

XML and Content Reuse Systems for Instructional Design Part I Introduction to XML and Repository Technology

Henry Meyerding

January 2004

Abstract

XML as a content reuse methodology¹ is a very complex and technical topic. Instructional designers who seek to discover the impacts of XML technology to the field of instructional design very often are frustrated by the technical white papers available on this subject. Those white papers usually fail to speak to the practical needs of instructional design. This three-part series is written for experienced instructional designers who wish to understand content reuse in an XML environment. The first article explains the basic underlying concepts at a non-technical level and relates those concepts to instructional design processes. The second 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.

Copyright 2004 - Henry Meyerding

All rights reserved - Neither this article nor any part may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, microfilming, and recording, or by any information storage or retrieval system, without prior permission in writing from the author.

¹A structured approach to more efficiently reuse static content in multiple deliverables.

1 Introduction



Write it once ... use it many times.

This simple statement describes the function and attraction of extensible markup language (XML) as an authoring methodology. This paper describes XML and discusses how XML can be used in an instructional design setting to manage and facilitate the definition, use, and distribution of learning objects.

In its simplest form, a learning object is some discrete information that speaks to a specific learning objective² That being said, learning objects are not simple things, but complex constructs of information, presentation and interaction. Designers traditionally have seen themselves as artisans who created unique learning tools for each new learning situation. They have been very slow to transition their thinking to a systematized approach to the development and delivery of learning. They have also spent long weary years learning how to use particular tools and will resist giving up their hard earned virtuosity, even when the tools in question are obviously a barrier to meaningful improvement.

At the same time, the enterprises for which most instructional designers work have been under increasing pressure to provide training more efficiently. Training departments, unwilling or unable to deliver substantial increases in efficiency, risk being replaced by outside contracting firms that promise to deliver these efficiencies.

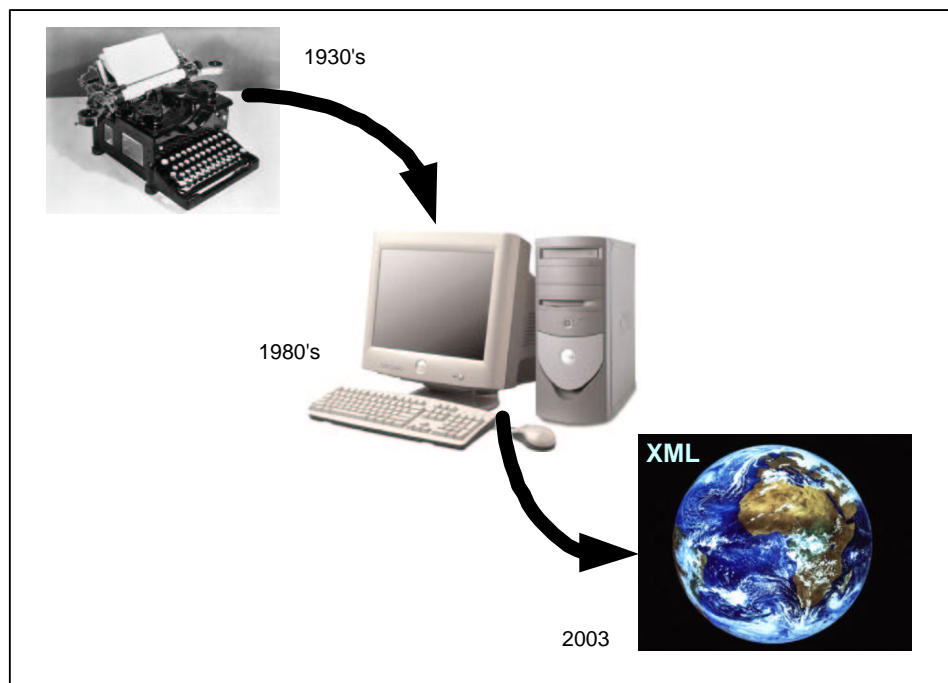
XML has become a standard means of information interchange within the computer industry. Using XML to create and manage learning objects is not a theory but has a long track record of use in the real world. It offers greater efficiency without reducing the quality of the training deliverables.

Working within an XML environment requires a change in perspective. Instead of approaching each task as the resolution to a specific obstacle to learning for a specific audience, the designer must analyze the task in a wider context. With whom does this audience share this learning requirement? How can this learning obstacle be resolved for all students? If this objective does not apply to other groups, are there components

²Direct instruction typically employs clearly articulated external learning objectives. These tend to isolate critical information and concepts, organize to-be-learned concepts into carefully ordered sequences to reflect the presumed hierarchical nature of knowledge, and employ strategies that induce differential allocation of attention and cognitive resources.[9]

within it that do apply to a wider audience? What existing training can be pulled into this task and modified to work, without affecting the quality of the learning? What other training is relevant to the content currently under development? How can these new content objects be fitted to other uses?

Some experienced designers, who are more used to routine and repetitive iterations of vast waves of training materials, may find very little in the previous paragraph that speaks to their job description. Their organizations have identified deliverable requirements, methodologies and audiences for them and the designers are charged with producing the required quantity of training that meets a relatively low quality standard. The attractive feature of XML to these designers is that it offers a way to respond to practically impossible demands for training with substantially less drudgery, thus allowing designers to build at a higher standard of quality, which is (or should be) always on the nice-to-have list.



Instead of being a creative artist fashioning unique responses to specific learning requirements, the designer becomes a production professional who analyzes the learning needs of a specific group as those needs relate to the generalized requirements of the entire learning community. The importance of designers understanding the capabilities and rationale behind the content reuse system cannot be stated strongly enough.

The change over into an XML content development and production environment really represents the same quantum leap in capability as was achieved by replacing typewriters with word processing on computers. Managers must be evangelists of XML technology's liberating capabilities. They also must be zealous in training their staffs to understand these capabilities. There never has been a system so good that it could not be rendered totally ineffective by resistant participants.

One of the incidental benefits of operating in an XML learning object environment is that designers are exposed to content created by other designers much more than in traditional project environments. Properly managed, the specialized understanding of different teams is more effectively shared and the quality of the output is increased[7].

1.1 Terminology

Before launching into a discussion of XML[1] in more depth, it is important to understand some of the terminology that will be used throughout this paper.

Attribute - The characteristic of an XML element that defines the content. Example: If the elements are class, type, and color; corresponding attributes might be toy, rubber ball, and red.

Chunking - The process by which legacy content is tagged for inclusion in the content database.

Content - Content is information. It may take the form of text, graphics, audio, or video.

Database - A hierarchical distribution of data arranged in relationships that provides quick access to information of interest.

Document - Strictly speaking, when working in XML there is only one super document that contains all the content. This content fits into a common structure. We extract pieces of this super document and publish it as document instances, which may be either static or dynamic.

Dynamic Instance - When publishing a document instance, it may contain information that changes continuously. In defining the document publication instance, it may be desirable that every time a user opens the instance they see the most recently updated information. Contrast with static instance.

Element - An XML element is a definition for content. Any piece of content may be defined by one or several elements. Example: class, type, and color.

HTML - Hypertext Markup Language - developed from SGML as a means of conveying information on the web.

Learning Object - A functional component of training curriculum; a building block. Each learning object generally addresses a specific learning goal. Just how specific a goal is defined varies from system to system.

Legacy Content - Content, usually in electronic form, such as text, graphics, audio or video that has been developed outside of an XML content environment. Legacy content often resides in short-lived proprietary formats, which make reuse or conversion problematical.

Metalanguage - The language that is used to talk about (expressions of) another language, the object language. XML contains and identifies content, but the XML is not the content.

Parse - To divide into components from a larger set based upon some identifying feature or content.

SCORM - Sharable Courseware Object Reference Model - A set of specifications for developing, packaging and delivering high-quality education and training materials whenever and wherever they are needed.

Static Instance - When publishing a document instance, it may serve as a standard or reference. In defining the document publication instance, it may be desirable that users see a single, unchanging document, until any changes have been approved by a ruling/governing body. Contrast with dynamic instance.

SGML - Standard Generalized Markup Language - the international standard metalanguage for text markup systems³.

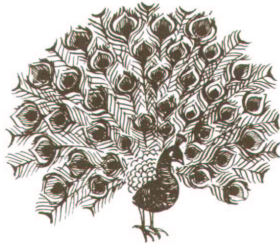
Taxonomy - a system for naming and organizing things, into groups which share similar qualities.

XML - eXtensible Markup Language - developed as a more manageable subset of SGML.

The next section introduces XML and discusses some of the features of XML that make it particularly appropriate for learning object development and learning content reuse.

³ISO 8879

2 XML Basics



The loftier the building, the deeper must the foundation be laid. – Thomas Kempis.

What is XML? The typical definition of eXtensible Markup Language (XML):

XML is a new World Wide Web Consortium (W3C⁴) specification. XML is a pared-down version of SGML, designed especially for Web documents. It enables developers to create their own customized tags to provide functionality not available through HTML.[5]

This definition provides a lot of information about XML, but it obviously was written by someone who lacks an understanding of XML and its capabilities. XML was designed as a database metalanguage[4]. It was designed as a means to structure content so it could be put into and be retrieved from a database in a form that was useful for content reuse. Information content can be text, graphics, audio, video, or complex constructs of all these learning components.

The principal difference between XML and HTML is that the former uses "smart tags." Smart tags convey information about the content they contain. Because of this, you can use the structure you create to put your content into an easily retrievable form. One exciting aspect of XML is the ability to define your content your way, creating custom tags for different kinds of instructional objects such as objectives, test questions, feedback and other common components of the training content[3].

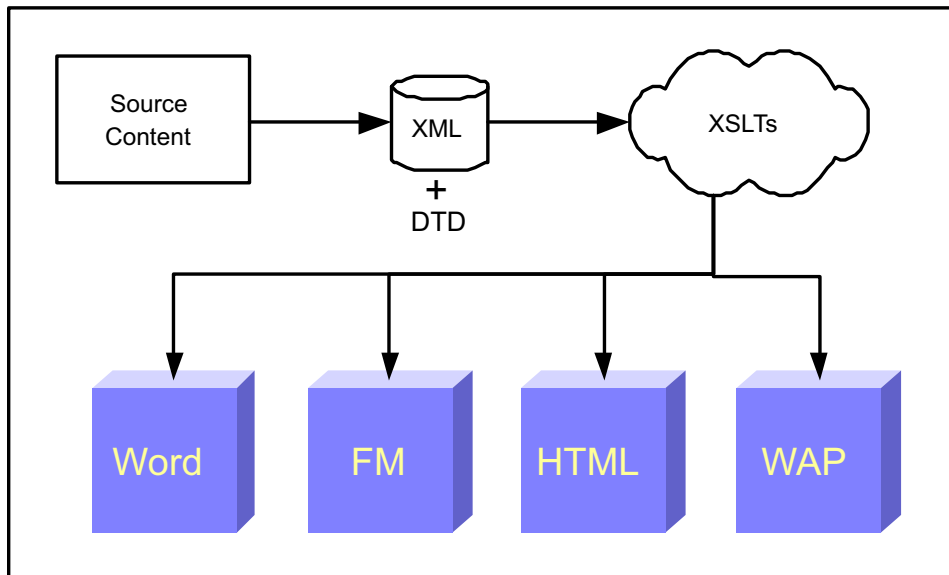
Your content management system reads the smart tags and parses your content into useful chunks that you can assemble into subsequent documents. When you need the same (or similar) content, you construct a query of your content database. You then review the resulting content and if it matches your current use, you use it. If you find nothing useful, you add new content for your current document – and future use.

By using smart tags, it is also possible to define very specific criteria for making recursive changes. A branding change, which might take weeks to implement across an entire curriculum can be implemented in minutes. Editorial and style changes can be very exactly implemented in the precise circumstances defined by the editors. Legal reviews can be conducted on exemplar text, which is then recursively edited throughout the content library.

⁴The same people who brought you SGML and HTML.

Each of the content elements is consistently tagged so that it fits together with other elements to form consistent document instances. In other words, each content object contains an introduction, main matter, illustrations (if any), conclusion and transitions. Several content objects are aggregated to form a document instance. Given a different use, it may be necessary to slightly modify some of the introductory or transitional materials. You also can structure your learning objects so that they can be specifically relevant to different user groups or to audiences of different aptitudes. The next document instance you require may contain the same content objects, which have been modified, just a little here and there, to meet the requirements of the different audience.

The process of defining these structural components is analogous to creation of animation cells from layers. Each layer contains a different quality of information pertaining to the same object. These objects are then added together to produce an instance of learning delivery.



There are three main components of XML objects with which instructional designers need to be concerned: XML, DTD and XSLT. The XML file contains the content, just the content and only the content. The Document Type Definition (DTD) specifies how the XML is structured. The XML Style sheet (XSLT) contains all the formatting information for the output document instance.

2.1 Document Type Definitions (DTD)

Most XML documents (and all SGML documents) refer to an external document type definition (DTD). The DTD provides a list of the elements, attributes, comments, notes, and entities contained in the document. It also indicates their relationship to

one another within the document. In other words, a DTD is the grammar of an XML document[8]. Without a DTD (or schema, see below), all you have is data. The DTD is rather like the header row in a table that identifies what lies below it. It does not itself contain any data, but makes the data that follows useful.

DTDs are complex and difficult to create from scratch. Luckily, there are thousands of open source DTDs from which to choose. Most of them are related to a specific industry or enterprise. Unless you have unlimited time and capital, there probably is no reason to even contemplate making a custom DTD that models your core organizational elements. The most widely distributed DTD is DocBook[6], which originally was developed for documenting computer software. Many popular DTDs, such as the Telecom Interchange Markup (TIM) are derivatives of DocBook. Because XML is extensible, you can begin with a DTD that meets your core requirements and then make minor modifications and additions to it at a later date without invalidating all your XML.

The DTD defines the taxonomy that you will use to parse information about the subject matter into re-usable form[2].

2.1.1 XML Schema

Considerable confusion exists about XML Schema. Simply put, XML Schema represent another, more flexible way to accomplish the same functions as DTDs. Just like the DTD, the XML schema is an external document referenced from the XML document. In terms of learning objects, XML Schema do very little additional, useful things, as compared with DTDs.

Where XML Schema do come into their own are in validation of numerical data. For example, if you are passing instructions to a machine that cuts precision parts from blocks of steel in XML format, it would be handy to have that data automatically validated before it is handed to the machine, since accidentally sending the wrong numbers can ruin not only the production piece but the very expensive machine as well.

It also is true that most of the document-handling software has been developed in an XML-with-DTD environment. If, in the future, as some forecast, XML Schema replace DTDs, there will be simple tools to convert DTDs to equivalently behaving XML Schema. For the time being, in regard to learning content management, it is not necessary to pay any attention to schema[10].

2.2 XSLT

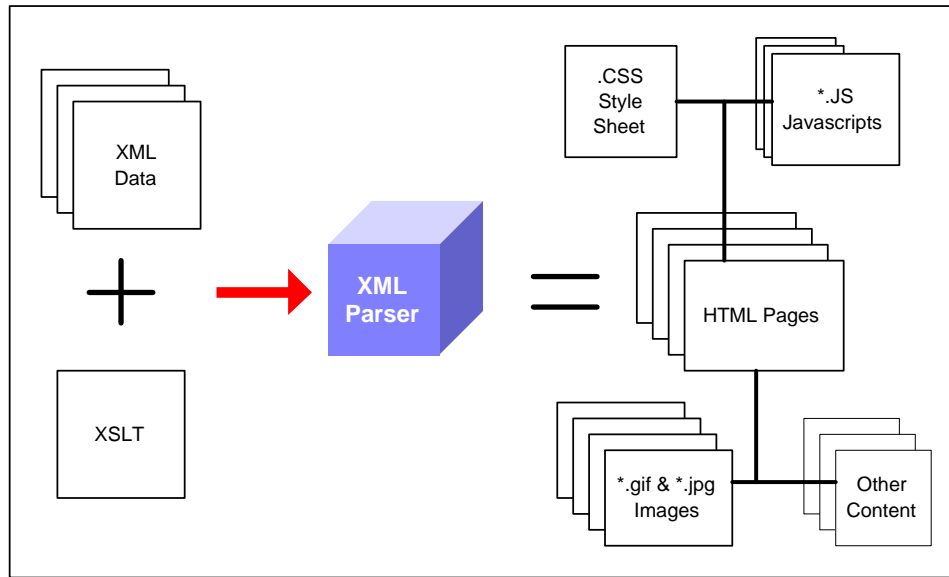
XSLTs are the method used to transform raw XML data into useful content[11] - creating a Word document or a FrameMaker document from the same source files is accomplished by applying two different XSLTs to the same XML data files. They are referred to as style sheets, but they are not style sheets in the same way as a cascading style sheet defines the look of an HTML document instance.

It is important to note that authoring in XML divorces the content from the format. The reason for this is quite simple: consistency. When you make a change to the source content of a series of deliverables, you should be able to produce those new versions with a minimum of intervention; ideally with no intervention. This is accomplished by creating an XSLT for each document instance type and then processing content through that common XSLT for all versions of that output deliverable type. The same revised content can be processed through XSLTs to produce an update for four different courses, which may be either web-based or paper-based.

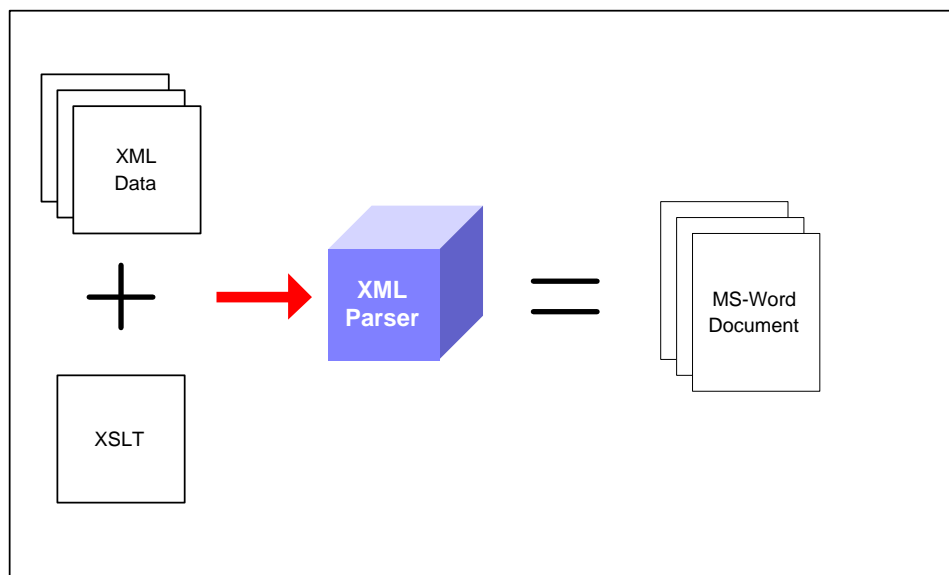
This is a departure from traditional word processing and publication tools, which merge these two operations into a single work flow. Some tools used to author XML, such as FrameMaker, allow the designer the illusion of authoring content with format, but all formatting is defined in the XSLT, which authors very seldom modify. Rather than spending endless hours fiddling with documents to make the styles come out right, developers work with the content to make that right. When the words and graphics are right, they are done.

XML is data and contains no formatting. However, because it is structured data, the formatting can be applied to it programmatically. The XML data and the XSLT are parsed⁵ to become a web site. The website contains the HTML pages, images, javascripts, style sheets and other content associated with a full-featured Web site. The same content can be parsed with a different XSLT to result in WAP fully featured web content.

⁵There are many kinds of parsing engines, come written in C++, Perl, Python and Java.

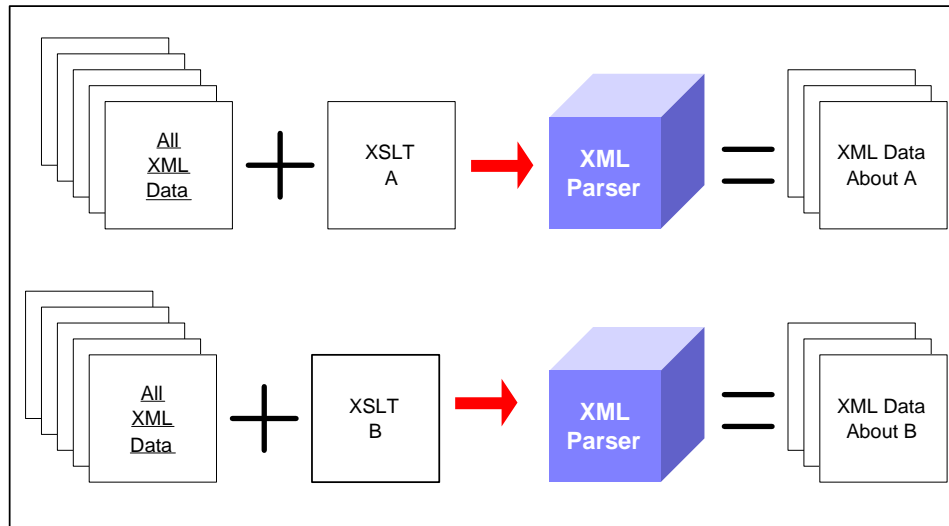


An XSLT also can convert the same content to an Microsoft Word document (or any other document format that is desired).



Lastly, an XSLT can be used as an intelligent query to result in the XML content that conforms to a specific set of rules. These rules can be very complex and can be a more efficient way to query data than a direct Structured Query Language (SQL) query to the database.

You may have all your content about a subject in XML format and use an XSLT to separate out from the total content only those passages which conform to specific criteria.



2.3 What about ...?

The following sections describe several initiatives and protocols that are related to XML.

2.3.1 SCORM

The Sharable Courseware Object Reference Model (SCORM) is very prominent in the marketing literature of a lot of content management vendors. To hear these vendors talk, if your XML is not SCORM compliant, there is something wrong with you. SCORM is a set of standards, implemented in XML, which passes instructions to a Learning Management System (LMS). It is analogous to an application program interface (API)⁶.

If your organization selects a SCORM-compliant LMS, then your repository should contain an XSLT to present your content with additional SCORM elements that enable

⁶APIs are shortcuts for doing complicated things. For example, programmer A writes an accounting program and wants to allow programmer B to write a program that queries the accounting program for specific kinds of information. Programmer A does not want programmer B to know exactly how his accounting system works, so he writes an API that allows any application to send the accounting system a very specifically worded question, to which it will reply with the desired information. Programmer B now only needs to know how to ask the question and not how the account system handles the operation of answering it.

the LMS to know how to deal with it. The nice part about SCORM is that it is an open standard. If you decide that this LMS does not meet your needs, you will be able to select another SCORM-compliant LMS⁷ and only have to tweak your repository here and there to make it all work correctly.

2.3.2 SOAP

The simple object access protocol (SOAP)[12] is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

Plain language version: You own an Internet-connected soft drink machine. Once a day, you want your soft drink machine to send information to the inventory management package. You then want the inventory management package to talk to your payroll system so that the salesman for the territory gets paid the right amount. Either you buy all these systems from the same vendor, or you buy each one from the lowest bidder and make sure they all talk SOAP to one another. You then only need one programmer to make it all work instead of a staff of fifteen programmers to write it from scratch.

Although usually associated with Microsoft, it is actually maintained by a consortium of companies, of which Microsoft is one.

2.3.3 .NET

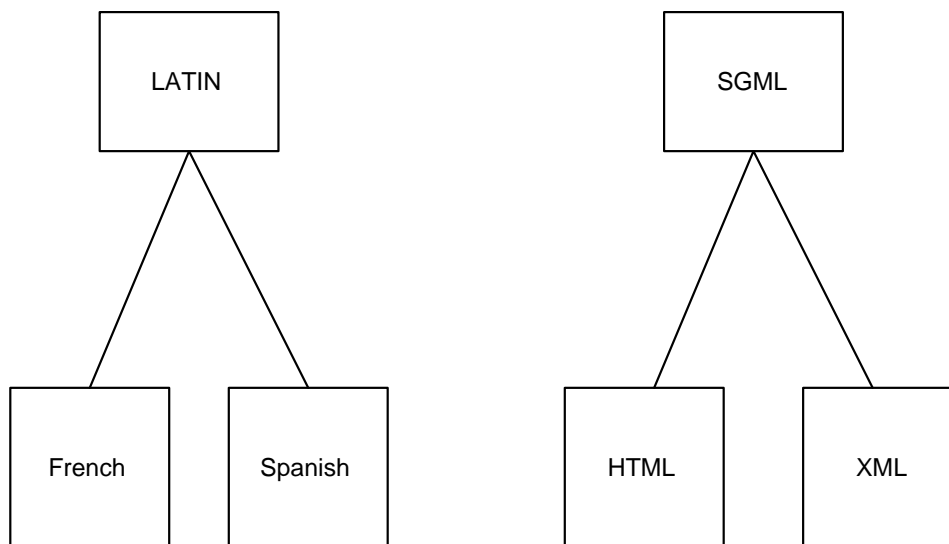
Microsoft's "dot net" (.NET) is a "software platform". It's a proprietary language-neutral environment for writing programs that easily and securely inter-operate. Rather than targeting a particular hardware/OS combination, programs instead target ".NET", and run wherever .NET is implemented. In this way, it is just like java. .NET is also the collective name given to various bits of software built upon the .NET platform. These are both products (Visual Studio.NET and Windows.NET Server, for instance) and services (such as Passport, HailStorm, and so on).

.NET uses XML in many instances, especially when complex data is transferred. It is a case of using what works best to accomplish some tasks. Other than this, it has very little to do with XML, despite the mounds of marketing that issue forth from Redmond which try to make .NET and XML seem synonymous.

⁷Your mileage may vary. How SCORM-compliant is a piece of software? Unfortunately, it is often only as compliant as the maker says it is, without any testing or third party verification.

2.3.4 SGML

Standard Generalized Markup Language (SGML) is the root language for many other languages, such as HTML and XML. It has been around for a long time⁸ and is still is a universal tool, both platform and system independent, for describing text. SGML is rather like Latin – the language from which other languages have been derived. Like Latin, it is more difficult to use in a contemporary context than are its children. Also, there are many more tools being developed every day to do things in XML. This is not true of SGML anymore.



XML owes much of its functionality and usefulness to the lessons learned by people using SGML.

⁸1968

References

- [1] Robert Eckstein With Michel Casabianca. *XML Pocket Reference*. O'Reilly, Sebastopol, CA, 2001.
- [2] W. Scott Means Elliotte Rusty Harold. *XML in a Nutshell*. O'Reilly, Sebastopol, CA, 2002.
- [3] Willaim Horton and Katherine Horton. *E-Learning Tools and Technologies*. Wiley, Indianapolis, IN, 2003.
- [4] Jan Kampherbeek. *Crash Course in XML*. <http://www.spiderpro.com/bu/buxmlm001.html>, June, 2001.
- [5] Harry Newton. *Newton's Teelcom Dictionary, 17th edition*. CMP Books, New York, NY, 2001.
- [6] Leonard Mueller Norman Walsh. *Docbook, the Definitive Guide*. O'Reilly, Sebastopol, CA, 1999.
- [7] George M. Piskurich. *Rapid Instructional Design*. Jossey-Bass/Pfeiffer (Wily), San Francisco, CA, 2000.
- [8] Erik T. Ray. *Learning XML*. O'Reilly, Sebastopol, CA, 2001.
- [9] Charles M. Reigeluth. *Instruction-design Theories and Models, Volume II*. Lawrence Erlbaum Associates, Mahwah, NJ, 1999.
- [10] C. M. Sperberg-McQueen and Henry Thompson. *XML Schema*. <http://www.w3.org/XML/Schema>, April, 2000.
- [11] Doug Tidwell. *XSLT*. O'Reilly, Sebastopol, CA, 2001.
- [12] W3C. *Simple Object Access Protocol (SOAP) 1.1, W3C Note*. <http://www.w3.org/TR/SOAP/>, May, 2000.