document including a large-print version accessible to visually impaired readers.

Templates can include EDDs. An EDD contains both structural rules for the document, which can be directly based on an XML DTD or Schema, and definitions of the mapping of such structure to the formats and styles provided by the rest of the template. The power of templates and their embedded EDDs facilitate the WYSIWYG editing capability provided by FrameMaker.
In the publishing process, templates are used to stylize XML for content delivery. Because they provide a complete definition of formatting and style, and precise mapping of this presentation layer with defined XML structure, templates can serve as part of a powerful template-driven workflow in which XML is delivered as composed print and PDF output in a completely automated fashion.

FrameMaker templates allow delivery of XML content in multiple styles tailored to specific publishing needs.

Publishing to multiple output media
Beyond its advanced formatting and delivery capabilities for print and PDF, FrameMaker offers sophisticated means of publishing to HTML and XML-based output formats. XML content from a single FrameMaker source can be delivered to a wide audience in a diverse spectrum of media and formats.

Delivering XML and HTML
The full range of XML processing capabilities can be applied to XML created and maintained in FrameMaker. FrameMaker also includes WebWorks® Publisher Standard Edition from Quadralay. WebWorks Publisher serves as a powerful extension to the HTML and XML publishing capabilities that FrameMaker provides.

Using CSS generated from FrameMaker
The styles defined in FrameMaker templates can be automatically exported to Cascading Style Sheets (CSS). This time-saving feature provides a starting point for development of stylesheets that can be used either to directly style XML, or serve as a stylesheet referenced by HTML or the result of an XSLT transformation. The CSS exported from FrameMaker conforms to the CSS2 specification defined by the W3C, which is available at www.w3.org/TR/REC-CSS2/.

Implementing a FrameMaker XML Solution
Implementation of a structured authoring solution built around FrameMaker includes setting up XML Applications that define structure and formatting, integrating with other systems as needed to facilitate collaborative authoring—including review and approval of content—and automating the delivery of content to multiple channels with the powerful publishing capabilities afforded by FrameMaker Server.

Setting up XML Applications
A FrameMaker XML Application enables users to work with the multiple files required to structure XML (for example, DTDs or Schema) and to define mappings between XML and formatting and document objects (that is, EDDs and Read/Write rules). An XML Application can also associate XSL or CSS stylesheets with sets of XML documents. A structured application specifies the collection of files that work together to ensure ease of use.
XML Applications manage files and properties to facilitate easy authoring of XML.

Not every user needs to set up an XML Application; creation and maintenance responsibility is usually shared between document architects and template designers. Authors maintaining XML content with FrameMaker simply load the source XML document, and the XML Application automatically loads the correct auxiliary files.

With a properly configured XML Application, authors can benefit from the intuitive, parallel Structure View and the WYSIWYG view of the document, effectively maintaining XML content without extensive coding.

**Defining an XML Application**

XML Applications are defined in a special FrameMaker file named "structapps.fm." In this file, which is edited in FrameMaker, you can indicate all of the files that are associated with your XML project (such as the DTD, FrameMaker template, and the Read/Write rules file) as well as parameters required by an XML Application (such as the DOCTYPE and associated XSLT or CSS stylesheets). A single structapps.fm file can contain definitions of any number of structured applications.

XML Applications are themselves structured documents and can be edited with the same methods that FrameMaker provides for general XML authoring.

**Using an Example XML Application**

FrameMaker provides example XML Applications for some of the most popular industry-standard information models — XHTML (an XML formulation of HTML), DocBook, and DITA. DocBook is an information model that is popular with publishers of technical books. DITA, the Darwin Information Typing Architecture, is a relatively new information model that supports topic-oriented authoring and reusable content chunks. If you are working with any of these information models, you can use the appropriate example application as is, or customize it for your organization's needs.

**Defining a document structure with a DTD or Schema**

The selection of a DTD or Schema for critical documents can be a major decision for organizations working with XML. There are often proven, industry-standard DTDs from which to choose; however, an organization may have custom needs that are not met by a particular DTD in its “standard” form. Organizations often have to choose between creating a new DTD and modifying an existing one.

As FrameMaker can work with standard XML DTDs or Schema, it is well suited to either approach. The process of creating a DTD is the same for FrameMaker as it is with general XML DTDs, though FrameMaker provides special features to let you associate DTDs with the other files used in XML authoring. FrameMaker lets you reference DTDs in the definition of the XML Application and provides a means of automatically generating an EDD (used to map formatting to XML context) based on a DTD.

XML DTDs can be very modular: Entities can be used to create a DTD out of distinct files, which may facilitate independent customization of different structural characteristics. In such cases, the XML Application provided by FrameMaker keeps track of entity locations, allowing integration of such modules into a single definition of document structure.
Designing an EDD

Element definitions are defined by EDDs. The EDD contains both structural rules for the document and styling rules, which dictate how elements of a specific type are styled. EDDs can leverage reusable styles and document objects created within the user-friendly graphical interface provided by FrameMaker.

You can create an EDD from a DTD, and the automatically generated EDD will provide the basic structural definition of the document. Given this structure, you can define the formatting associated with each element, including default formatting and special formatting rules that are applied when the element occurs in a specific context or is based on certain conditions such as attribute values. An EDD is typically embedded into the FrameMaker template referenced by the XML Application. You can also update an EDD automatically to reflect changes made to a corresponding DTD.

Like XML Applications, EDDs are themselves structured FrameMaker documents and can be edited with the same powerful tools FrameMaker provides for general XML authoring.

Working with Read/Write rules

The Read/Write rules in FrameMaker determine how XML and graphic objects map to FrameMaker document objects. This can include mappings from XML elements to FrameMaker elements used for special document objects such as tables and footnotes, and can provide a means of translating element names in FrameMaker to different element names in the XML in the save and load process.

Read/Write rules are two way and symmetrical by default, but can be specified to only affect translation in one direction (that is, either when saving or loading). Read/Write rules are maintained in a simple text file that is referenced by the XML Application.
Integrating with other systems

Because FrameMaker lets you author and exchange XML, FrameMaker content can be used by any application that works with XML. FrameMaker can also be customized to meet specific needs, such as direct integration with workflow management or document management systems, through the freely available Frame® Developer’s Kit (FDK). FrameMaker also enjoys out-of-the-box integration with many third-party tools.

Systems that can be integrated with FrameMaker in an XML publishing solution include document repositories, workflow management systems, and relational or other database systems.

Custom integration using the FDK in FrameMaker

The FDK provides a way to customize the FrameMaker environment and automate interfaces with other systems. The FDK lets you control the FrameMaker application through its powerful API. The API exposes all FrameMaker document objects to programmatic control, letting you write C language programs that can do anything an interactive user can do—and more.

The FDK lets you create plug-ins that extend the capabilities of FrameMaker (such as adding custom menu items that will run functions you have created). With the extensibility provided by the FDK, FrameMaker can be integrated with many third-party applications. The FDK can play a variety of roles in the architecture of publishing systems built around FrameMaker—from providing connectivity to databases and other information sources, to directly controlling the FrameMaker composition engine.

Content repositories

Content repositories can house XML and binary formats, and can serve as the central data and document storage location in a collaborative authoring environment.
Document management can occur on the file level, where entire documents are checked in and out of a central repository, or on a more granular level, in which XML components representing document sections can be accessed individually. In more granular forms of document management, repositories can allow simultaneous editing by multiple users of different sections or components of a single document. Content repositories typically provide version control and a record of changes made to documents between versions.

**Workflow management systems**

Authoring can often occur in collaborative environments, where many authors work on a single document and have specific responsibility for either particular dimensions of documents (for example, the structural dimension owned by a document architect or the presentation dimension owned by a template designer) or specific document components. Additionally, there is generally a review and approval process for published content that often spans multiple departments.

Workflow management systems can help streamline the authoring, management, and delivery of published content.

In such an environment, workflow management systems can help ensure an efficient flow of the authoring, review and approval processes. By continuously monitoring the state of such processes, and providing notification such as triggered e-mails in response to defined events and milestones, workflow management systems can make collaboration efficient and effective. FrameMaker can interface with workflow management systems either directly (through customization using the FDK) or indirectly (through intermediary systems such as document repositories and content management applications).
Database systems
Information from database systems can flow into FrameMaker by means of intermediate translation to XML (either entire documents or document fragments) or by direct access from the FDK or third-party tools such as Miramo® software from Datazone or PatternStream® software from Finite Matters Ltd. Source data mapped to XML elements can be formatted for presentation with the same context-sensitive mapping used for XML documents.

As database systems increase support for XML with features such as queries that return results in XML and capabilities to persist XML along with relational data, it becomes easier to use the XML formatting capabilities of FrameMaker to publish information from database systems.

Defining XML publishing streams
One profound benefit of XML-based, single-source authoring is the publishing power that such a method typically affords. Because FrameMaker offers industry-strength page composition and layout capability, and tight integration with PDF, high-volume publishing to print and PDF can be fully automated using tools such as FrameMaker Server. Because core content is available in XML, it is also possible to leverage the expanding range of available tools for processing and publishing XML.

Server-side generation of PDF and print output
FrameMaker Server provides batch or server-side generation of formatted print, PDF and other forms of output. FrameMaker Server uses the same core engine as the FrameMaker desktop product, so such publishing can fully leverage documents and templates created in the desktop version of FrameMaker.

FrameMaker Server provides a formatting engine component for automated publishing applications. FrameMaker Server can be customized and integrated with other application components with the FDK, or can be used with third-party programs such as PatternStream or Miramo software. The flexibility of the FrameMaker Server applications allows for a number of different architectures to meet a range of possible publishing and systems integration requirements.

In one possible architecture for delivering XML content to print and PDF output, XML that conforms to the same XML DTD of a FrameMaker XML Application is streamed into FrameMaker under control of a lightweight FDK client. The source XML is formatted for print or PDF output according to the rules of the XML Application.

HTML and other output using WebWorks
WebWorks Publisher can facilitate automated delivery to HTML and other formats. Like FrameMaker, WebWorks Publisher follows a template-driven model of mapping structured content to formatting. While FrameMaker focuses on the mapping between XML structure and page layout, WebWorks Publisher focuses on delivery to web formats, including HTML, XML and associated graphic formats such as GIF, JPG, and PNG. Together, FrameMaker and WebWorks Publisher provide a powerful means of publishing from a single XML source to a diverse stream of media and formats with fine-grained control over presentation and a high level of automation.
As the nature of HTML delivery can be very different from formatted print and PDF delivery, WebWorks Publisher offers tools specific to the demands of publishing to the web. While it is often acceptable to deliver a large document to print or PDF in its entirety, this is not generally the case with web pages: Bandwidth constraints demand that HTML content is delivered in manageable components. For small or large articles, content will often be spread across multiple HTML pages. WebWorks Publisher provides an automated, customizable means of "chunking" documents into multiple HTML pages, with the automatic generation of framesets and other navigation aids for efficiently accessing such content. Large XML files that are published in multiple volumes for print output can be automatically delivered to the web in an efficient, easily navigated fashion.

With Adobe GoLive®, you can easily customize the professional-looking templates that are included with WebWorks Publisher, or you can build your own templates. WebWorks Publisher also supports delivery to Microsoft Reader and Palm Reader formats.

**Output to a range of media using XSLT**

You can use the XML export capability of FrameMaker to produce high-quality XML output. You can define your FrameMaker XML Application to automatically apply XSLT to transform your XML to HTML or to XML-based formats such as WML, VoiceXML or XHTML accessed through Wireless Application Protocol (WAP)-enabled cell phones, Personal Digital Assistants (PDAs) or other devices. Your workflow may include XSLT processing outside of FrameMaker, perhaps on a web server or through a batch process.

As XSLT can produce XML output conforming to an arbitrary DTD or Schema, this method of delivery from FrameMaker can offer potential preparedness for the future. Changes to XSLT stylesheets may be the only change required to adapt to new XML standards or new versions of existing standards. With XSLT, metadata of source XML authored in FrameMaker can be used to provide specific information for delivery to a particular format (for example, an XML attribute defined in FrameMaker may determine the inflection of speech output to a VoiceXML browser).

**Next Steps**

If, after reading this paper, you feel that your organization could benefit from structured authoring, a critical assessment of your needs and an evaluation of potential tools might be useful.

**Planning for an XML solution**

When contemplating an XML solution, there are three broad areas to consider:

- Input to the system (source documents, data sources and legacy information)
- Output from the system (the required media and formats of published content)
- Internal workflow and specific organizational processes for content creation, aggregation, collaboration, and review

By understanding such required inputs, outputs, and desired forms of collaboration in the authoring process, it is possible to identify the correct system architecture for an XML solution.

Structured authoring system considerations:

- The types of source document that may flow into the system (word processing files, XML source documents, or document fragments)
- The degree of connectivity to real-time information systems (relational databases, server-based applications, and web services)
- The formats of integrated graphics (SVG, TIFF, and EPS)
- The degree of direct usability of input sources (any known requirements for structural transformation, file conversion, or other form of preprocessing)
Published output considerations:

- The range of desired output media and formats (PDF, HTML, and XML)
- Specific document structures required for XML output (industry-standard DTDs)
- The required throughput of the system (documents per hour for each output format)
- The scope of required personalization and interactivity of the system
- The complexity of document formatting and the degree of advanced composition requirements (autogenerated indexes and TOCs)

Issues to consider regarding internal workflow include:

- The granularity of document sections being authored (the extent to which authors "own" entire documents vs. particular document components)
- Specific needs for review and approval (the number of reviewers and departments involved, and handoff points with other workflows in the enterprise)
- Archival needs (what formats are required or desired for archival, for what period of time and with what form of access)
- The number of roles in collaboration (larger authoring groups will often assume specialized roles, while smaller groups may overlap responsibilities)

Together, XML and FrameMaker offer a flexible solution that can accommodate a diverse array of needs. FrameMaker offers powerful XML authority and multichannel publishing capabilities, and can either be customized to interface with other systems or integrate with a number of third-party publishing systems out of the box. FrameMaker can be the ideal tool for organizations that hope to enhance their structured authoring capability or migrate from an unstructured to a structured XML authoring environment.

Evaluating FrameMaker
For more details about FrameMaker, including information on the availability of evaluation versions of the program, please go to the FrameMaker product Web page at http://www.adobe.com/products/framemaker.

Appendix: Resources

- http://www.w3.org/TR/REC-xml
- http://www.w3.org/TR/REC-xml-names/
- http://www.w3.org/XML/Schema
- http://www.w3.org/TR/xslt
- http://www.w3.org/TR/REC-CSS2/
- http://www.w3.org/Graphics/SVG/
- http://www.oasis-open.org/committees/docbook
- http://www.oasis-open.org/committees/dita/
- http://www.w3.org/WAI/
- http://access.adobe.com/

FrameMaker product page
FrameMaker Server product page
XML specification
XML namespaces specification
XML Schema home page
XSLT specification
CSS2 specification
Scalable Vector Graphics (SVG) home page
DocBook technical committee home page
DITA technical committee home page
W3C Web Accessibility Initiative home page
Accessibility information about Adobe products