XML and Content Reuse Systems for Instructional Design
Part II
Implementing Content Repositories & Selecting Tools

Henry Meyerding

January 2004

Abstract

This is the second of a series of articles about XML and content reuse systems for experienced instructional designers. The first part explained the underlying concepts of XML and content reuse systems and related those concepts to the instructional design process. This article discusses taxonomies, processes and tools that can be used in conjunction with different source repositories. The final article describes different implementations and the determination of the return on investment for content reuse systems.

1 Introduction

Write it once ... use it many times.

The previous article in this series discussed the basics of XML as it applies to learning content management. Briefly, XML is a derivative language from SGML that represents a simplified and optimized approach to creating databases. In other words, XML is an object language that allows you to create learning objects and to transform them into a variety of different forms.

Granted that this can be done, why should it be done? This article focuses on many of the methods and benefits of organizing content into repositories and the tools that can be used to create the content stored there.

2 Taxonomy
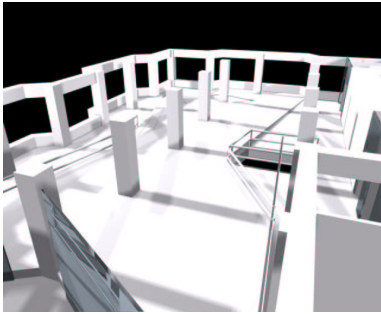
Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer. – IBM Training Manual 1925

Our scientific understanding of any topic is founded upon taxonomic processes: we take things apart to see how they work. We can gain a better understanding of the intricate parts of a whole system by examining its parts and then combine them together, gradually coming to understand how those parts interrelate.

In a very basic sense, what a content reuse system does is to divide up content scientifically into associative, functional, or structural taxons[1]. This taxonomy of information makes useful reuse feasible. The application of this useful taxonomy to enterprise information is what determines whether the content reuse system produces benefits for the organization or becomes just another expensive good idea.

All learning objects are defined by taxonomies. These taxonomies express the way in which each object is understood, used and maintained. In evaluating how to construct learning object models for an XML repository, it is very important to understand that these are used to define queries. The value of the system depends upon the ease and accuracy of queries. Many organizations discovered too late that they had expended substantial resources in creating an XML (or SGML) repository that provided no additional benefit over cutting and pasting documents from a file server. This is because their content authors could not find anything that was placed into the repository.

2.1 Repository

A content repository has several different purposes:

- Store controlled versions of documents

- Store current versions of learning objects

- Store in-work versions of learning objects

---

[1]A taxon is a category of information. An internally consistent collection of taxons constitutes a taxonomy

- Publish content to web servers

- Publish content to other servers (LMS)

- Function as ISO Repository

The most important reason for having a repository is to to facilitate collaboration between content creators, editors and production staff. One mistake often made with a complex repository is to make customized views that are not shared between different team members. This can be frustrating and time consuming.

Once everyone has gone through the arduous task of chunking and labeling their legacy content, this content needs to be put into a repository where it can be easily accessed. The best way to do this, for instructional designers, is to put the content into a version control system that is linked to a database. ClearCase, for example, can present several different views of the repository for different uses. One view presents a virtual file server that contains all the most recent versions of the training documents. Another view presents selected documents to a web server or LMS. Yet another view presents the XML database elements.

Other views can be developed for specific uses, such as creating archives of content, presenting catalogs of approved artwork or source content for other servers such as Adobe Document Server or FrameMaker Server.

The road to XML content reuse is a simple progression of responses faced by learning organizations. Generally speaking, there are six steps taken on the path from no content sharing and reuse to a comprehensive XML repository system:

1. **File Server** - A "shared drive" accessible to all team members with read and write permissions to all.

2. **Version Control System** - A collection of documents, stored by document version to protect against accidentally overwriting files.

3. **Document Manager** - A software system that provides different levels of access to documents based upon selectable criteria.

4. **Learning Management System**(LMS) - A system that provides access to learning content for students, authors, and editors. The modern LMS usually provides some kind of virtual campus paradigm.

5. **Learning Content Management System**(LCMS) - A system that divides up learning content into manageable components, which can be dynamically revised in some or all of its instances in the curriculum.

6. **XML Repository** - A system that applies content taxonomies to organize content into associative, and structural classifications so that content can be created and managed with maximum efficiency.

Not every organization progresses through each step in an orderly manner. It is often the case that different groups within a learning organization implement different steps and different times and then face significant challenges integrating the results. The following table summarizes some of the objectives and limitations of each step in the progression:

Step	Objective	Limitation
File Server	To permit access and sharing of files between many users.	Slow, insecure and does not scale well
Version Control System	To maintain different versions of the same document so that the newest (best) can be identified.	Complex to maintain and difficult to use when additional features are added.
Document Manager	To automate more complex features (and rules).	Proprietary - software does not keep pace with new tools and processes.
LMS	To improve the efficiency of training content delivery and progress tracking.	Can limit designers in terms of format or delivery methods, may not accommodate editing and version control well.
LCMS	To improve the efficiency of training development through content management and reuse.	Often includes a poor user interface; extensive customization required.
XML Repository	To provide content reuse, multiple output formats, and extensibility to react to changing needs.	Requires rethinking of the development model by designers.

When computer networks became common in the workplace, people abandoned the file cabinet for the file server. They soon learned that file servers have their own defects when it comes to sharing important information. The next logical step was to try to remove the most glaring defects of the file server by implementing a version control system. The version control systems made it safer to put your documents onto the network and easier to find things, but when large numbers of people put large numbers of documents into the system, it became harder again. Enter the document management system, which made it simpler to find things, but which usually locked you into tools and processes, which rapidly became outmoded.

A good example of this last hurdle to progress is a large legal firm that implemented a complex macro-language driven documentation system that interoperated with their

document management system. When the next version of MS-Word arrived, they were very upset to find that there was no backward compatibility. So, they remained with the older version of MS-Word for ten years.

Learning management systems (LMS) are student-facing applications, primarily. Their purpose is to present training to a student population and to provide tracking of student performance. Over time, more and more content management facets have been sneaking into these learning delivery platforms. That is not their core function[2]. Although learning content management systems (LCMS) are designed to efficiently manage content, they suffer from a lack of flexibility and timeliness.

Everything that is true of the document management tools locking you into particular tools and processes is true of LMS/LCMS deployments, only much more so. Most LCMS systems have their own content creation tools, which may be very well intentioned, but which also fall very short of the functionality and finesse represented by other commercial applications. Of course, most will work with major content generators (more or less) such as MS-Word and Adobe FrameMaker, but they increase the complication of version upgrades by several orders of magnitude. This is a significant expense that must be factored into the cost of ownership and operation of these systems.

The best of the available LCMS systems are blended XML solutions. These systems use XML/XSLT technology as a tranformation mechanism, but retain a proprietary data architecture for database functions. In this way, they have many of the advantages of XML technology, such as interoperability, SCORM-compliance, and access to XML enhancements, and they can also customize the database engine to provide better system performance for content management functions. OutStart Evolution®, and Aspen™, and learn eXact® are all examples of the blended XML systems.

Once you have an XML repository, your repository can inter-operate with other systems, such as LMS or even LCMS, but the content is organized for your exclusive needs and convenience. If your needs or tools change, so can the repository. You have created for yourself an "Open Source[3]" solution. For that reason, the XML repository is simpler and less difficult to upgrade than many proprietary solutions.

XML and SGML were developed specifically to provide a structure and methodology for content reuse. Many of the lessons learned from early SGML implementations were built into XML, which provides a more streamlined and less labor-intensive means of achieving high quality content reuse.

---

[2]To deliver existing content to students efficiently.

[3]An open source application is one that you have access to every line of code. If you have the expertise, you can modify it in whatever way is necessary for your own purposes, rather than bartering with a vendor to get changes implemented second-hand.

2.1.1 Proprietary Footnote

Question: If XML Repositories are so great, then why doesn't anyone market an XML repository as an LCMS?

Answer: Practically all LCMS vendors are organized according to a service consulting business model. They invest massive amounts of time and money to create efficient systems, which they practically give away for free. They do this so that they can sell you customizations, service, training, maintenance, and support. A pure XML repository system could be serviced and maintained by a wide variety of vendors. They might never earn back the investment they made in creating the solution.

The proprietary product offer does tie the business to the vendor, but it also ties the vendor to the business. The vendor has a huge stake in the outcome of the LCMS implementation.

There are some pure XML repository LCMS solutions that have been developed by the Open Source community (principally by and for academic institutions). They are more like do-it-yourself kits than a fully-developed product offering and do not offer the reliability, features, or performance of COTS[4] solutions.

2.2 Reusing Content

Legacy content comes in many different forms. Most of these forms represent document instances. Most organizations attempt to maintain a repository of these document instances according to some meaningful hierarchy. ISO[5] documentation standards are an example of this kind of document-centric hierarchy. If documents are correctly named, stored and updated, then the information they contain can be reused, but the process is slow, laborious and susceptible to human error. The utility of simple file sharing is inversely proportional to the number of documents to be shared.

When existing content is chunked, it usually begins in documents that are broken down into component topics and then broken again into smaller pieces identified as introduction, main body, and transitions. Content should sound natural and appear to have been written specifically for each use. Content also is chunked by audience and complexity so that relevant material and more complex discussion can be added or removed easily.

Audience plays a big role in content reuse. Identifying specific blocks of informa-

---

[4]COTS: commercial off-the-shelf.

[5]ISO. A network of national standards institutes from 147 countries working in partnership with international organizations, governments, industry, business and consumer representatives. A bridge between public and private sectors[7].

tion as appropriate or inappropriate for different audiences can simplify document creation immensely. It also is the hardest classification to accomplish.

For example, an Offer Brief: a document that quickly informs sales staff of new offers, pricing and conditions that apply to selling a product or service within a given market. These things are constantly changing. It is a Stygian task to keep this kind of training content accurate and timely. Most of the documents have a similar look and feel. There may be specific types for different audiences or products, but a single item of information may find its way into 30-40 different presentations. Along the way it may get a different style - it may appear in a table here and in a paragraph of text there, but the data behind it is identical. It is possible to do a keyword search through a documentation set and locate all known matches, copying in the revised information with each new iteration. That usually takes too much time and trouble to be worth doing on a regular basis, unless it is very special information.

In comparison, with a properly constituted XML repository, the process is much more direct. Instead of working backwards from finished documents to find the appearance of specific content in context, the source content is already organized according to what it contains. The author goes to that container, revises it, refreshes the repository and the next time the document instance is called, it collects its source content from the updated source, applies the proper formatting, and compiles the finished document. All 30-40 documents that touch this same source content are automatically updated.

There was more work done in the very beginning, to properly analyze and attribute the content, but as the content is used to create more and more instance documents, those documents become progressively less expensive to create, manage and update. It makes it possible to do the previously unthinkable:

- Individualized training syllabus for every employee.

- Weekly updates acrossc training syllabi.

- Monthly updates to training.

- Global identification of misinformation; global liability reduction.

- Personalized web-based training tied to employee reviews.

By increasing the efficiency with which content can be created, the quality and timeliness of all the training deliverables can be increased without raising the cost into the stratosphere.

3   Process

For every human problem, there is a neat,
simple solution; and it is always wrong
– H. L. Mencken.

This section describes the development process to implement the XML content reuse system. Each description includes a discussion of the costs and benefits associated with each process.

3.1   Manual Reuse Systems

In traditional, project-oriented design settings, each new project was a separate entity. Analysis, development, and production were defined by the time line and requirements of each discrete project, and instructional designers produced design and content as an artisan custom-crafting a product for a customer. When this process has worked correctly, it has worked very well. Students receive curriculum that is specifically fashioned to address their needs. Trainers and designers can be student advocates at many different levels. Everybody wins. However, there are some important limitations to this methodology.

It is important to understand that these limitations and disadvantages are not a function of the skills or artistry of the designer. However dedicated and talented a designer might be, armed with a typewriter and a mimeograph machine, he will be at a disadvantage compared with someone of, perhaps more pedestrian talents, but provided with computers and web-based delivery options.

At the same time, it must be admitted that the best tools will not make a poor designer produce excellent training content. Really good tools have been used to camouflage poor design. It is certainly easier for a incompetent instructional designer to produce much more crummy training deliverables with an XML content reuse system than when working alone with MS-Word.

Assuming competent designers. some of the most important limitations and disadvantages of the cottage industry approach to instructional development are:

- **Inconsistency** - Since every project is independent of every other, it is very difficult to create and enforce standards. Even if templates are used, designers tend to create exceptions.

- **Inefficiency** - There are many opportunities for reusing content that are missed, either because designers are unaware of legacy content that could be adapted, or because the legacy content is in a format that makes it difficult to adapt to their current project.

- **Inaccuracy** - Because each project recasts some of the same information in a different way, there is no way to globally update information and reissue training when changes occur.

- **Scalability** - As workloads increase and staffing levels decline, there is no way to maintain output and quality levels. Designers become frustrated, being unable to meet the expectations of their audience.

- **Tool Costs** - Reliance on outmoded tools, different versions of standard tools and fringe tools complicates things and makes people less efficient. The cost of maintaining learning materials sourced in multiple tools is enormous. Standardization on a few tools and methods makes a substantial difference to the production cycle.

3.2   XML Automated Systems

The following figure describes a content authoring/delivery system for both online and hard copy training deliverables. In this example, light blue indicates tools from Adobe, orange Macromedia, yellow Microsoft, and purple for open source components or outputs. This is only one of many equivalent solutions.
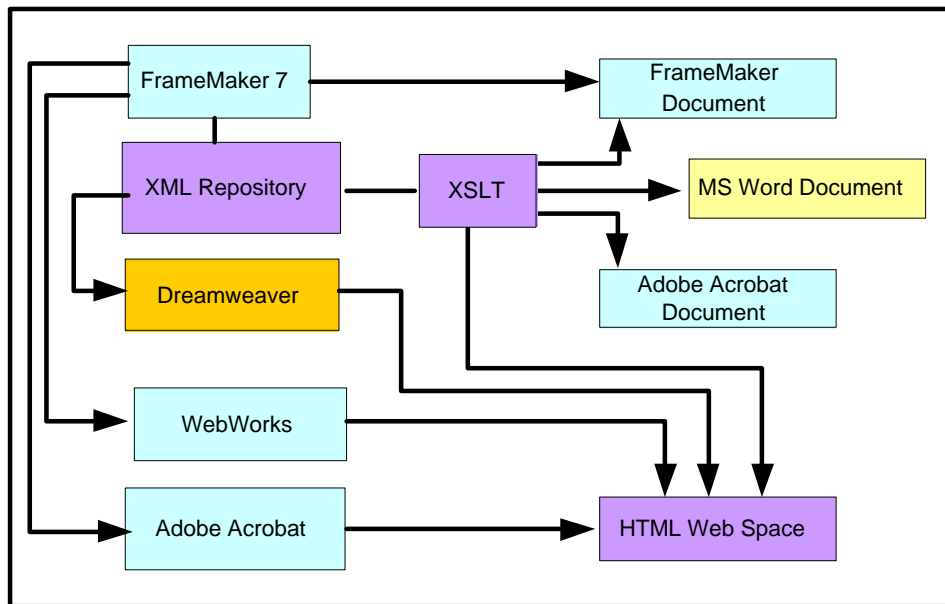
The structured approach to instructional design is seen to have the following benefits[6]:

- The same courses are delivered across multiple media and delivery environments. Just because it happened to be developed by X with Y, this doesn't stand in the way of it being reused in a completely different environment or with different tools.

- The structured development model supports a consistent instructional design and development process. Designers have many new options that come from an efficient production design.

- XML content can be analyzed and repurposed much more efficiently than legacy content. The content does not hide in a forest of words. When needed, new and legacy content can be efficiently blended to create educational tools to suit different needs of different student audiences.

- Learning content is organized for use. Related content is accessible. Related procedures and policies are obvious - as are conflicts and inconsistencies.

- Because the relationships between concepts and ideas are mapped according to the taxonomy by which the content was chunked, identifying content for reuse and the updating of legacy materials is streamlined significantly.

- Conforms to Information Technology standards to ensure portability and long-term use.

There are three steps in the process of implementing an XML content reuse system: 1) Analysis, 2) Chunking, 3) Operation. The process is very simple, in theory:

- A DTD is selected and tested.

- The repository is created using tables that mirror the DTD.

- Legacy content is converted to XML.

- XML content is placed in the repository.

- Users query the database to construct new documents

- Users add new content to the repository as needed.

As mentioned before, the initial analysis is perhaps the most difficult stage of the implementation and it is the one stage that has the most persistent effects. Having once decided upon the one and only way the content will be parsed, staff members are trained carefully in how to accomplish the chunking of legacy content into the system.

3.3 Legacy Content Chunking

Whether this chunking process is slow and manual or quick and automated really depends on how much legacy content was created using standardized styles and templates properly. If practically none of the content was created using standard styles and templates, then there is a great deal of manual evaluation that must be done.

The most important aspect of the chunking process is to have the people doing the chunking UNDERSTAND what they are doing. This is best accomplished by providing them with thorough training, support, and supervision. Consistency is the key. Select a single process, train everyone in that process and execute the process without exception.

NOTE: The importance of thorough and consistent content editing increases by several orders of magnitude when content is entered into the database. *Enter it wrong once ... use it wrong many times.*

> "Organizations that implement highly configurable or customizable products need to rely on their software vendors to meet the early training needs of the planners and technicians. To the degree that they wish to own or control product configuration, customization, and the ongoing support of those modifications, they also need to be prepared to invest in the staff development required to enable those capabilities."[5]

There are two approaches to legacy content that are usually successful.

- Identify a small select team of designers who specialize in converting content. They do nothing else until the original body of required content has been put into the database.

- Spread the conversion duties among all the design team. Each member converts documents among their other duties, but at least a fixed minimum number of hours per week.

The advantage of the first method is that you generally obtain a more consistent conversion with fewer errors. The advantage of the second method is that you train your entire group in the XML database and process. You also may learn some things early on that allow you to modify the database or your processes so that they are more applicable to your training.

As with any complex operation, when there are advantages, there are also risks. The risk inherent in the first method is that it may result in a fully functional content base and no one trained to use it properly. The second method risks creating a database with so many inconsistencies that it is practically useless. The correct method for each organization depends upon the technical background of the team and their workload. Organizations with lower levels of technical proficiency and higher per capita workload generally do better with the first method.

3.4   Using Chunked Content

The theory of developing new documents from legacy components is fairly simple, if the repository is implemented properly. First, the designer needs to know what previous training this new training is similar to. This is accomplished by querying the database and seeing what existing content comes fairly close to the current need. If it is completely new and dissimilar from other training, then the designer gets nothing from the repository but templates. Having made a shrewd guess about some other similar training, the designer has to define how this new training is different from the similar training that has been identified.

One method of handling the query process is by a web page containing drop down field list properties. Define 5 or 6 of these and then add in some more specific customizing terms, click submit and get a list back of matching content. It is just like doing a web search, except that the web you are searching is a discrete database. What is returned from the search can take many different forms: FrameMaker document, raw XML, Word document or HTML. When the query results in more "hits" than desired, then you reformulate it to be more specific. If little or nothing results, then you try a more general query until you get the desired results.

The authoring process is iterative, a succession of repetitive operations performed to collect, modify and upload new content.
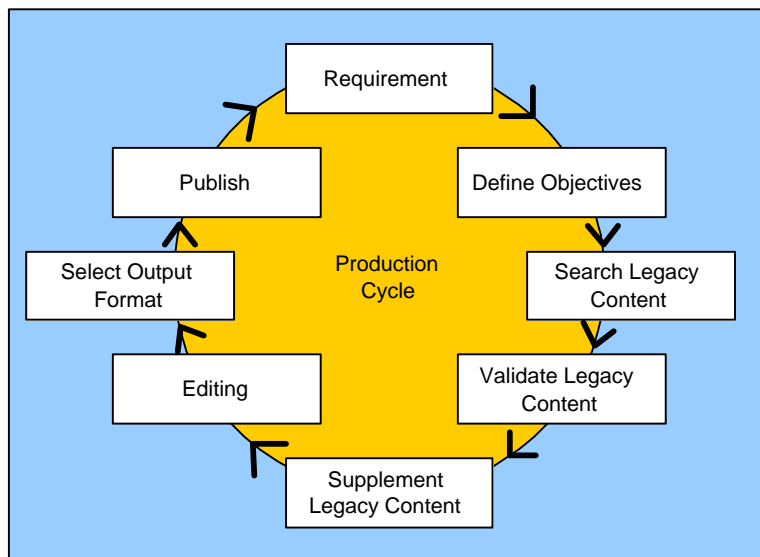
As time goes by and the authors and production people get used to using the system to produce the required results, productivity increases and frustration decreases. There will be some people who cannot adjust to the new work methods, just as there were some very talented people who could produce marvelous typed documents who could never quite make a word processor work right.

Some authoring environments, such as Epic Editor, work from the data structure to the content. At the beginning, these tools can be difficult for some designers to understand and use efficiently. After the designers become familiar with the database structure, they rapidly learn to navigate through the maze of information they encounter on cross-functional teams to find the parcels they want. In practice, authors working with common, standardized documents rapidly learn what five or six elements they must identify to generate the greater portion of their training. It is more difficult, at the beginning, than cutting and pasting content, but once you get it into your stride, it becomes 10 times faster and easier to do your job. Even in a pure XML environment, designers still find invaluable the ability to easily query the database.

It should be noted here that no content management system can stand in for the designer's knowledge and understanding of the corpus for which training is developed. XML has no real impact upon the analysis or discovery phases of new training development. XML is a set of tools. Having the skills to manipulate those tools does not in and of itself result in training any more than reading a manual makes you an expert.

How the content is organized into new instances is a question of authoring tools, not XML.

3.5   Multi-Sourcing

Multi-Sourcing has been the Holy Grail of the documentation industry for a generation. Simply put: information goes in the hopper, press button A and a good marketing document results. Press button B and you get a User Manual, and pressing C creates the getting started pamphlet that goes in the box with the product. The heart of multi-sourcing is content reuse.

Until the advent of SGML, content reuse was impractical. Until the advent of XML, content reuse was out of the reach of all but the largest organizations and institutions. In the last 30 years, tremendous advances have been made in content reuse technology to enable multi-sourcing of documents.

Whenever the notion of content reuse is raised, one hears the same kinds of objections voiced time and time again. These kinds of questions are entirely typical and a natural reaction to the concept of content management and reuse. Adopting an XML or other reuse system asks people who already know how to do something well to change their process and to adopt methods they do not know. The following FAQ[6] address the four most common questions from seasoned designers:

Q1   How will XML help me to tailor my materials to meet the needs of my audience?

A1   When you are creating training now, if you have a good, useful piece of content that speaks to the same point in another class – don't you copy and paste it in? If you could do this more often, and maintain the same quality of output, would you do it? We all do that, within the body of our own work, and sometimes from other authors, too. We use our own documents as source for reuse more because we are intimately familiar with them. We know we can find that great paragraph we used to describe that weird thingamajig. More seasoned authors annotate their own works with notes that help them find those good opportunities to reuse content. If we work with another designer who does the same for long enough, we can get so that we can read each other's notes and make more use of each others' work as reuse content. XML, in this sense, is like a common form of notes with which we annotate each others' work so that we can access it and reuse it, when that is be best thing to do.

---

[6]Frequently Asked Questions

Q2 How can I leave out technical information in one document, but include it in another?

A2 That gets to be something of a tool question. What are you authoring with? You may collect the entire content for a document, run through it, individualizing it for this instance, cleaning up any transitions and output the result. Instead of copying and pasting text between documents, you are attaching document objects to one another like a jigsaw puzzle. Pieces that work well consecutively have the right "shape" to fit together that way. They add up together, you edit them as needed and produce the finished product.

Q3 What prevents this from resulting in documents that seem mechanized and impersonal?

A3 This is not a machine imitation of human communication. This is human beings using a system of shortcuts to make their work easier and more productive. It certainly can sound mechanized and artificial, but it doesn't have to be so. When the same item or process is described identically in five different places in 4 different classes - is that mechanical or is it using repetition to reinforce?

Q4 What about the shifting voices of the authors; won't that cause confusion?

A4 Do you have more than one instructional designer on your staff now? Are your students confused by having to attend classes created by different people in isolation? When everyone else's work is more available and when opportunities for collaboration and knowledge sharing are facilitated by the system, instead of hampered by it, will not that help these different people find a more common voice. If you never sing in a choir, you never get the knack of singing like everyone else.

3.6 Single-Sourcing

All printed documents, since Gutenberg were multiple copies of a single original source (single-sourced documents) until quite late in the 20th century. Advanced printing technology allowed compositors to create multiple versions of documents by reusing the same printing plate sources in different permutations. Computers made practically anything possible, but a comparatively tiny slice of the possible became routine.

In an enterprise environment, there are many uses to which information is put. Some of those uses include documentation, training, knowledge-base applications and marketing. Traditionally, these disparate uses have all maintained their separate knowledge management environments. As a result, the information provided by these various sources is usually inconsistent. In the worst case, considerable misinformation results.

No one would think of using typewriters and mimeographs for corporate communication, although these were once ubiquitous. In the near future, single-source systems will seem just as antiquated.

When these information sources are unified into a single repository, from which all outputs derive, significant improvements in efficiency, consistency and overall quality of information result. Also, when the costs of implementing the content repository are spread among different organizations within the enterprise, a greater return on investment naturally occurs.

Communication is the unstated core competency of every successful business. When the information about its products, processes, policies and procedures is available to all associates, this has a unifying effect on all the organizations within the enterprise. Although the process and deliverables of different organizations vary tremendously, their need for accurate and timely information is identical.

In its best form, the XML content repository can be a significant competitive advantage to an enterprise, particularly one that operates in diverse markets. In this sense, the economies and productivity conferred to the training organization are a byproduct of a larger benefit to the entire enterprise.

4   Tools

Our view of the possible is shaped by our tools.
– Carl Sagan.

This section analyzes some of the common tools that can be used with the XML content reuse repository. There are many tools available and one size does not fit all. The choice of tools is an important one, because the tools will have the biggest and most immediate effect on the designers. For that reason, it is very important to include designers in the tool selection process.

Anyone who grew up in the typewriter age might well be amazed at the layout, page formatting and document management capabilities of the current crop of software applications. As with all technology systems, there are prerequisites and agonizing revelations – and at least three ways to do something:

1. The RIGHT way - the way originally envisioned by the developer and facilitated by the program. This way works best in the long run.

2. The WRONG way - the way that someone found to make it work, because they didn't know what the right way was. This way complicates editing and later revision of the material.

3. The OTHER way - the way that outwits the program and allows you to do something that should not be done, but needs doing. This way has everything wrong in common with the WRONG way, with the added disadvantage that it may actually make your application or their documents unstable.

Unfortunately for anyone who is facing the prospect of converting documents from various formats to XML, there is a considerable amount more WRONG and OTHER than there is RIGHT out there to be converted. Computers are infinitely stupid and must be told precisely what to do. In order for consistent content to result from an automated conversion to XML, consistent base content must be available.

Consistency in the use of content creation applications is not a hallmark of most groups of instructional designers. Designers on a deadline are pragmatic and care more about making it work now than about finding out how to make it work right later. It is paradoxical that a less intuitive tool, which requires more instruction and has a steeper learning curve, may be used more correctly and consistently than the naturally intuitive tool that everybody figures out for themselves.

4.1 Microsoft Word

Microsoft Word is the ubiquitous tool that does not play well with others. It has a long history of file format changes and inscrutable macros. Whether it can be used in conjunction with an XML content repository – and how well it can be used – comes down to two things: styles and templates.

To use Microsoft Word as an authoring tool is certainly possible. It is a fairly simple process to create an XSLT to convert XML content into a *.doc or *.rtf format so that it can be brought into Word. For example, if you are working with an XML document instance, you can process that instance into an *.rtf and send it to a reviewer who prefers to edit in Word. The problem happens when that review is returned to you and you wish to transfer those edits back into XML content.

Because users seldom use Microsoft Word properly[7], it is rarely possible to convert Microsoft Word files to XML programmatically. Therefore, using Word decreases the productivity of the designers. Word does not operate in a manner consistent with structured documents. Using Word to author XML is like eating soup with a fork: you can do it, but it complicates things.

It is also true that practically every new version of Microsoft Word incorporates a plethora of undocumented changes in the file format. Changes in the format of the resulting Word files invalidates any programmatic automation that has been created. For this reason, most XML content systems use the more stable, but less capable, *.rtf format to transfer files to and from Word.

Many people[8] consider that Microsoft Word has no place in an enterprise XML content reuse system. In this view, using word processor technology to author content objects is counter-intuitive, inefficient, and ineffective. Regardless, people resist trad-

---

[7]People quite often learn to use Word by trial and error without instruction. They seldom know how to use templates or the styles they contain. When they want to have something in a different font or size, they apply that change from the tool bar, instead of applying a standard style to the text. Some Word users seem addicted to the space bar: instead of setting tabs appropriately, they achieve their indents through the use of multiple spaces. Word documents often contain revisions, highlighted text and complex section breaking. This kind of formatting makes programmatic chunking very difficult.

[8]"Microsoft Word is the most popular word-processing application on the planet, a fact often regretted by die-hard XML aficionados." - XML Workshop Ltd.

ing tools, even when they have good reason to do so. Some dedicated XML editors, such as Epic(see below) even include filters to import Word content to XML. Indeed, there has been a significant amount of effort to create robust, reliable conversion tools for making XML extracts from Word documents[3]. The newest generation of blended XML LCMS systems, such as OutStart Evolution, include an impressive amount of bi-directional filtering of content to and from MS-Word.

Some other Microsoft programs, such as PowerPoint, can be used to create content and have very similar advantages and disadvantages to Word. Other Microsoft programs, such as Publisher or Front Page, pose another order of magnitude of difficulty in interoperating with content reuse systems.

4.2   Adobe and FrameMaker

Adobe FrameMaker is the WYSIWYG authoring tool of choice for XML applications. FrameMaker 7.0 includes a wealth of features that make authoring XML content much more efficient and practical[9]. There are direct exports for both HTML and PDF document instances. Authoring in the structured view provides designers with an excellent means of understanding and using FrameMaker to create valid XML documents[10]

Adobe FrameMaker imports the XML data elements into a template. That template defines styles associated with the element definitions in the element definition document (EDD[11]). This means that it is not necessary to parse the XML and XSLT together to result in a formatted document instance. As the document is created, by adding structural components to the current document, the user sees the final format of their document. This only makes sense when you are using FrameMaker as your publication tool. Otherwise, what the author sees is only the best approximation the XML programmer can make with an XSLT of the normal output from FrameMaker styles.

Adobe FrameMaker can export files directly through Webworks to HTML[12] This is an option for training projects that rely extensively on interrelated print and online media. It can be easier to coordinate and publish the learning materials required if they are developed as a single source project. The base content is available from the XML repository, either as FrameMaker files or directly as XML.

---

[9]A structured document view for creating valid XML, several different levels of styles, the ability to discard exceptions to styles are three of the most important features that impact XML.

[10]Well-formed XML conforms to the syntax rules of XML: it is tagged correctly. Valid XML is well formed XML that conforms to the data structure defined in the Document Type Definition (DTD). All valid content is well-formed. Not all well-formed content is valid.

[11]The imported copy of the DTD used by Adobe FrameMaker to validate the XML content.

[12]See the notation under Dreamweaver about working with Webworks generated source code as an HTML output.

Adobe FrameMaker also exports into Adobe Acrobat very well. Creating Acrobat files with FrameMaker allows you to include a lot of advanced Acrobat features[13] directly in the FrameMaker document, rather than having to modify the resulting PDF with Acrobat later. Creating Acrobat files with other programs, such as Microsoft Word, is much less efficient, unless the advanced features of the Acrobat format are not needed.

The downside of Adobe FrameMaker is that all this additional capability comes at a cost: it is not really very intuitive, especially for designers who are accustomed to Microsoft Word. It requires specialized technical expertise to set up correctly. Once it is set up, designers must be extensively trained in how to use FrameMaker properly. Many Word users are frustrated by the additional structure imposed by using XML. On the plus side, FrameMaker helps users to construct valid XML and informs them when their content is not valid. Of course, once they know that their content is not valid, they may need to have someone handy who really knows FrameMaker and its templates to help them fix it.

Templates are the key. It is absolutely necessary to employ a dedicated Adobe Frame-Maker expert to create templates. Most organizations do this on a consulting basis with one of the many Adobe/FrameMaker consulting firms.

Adobe has extensive training resources available, for a fee. They have a great deal of experience in implementing Adobe FrameMaker as an enterprise tool. If your organization makes the top-level commitment to pursuing an Adobe-enabled XML solution, the kind of support and expertise available from Adobe is unequaled elsewhere in the industry[2].

4.2.1 FrameMaker Server

Adobe FrameMaker Server provides an opportunity to create a variety of dynamic documents. These documents, when accessed, perform real-time lookups of information from databases. That allows designers to access current information in a printable form. That ability is a great advantage for customer-facing training that requires frequent updates. It also could impact differential training, allowing designers to fill in the blanks with volatile information, instead of constantly trying to keep up with maintenance changes.

FrameMaker server works with the FrameMaker software on the desktop to provide more groupware solutions to enterprise publication challenges. It is designed for working in a distributed networking environment and provides convenient document management functions from within FrameMaker itself that make many group collaborations simpler to manage.

---

[13]Such as bookmarks, different kinds of linking, different security modes and so forth.

4.2.2   Adobe Document Server

Adobe Document Server[1] supports the dynamic creation of Adobe Acrobat documents from XML data. By flowing XML data retrieved from the XML database into document templates, you can generate instance documents and automated forms on demand. These document instances and forms can be highly complex, including graphics and audio to produce bi-fi[14] multimedia presentations. Because they draw their content directly from the XML database, users always get the most current information. In addition, documents can draw upon multiple sources to populate document instances: XML content, PeopleSoft, SAP, LMS and other server content can combine in a single document instance that the user receives.

Adobe has many products and services designed for XML-based solution environments. XML technology and Adobe software work together in a highly complementary fashion (See the figure below, representing the Adobe server solution implementation of XML). This is not an accident.
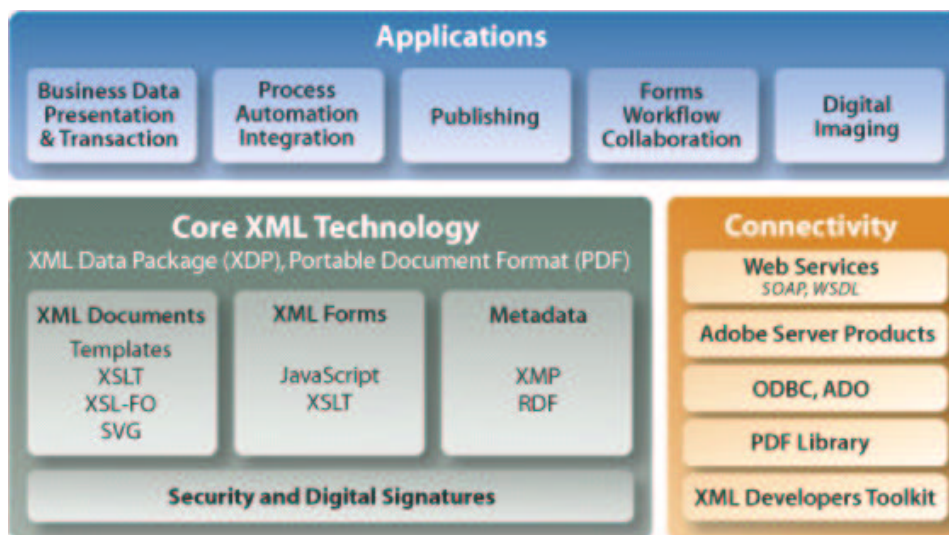
4.2.3   Adobe InDesign and GoLive

Adobe InDesign is a page-oriented[15] software that includes built-in, extensible support for importing and exporting XML files. InDesign also allows you to export pages directly to Adobe GoLive 6.0 to use in dynamically generating Web pages. It supports

---

[14]A term for selectable low-fidenlity/thin bandwidth or high-fidelity/wide bandwidth distributed content.

[15]Page-oriented software allows a lot of flexibility and precision in placing content on each page; used for brochures and presentations. Document-oriented software is intended for larger, more complex documents where management of cross-references, indexes and other features is more important; used for manuals and other books.

Scalable Vector Graphics (SVG) and share native Photoshop and Illustrator files and can share these with GoLive. Through its tagged Adobe PDF support, InDesign exports graphically sophisticated eBooks that can be viewed on different devices. InDesign also supports Adobe Extensible Metadata Platform (XMP) for embedding metadata in documents.

Because it is a page-oriented development tool, as opposed to document-oriented, Adobe InDesign is a good choice for small 2-5 page documents where consistent look and feel is very important: marketing materials, offer briefs, and so forth. Many users find FrameMaker difficult to use in smaller, graphics intense documents. InDesign can be an excellent alternative.

Adobe GoLive is Adobe's competitor for Macromedia Dreamweaver. It does just about everything that Dreamweaver does, only a little differently. What it does not do as well as Dreamweaver is integrate as well with Authorware and Flash. Both GoLive and Dreamweaver will send you scurrying into the source code at edit time. The WYSIWYG editing mode is very nice and handy, but it is maddeningly imprecise. If Lo-Fi[16] Web development is practically all your output, then GoLive may be an excellent choice, particularly if you are wishing to integrate more closely with print-deliverable development using FrameMaker. If Hi-Fi Web content is the majority of your online offering, then Dreamweaver has the edge in integrating with Flash and Authorware.

4.3   Macromedia Dreamweaver

For online content, Macromedia Dreamweaver is one of the most popular WYSIWYG HTML editors. Unfortunately, like Microsoft Word, it is often misused[17].

Dreamweaver's WYSIWYG editor is, as noted above, imprecise and you cannot make many edits without having recourse to the source code. Templates are very important. Dreamweaver uses templates much in the way that FrameMaker does to add format to XML content. Dreamweaver imports XML into templates and generates HTML directly. Dreamweaver also exports XML content, which is efficient for people who like to work in HTML, but want the advantages of an XML repository.

---

[16]Lo-Fi Web content is primarily text with a few graphics and moderate interaction; it is suitable for thin client delivery. Hi-Fi Web content is highly graphical with strong user-interaction; video, Flash, and Authorware content are Hi-Fi.

[17]Many people learn Dreamweaver by using it, without any training. Like Word, Dreamweaver has many buttons and widgets that are convenient, but don't result in very good or consistent HTML. For example, someone may have extensive experience creating Web content with Dreamweaver but not have a clue about using templates.

> **NOTE:** Dreamweaver does a good job of exporting the editable portions of templates as XML. However, it only checks whether the content is well-formed XML[18], not whether it is valid XML.

HTML via Webworks-FrameMaker may not behave well in Macromedia Dreamweaver, since it has less tolerance of HTML code that it interprets as badly-formed XML[19]. It should be noted that using Adobe FrameMaker to write HTML results in Web sites that lack many of the features needed for richly interactive eLearning[4]. Webworks-FrameMaker works best for document based learning, where a large volume of information must be provided to the student as reference material.

The big advantage of Macromedia Dreamweaver is that many people feel comfortable with it. It is another learning step, but a relatively easy one to understand how to import and export XML in Dreamweaver. Again, it is of paramount importance that the templates into which XML is imported are used verbatim. It is a very good idea to have those templates generated by expert consultants if sufficient Dreamweaver-specific expertise does not exist in your organization.

4.3.1   Authorware and Flash

It is perfectly possible to create learning objects in Flash or Authorware and store them in the XML repository. It is usually a good idea to break up longer Flash and Authorware segments into scenes. In this way you can reuse particular content without having to modify a large, complicated segment when only part of it is desired.

4.4   Arbortext Epic Editor

Unlike other text editors that have been stretched to fit the function of authoring XML content, Arbortext Epic Editor was designed from the ground up as an XML editor. It handles a broad range of applications and does a good job of providing an editing interface for XML content. The user interface is user friendly, but not at all like the standard WYSIWYG document editing environment. Like Adobe FrameMaker, it is a groupware product that is specifically optimized to handle:

- Content collaboratively written and maintained by teams of authors working in multiple languages.

- Content created in reusable components independent of their formatting, stored in content management repositories, and dynamically assembled on demand.

---

[18]Well-formed XML conforms to the syntax rules of XML: it is tagged correctly. Valid XML is well formed XML that conforms to the data structure defined in the Document Type Definition (DTD). All valid content is well-formed. Not all well-formed content is valid.

[19]HTML is well formed when it conforms to the syntax of the version of HTML supplied in the document definition

- Content personalized for specific audiences and formatted for delivery on multiple media: Web, CD-ROM, print and wireless.

- Content automation based on systems and software that are easily customized and leverage the broadest available support for XML and related standards.

- Content creation through client-based installations for occasionally disconnected users and through server-based installations accessed by Web browsers for users who are connected full-time.

Out of the box, Arbortext Epic Editor works with file systems and WebDAV-enabled repositories, and has configurable adapters for Documentum, Oracle CM SDK (formerly named iFS), and FileNet Panagon Content Services. Arbortext's other repository partners provide adapters to Epic Editor, including BroadVision One-To-One Enterprise, empolis SigmaLink, Progressive Information Technologies Target 2000, and XyEnterprise Content.

Arbortext offers separate products for content conversion and publishing. The Enterprise E-Content Engine (E3) converts content from Microsoft Word, Adobe FrameMaker and Interleaf documents to XML, and publishes dynamic content to print / PDF and Web / wireless. To publish to CD-ROM, Arbortext offers the CD-ROM Composer.

This E-content engine is an off-the-shelf parser that can be used to automate many different kinds of legacy chunking operations. It does not work miracles: Nothing will correctly parse badly formatted Microsoft Word files. That process requires human intervention and exercise of good judgment. It does provide to the enterprise a tool the equal of, or better than, many learning content parsers that typically require a much higher investment for the same return.
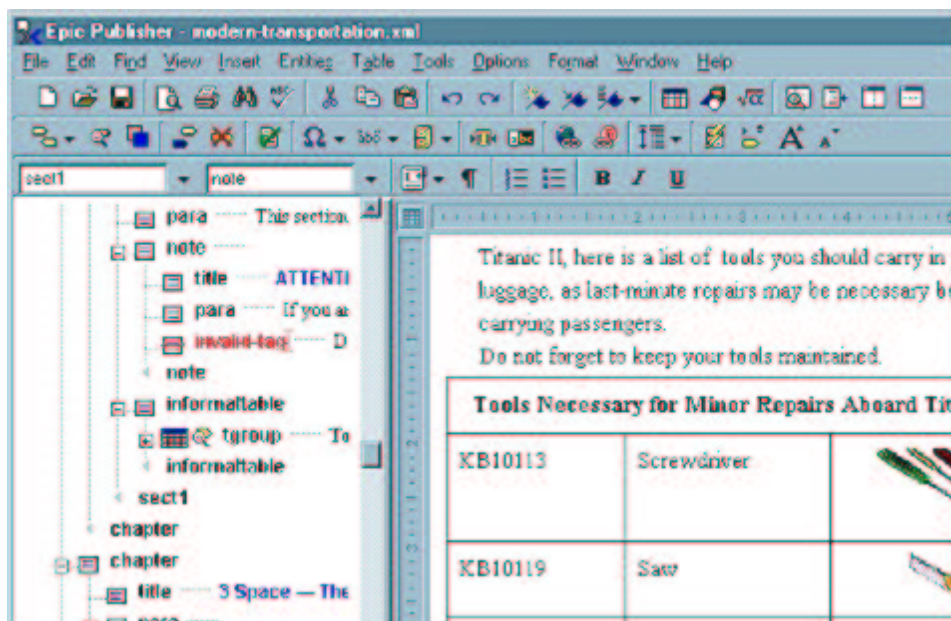
Arbortext Epic Editor is the best of a series of content editors that have attempted to get the most out of XML structure, which allowing users to see a visual representation of their output. Given that the designer understands XML and the learning content, Epic can out-perform FrameMaker as a tool for importing and creating new content. The Epic editor is very often imitated by LCMS vendors that work in structured document formats.

4.5   Corel XMetaL

Corel XMetaL is part of a suite of XML applications. It is an advanced structured editor that is relatively easy to use and highly customizable for applications based on well-known DTDs. It provides three views of an XML document: a plain text view in which you can view the underlying XML code; a tags-on view in which elements are represented as symbols in a formatted document; and a normal view that displays the formatted document and hides the markup. XMetaL supports use of cascading style sheets to control the formatted view of the document on screen.

Unlike an HTML editor, such as HoTMetaL, that works with a fixed tag set, XMetaL is meant to be used with any DTD and therefore requires customization. You will need a cascading style sheet and in most cases a set of macros for data entry for each new DTD. XMetaL supports the Windows Scripting Host, which means that you can write scripts in JScript, VBScript, Perl or Python to process XML documents or to create custom data entry interfaces.

XMetaL is intended to be integrated as a component of a broader XML solution, such as a content management system. The new Version 1.2 adds a built-in XSLT transformation engine.

4.6   Open Source Tools

In addition to the commercial offerings from vendors in the XML tools marketplace, there is a considerable body of other tools that have been produced to support SGML and XML content management by the academic and open source communities.

NOTE: This section is more technically oriented than the proceeding sections. In general, open source solutions require a more technically oriented user. To make up for this, they may contain extremely powerful features that are not available in other products at any price, let alone for free[20]
Some examples of these include:

- **Bitflux Editor** - A browser-based WYSIWYG XML editor written in JavaScript that uses XML, XSLT, and CSS for rendering. It is usable with any XML document and features tables, lists, images, special characters, clipboard, undo/redo, and easy customization.

- **Ektron eWebEditPro+XML** - A browser-based XML word processor-like editor that enables business users to apply XML to Web content. It provides a user layer between the XML tags themselves and user actions. Scripting and commands work together to control which tags the user has access to, and where the tags can be used. Business users will not realize they are working with XML tags, but instead think they are working within a set of content parameters, definitions, and/or rules. Customization is required to implement the DTD and produce valid XML, but once this is done, there is little need for further integration.

- **GenDoc**(formerly GenDiapo) - An XML editor based on a existing project, MerlotXML. It can use two kinds of plug-ins (DTD and/or action). The DTD plug-in can be used to customize the editor for a DTD, and an action plug-in can be used to publish documents in HTML or PDF format. The editor is composed of three views: tree view, attribute view for current element, and a "styled view". The aim of styled view is to show the document with a visual aspect.

- **Morphon XML-Editor** - A validating WYSIWYG XML editor that lets you create and modify XML documents in an intuitive manner. Using DTDs and CSS, the editor guarantees the integrity of your XML documents and presents them in a consistent and user-friendly way. The XML editor is bundled with the Morphon CSS Editor that can be used to customize your CSS, allowing you to change every aspect of the way the XML editor presents your document while editing. The CSS editor can also be used stand-alone to directly create CSS for the Web.

- **exchanger** - The eXchaNGeR XML browser is a browser and editor framework, written in Java, that visualizes elements in a XML document. The user can browse through and manage the visible elements in the document with external services, he or she can make changes to the content of the XML document with the built-in XML editor.

- **Arsdigita CMS** - A powerful content management system. It has a task list for production staff to track assignments and the status of current work items; a

---

[20]Many Open Source software solutions are available without acquisition cost, though the real cost of ownership may be considerably higher.

site map browser to view and organize pages and content items and determine access control to branches of the site; a standard interface for creating, editing, approving, and deploying content items; a template manager for creating, editing, and organizing presentation templates and related assets; a metadata manager for viewing and defining content types and associations; a category browser for managing a hierarchy of subject headings that may be applied to content items; and administrative and management tools for creating and editing user attributes and tracking global work flow statistics.

- **OpenLMS** - An LMS made at the Department of Geography, NTNU. The system is a fully functional LMS with support for group collaboration, file sharing, distribution of lectures, and other supporting features. It is a good tool for distributing lecture notes to groups of students, and for facilitating collaboration for groups of students and teachers.

- **Moodle** - An LMS for producing Internet-based course Web sites. It is written in PHP and is easy to install and use on Linux, Windows, and Mac OS X. It has been designed to support modern pedagogies based on social constructionist theory, and includes activity modules such as forums, resources, journals, quizzes, surveys, choices, and assignments. It has been translated into 30 languages, with more on the way. Moodle offers a free alternative to commercial software such as WebCT and Blackboard, and is being used by a growing number of universities, schools, and independent teachers for distance education or to supplement face-to-face teaching.

- And many more[21]...

4.7 Summary of Tools

The following table provides a quick reference for some of the important tools that have been discussed in this section. The following abbreviations are used in this table:

- **Word** - Microsoft Word, current XP version, some features available at additional cost.

- **FM** - Adobe FrameMaker + FM Server + Adobe Document Manager

- **ID/GL** - Adobe InDesign/GoLive combination

- **DW** - Macromedia Dreamweaver

- **AT** - Arbortext Editor

- **XM** - XMetaL

- **OS** - Open Source tools, in aggregate

---

[21]See: http://freshmeat.net search topic Learning Management

Advantage	Word	FM	ID/GL	DW	AT	XM	OS
Documentation	Y	Y	Y	Y	Y	*	*
Format Conversions	Y	Y	Y	Y	*	Y	*
LMS Integration	*	Y	Y	*	*	*	Y
Native XML	*	Y	*	Y	Y	Y	Y
Online Help	Y	Y	Y	Y	Y	*	*
Support Available	*	Y	Y	Y	*	*	*
Training Available	Y	Y	Y	Y	*	Y	*
Valid XML Support	*	Y	*	*	Y	Y	Y
Well-Formed XML	*	Y	*	Y	Y	Y	Y
WYSIWYG	Y	Y	Y	Y	*	*	*
XML training	*	Y	*	*	*	*	*

Disadvantage	Word	FM	ID/GL	DW	AT	XM	OS
Costly	*	Y	*	*	*	*	*
Extensive Prep Required	*	Y	*	*	*	Y	Y
Conversions Required	Y	*	Y	Y	*	*	*
Inefficient	Y	*	Y	Y	*	*	*
Not an XML Application	Y	*	Y	*	*	Y	Y
Steep Learning Curve	*	Y	*	*	Y	Y	Y
Training Required	*	Y	*	*	Y	Y	Y
Uncertain Future	*	*	*	*	*	Y	Y
Very Technical	*	*	*	*	Y	*	Y

Primary Output	Word	FM	ID/GL	DW	AT	XM	OS
Paper	Y	Y	*	*	*	*	*
Web	*	Y	Y	Y	*	*	*
Other Online	*	*	Y	Y	*	Y	Y
XML	*	*	*	*	Y	Y	Y

References

[1] Adobe. Adobe solutions for document generation. http://www.adobe.com/products/ server/pdfs/document_generation_wp.pdf, December 2003.

[2] Adobe. *Safeco*. http://www.adobe.com/financial/pdfs/safeco.pdf, March, 2003.

[3] Paul Daly Helen Watchorn. Word and xml: Making the 'twain meet. *Journal of XML Europe 2001*, 2001.

[4] Willaim Horton and Katherine Horton. *E-Learning Tools and Technologies*. Wiley, Indianapolis, IN, 2003.

[5] Michael Hughes. Keeping just-in-time from being way-too-little. *Performance Improvement*, 42(6):37–40, July 2003.

[6] Gerry Paille Solvig Norman Prescott Klassen John Maxwell. *The Effect of Using Structured Documents (SGML) in Instructional Design*. http://naweb.unb.ca/ proceedings/1999/paille/paille.html, February, 1999.

[7] ISO Online. International organization for standardization. *ISO Online Web Site*, 2003.

Index