Improving Web Based Training using an XML content base

Simon Wiest, Andreas Zell
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen, Köstlinstraße 6, 72074 Tübingen, Germany
{wiest, zell}@informatik.uni-tuebingen.de

Abstract

In this paper we point out the benefits of using structured content in Web Based Training (WBT) scenarios. Using an XML compliant subset of the DocBook document type definition –enriched with semantics for interactive hypermedia and educational elements– we present a complete publishing workflow for educational online material. Original content is authored in an SGML/XML editor, then processed into a set of XML files containing course structure and contents. Finally a web server extension written in Java (servlet) renders this XML content on-the-fly into HTML pages according to learner preferences, desired target media or current educational situation. All user requests and their parameters are logged into a relational database. This allows the collection and sharing of comparable data on applicability and effectiveness of the content and helps to improve the quality of future online courses.

1. Introduction

Interactive computer based training (CBT) has experienced a tremendous growth over the last few years. However, the rapidly growing demand for interactive CBT revealed that the traditional content creation workflow has serious shortcomings that make its large-scale application difficult:

Reuse and exchange of content modules are hard to implement, because most learning environments use proprietary authoring tools and data formats. It soon turned out that also HTML, the *lingua franca* of the Internet, lacks functionality in fulfilling the needs of on-line educators, e.g. no dedicated semantics for online exercises, no support for complex equation rendering, difficult reformatting of HTML content that is to be reused in new contexts.

This paper presents a workflow that tries to tackle some of these problems using an non-proprietary content structure and introduces a web server extension that enables HTML browsers to access the features made possible by XML content.

2. Benefits of structuring educational content

The key idea behind structuring educational material is to decouple style from content. Storing course contents in elements that reflect its semantics (e.g. introduction, example, quiz, ...) allows the reuse of data in different situations and formats - adapted to learner preferences, target media or purposes. All groups concerned with the online learning process benefit from the use of structured documents:

Authors

- Authors can concentrate on content not (necessarily) on style and formatting
- Structural integrity can be checked/enforced by an authoring application (e.g. hyperlinks can be validated automatically)
- Hyperlinks can point to document subcomponents, allowing fine-grained cross-references.
- Indexes, summaries, glossaries etc. can be generated automatically
- Content modules can be reused in other contexts (e.g. related courses, in a guided tour, in different skill levels)

Publishers / Internet Service Providers

- Contents are well machine-processable (i.e. well storable in databases to facilitate content administration and improvement of retrieval performance)
- Crossmedia publishing is possible from one, single source (e.g. online/offline versions, printed material). The concept of *viewgroups* presented in [12] addresses this idea.
- Structured content allows the attachment of meta data not only to the complete document but also to its subcomponents.
- Versioning is facilitated.

Learners

- Mandatory structure leverages navigating the course content.
- Advanced search and retrieval of content components.
- New features like multi-representation content become feasible (e.g. the same content in both visual and textual representation. [14] suggests a Presentation Markup Language.)

Researchers / Quality Assurance

- Enhanced possibilities for learner tracking and usage evaluation allow the collection and sharing of comparable data on applicability and effectiveness of the content and help to improve its quality [4].
- Flexible rendering allows experiments with screen designs, navigation concepts etc.

3. Existing applications of SGML/XML in the educational field

Like its predecessor, SGML [10], XML [5] is a scheme to define new languages. It does not specify what kind of application data is stored in a file by itself but rather provides a framework to define purpose-depended languages that follow one common syntax. This is done in a document type definition (DTD) that describes the overall structure and the components of generic data file. There have been already different suggestions to apply SGML, XML or at least structured documents to educational purposes:

The IEEE Learning Technology Standards Committee proposes a learning object meta data standard (LOM) [9] to facilitate the retrieval, reuse and exchange of educational content. Learning objects range from small components like a single sentence, a figure or a Java applet up to complete on-line courses. LOM contains information on technical, didactical and legal issues of educational content and provides support for versioning. LOM does not standardize the actual internal data structure of learning content.

The Tutorial Markup Language [6] provides a language to describe several classes of online exercises like multiple-choice tests, drag & drop puzzles etc. Other structured file formats [3], [11] pursuit the same goal but currently don’t use neither SGML nor XML.

The Synchronized Multimedia Integration Language (SMIL) [8] allows an XML compliant specification of both time-line based and event driven multimedia presentations. Supported by widely distributed player applications, it definitely will be used in educational context in the future although it does not contain dedicated didactic semantics.

4. Our implementation

At our department, we offer several on-line courses on computer science topics to our students. This material consists of some 200 linked HTML pages per course, enriched with additional interactive elements like Shockwave animations, Flash movies, and Java applets. The courses were created from legacy content in Adobe FrameMaker format as a starting point. Dealing with mathematical and statistical subjects the courses contain over 500 equations. A typical page is shown in Fig. 1.

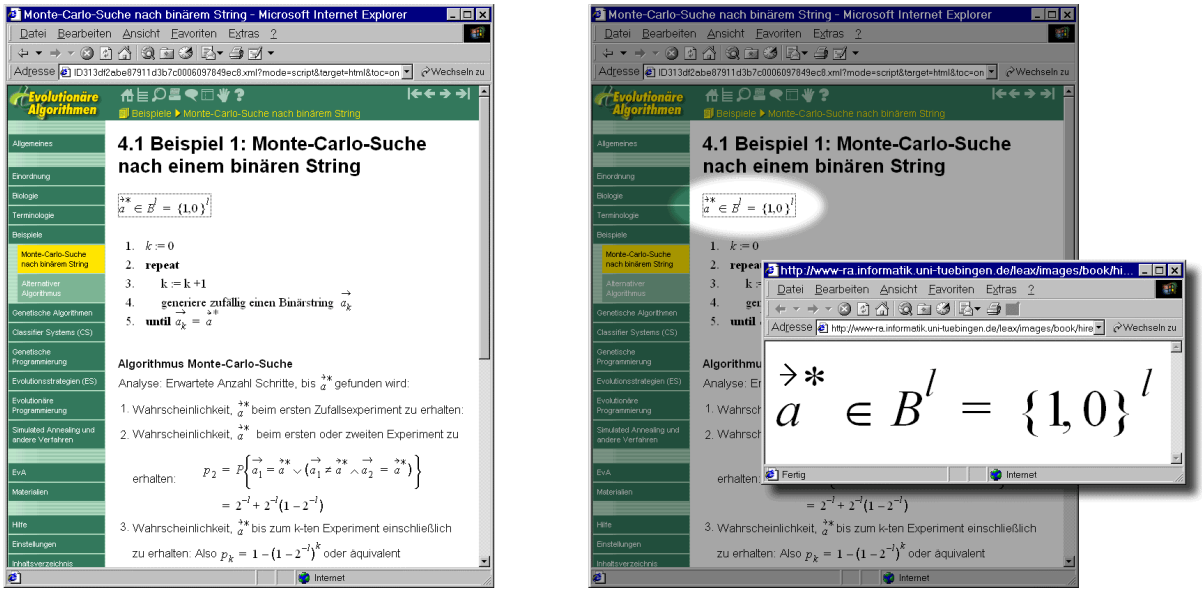

Fig. 1. Left: Typical page from online course “Evolutionary Algorithms”. Note the high number of equations and symbols in the main content area. Small symbols, especially indices, are hard to read on-screen and do not print very well. Right: Using our improved system, learners can click on any equation to open a high resolution version in a separate windows. An additional print view offers also a high quality hardcopy option.

Although FrameMaker ships with HTML/XML export functionality, files generated by the HTML export function needed manual clean-up which took one person about 1-2 days to complete for each new release of a course. Additionally, an evaluation among student users turned out that the poor display quality of equations was a major annoyance to learners.

From our previous experience we also could identify two main usage scenarios for our online content: the *tutorial mode*, in which learners tend to navigate through the course linearly and slowly paced and the *script mode*, in which users want to look up small amounts of information like definitions, proofs or concepts quickly. Because both modes are related representations of the same content, we wanted to keep maintaining course contents simple and generate both versions from one single data source. Because interface design for WBT is still a young discipline we also realized that there might be a need for additional modes in the future. Ideally, a learning environment would therefore apply formatting and user preferences to the desired content not until the moment of an incoming learner’s request.

We therefore came up with a new publishing workflow that consists of six steps:

1. Choosing/creating an appropriate document type definition (DTD).
2. Structuring a complete course according to the DTD in an authoring environment.
3. Converting the SGML content into XML and preparing it to be served on the WWW.
4. Creation of HTML page templates and navigational controls.
5. Developing a servlet that converts XML to HTML on-the-fly.
6. Tracking content usage into a relational database.

4.1. Document Type Definition (DTD)

In the search for an appropriate document type definition we realized that while there are quite a few standard DTDs for technical documentation like DocBook [13], our course content structure had also to support didactic and hypermedia elements. We decided to enrich an XML-compliant, light-weight variant of the DocBook DTD with additional elements and attributes that cater educational needs, different media types and hyperlink capabilities.

Educational elements: According to the idea of learning objects proposed in [9], educational content can be decomposed into subcomponents of different aggregation level:

Table 1 LOM aggregation levels

Level	Example
0	“content atoms” like single sentences, images or raw media files
1	A collection of level 0 objects like a single HTML page with some embedded pictures
2	A collection of level 1 objects like a linked net of HTML pages with an common index page
3	A collection of level 2 objects like a complete course

We classified the elements inherited from DocBook into the given aggregation levels appropriately. All level 0 objects were assigned a 32digit hexadecimal identifier attribute `id`, unique in time and space. All level 1 objects also got an attribute `time`, which reflects the assumed learner’s time to complete this object. A typical level 1 object is a short quiz which consists of a question and an answer. Further educational elements are `experiment`, `procedure`, `definition`, `algorithm`, `proof`, `example`, `important`, `note`, `aims`, `didactical`.

Media type support: We added several elements for different media types with appropriate attributes specific to the media type. Supported media elements are `graphic`, `applet`, `equation`, `shockwave`, `flash`, `quicktime`.

Hypertext support: Hyperlinks are defined according to a subset of the the W3C XLink working draft of July, 26 1999 [7].

4.2. Authoring environment

We chose FrameMaker+SGML as XML editor because it combines the ease of use of a modern word processor (especially equation handling) with the structural functionality of an XML editor. To track down learning objects during the following processing steps, each level 0 object was assigned a globally unique identifier. Once the identifiers are assigned, they will be present in FrameMaker+SGML files as well as in exported SGML documents. It is important to note that even though contents are kept in the FrameMaker+SGML file format, this format represents nothing more than a wrapper around a DTD compliant structure.

4.3. Converting and preparing XML content

The next step is to export the FrameMaker contents into an SGML file using built-in export functionality. Image representations of embedded equations are also generated in this step.

A common problem creating scientific content for the WWW is to display mathematical equations. While some browsers already feature limited MathML rendering capabilities it is common practice to embed equations as bitmap images, e.g. as GIF or PNG files. This one-way conversion complicates later-on modifications for the author and results in poor image quality of the content (especially when hardcopied. We decided to keep equations in this format within FrameMaker but to convert them in GIFs during SGML export. By exporting the course contents twice with different resolution (72dpi, 300dpi), we get two versions of the same equation. This allows features like zooming into equations that are hard to read (see Fig. 1 right) or high quality hardcopy options.

In a final step the *course description*, a hierarchical table-of-contents structure is extracted from the SGML document to serve as website structure. Course description, course content (stored in a set of XML files) and equation images are now copied onto the web server file system, ready to be served.

The complete conversion procedure is depicted in Fig. 2. Export and equation rendering of a 200 page course on a PII 350 MHz Windows PC takes roughly 10 minutes for the 72dpi version (40 minutes for 300dpi) and does not require any human interaction.

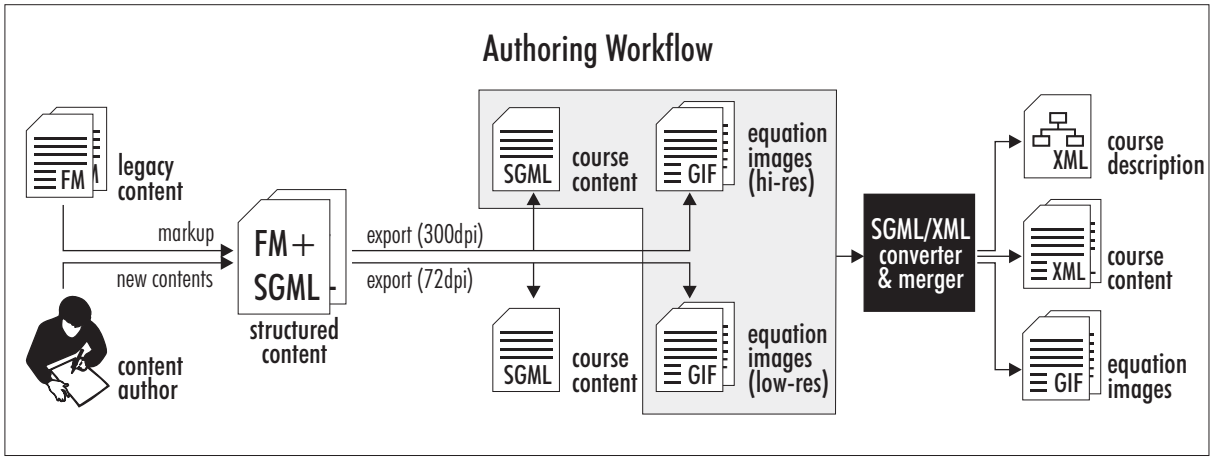

Fig. 2. Authoring workflow. Content creation takes place inside the SGML editor (like FrameMaker+SGML). In two passes, SGML versions and two variants of equation graphics are exported. Finally these file get converted to XML and prepared for being served by our servlet-based system.

4.4. Page templates and navigational controls

Page templates were designed in a HTML editor, containing placeholders for dynamic content. These placeholders get replaced by applying several XSLT transformations [17] using an open source XSLT formatter [1] to the requested course content file. XSL style sheets exist for several target media (on-line, print) and usage modes (tutorial and script mode). Fig. 3 shows three examples of different modes and target media rendering. The globally unique learning object identifiers are preserved during conversion, allowing web browsers to make use of this information. We are working on a smart, fine-grained annotation system that works at the subcomponent level within HTML pages.

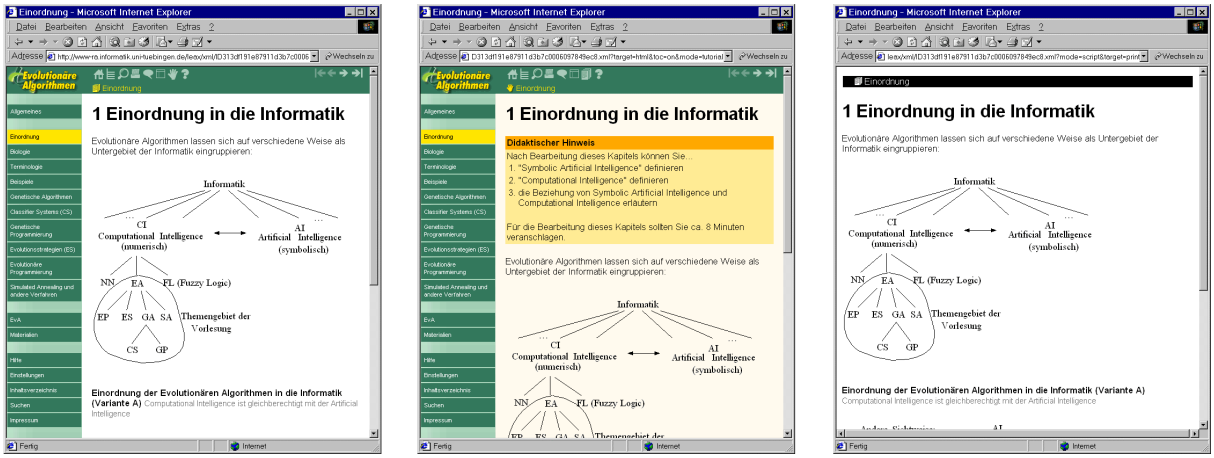

Fig. 3. The same XML content rendered for different modes/targets. Left: Script mode for on-screen usage. Middle: Tutorial mode for on-screen usage. Note the box containing learning goals in the middle of the main content area. Right: Script mode for printout. It has higher resolution and contrasts that enhance printability. Navigational elements are suppressed because they don't make sense on printed paper.

4.5. Serving content

Because most browsers currently provide only limited XML support, the course contents are server-side converted from XML to HTML. Additionally, this allows adaptation of the contents to a specific user's learning context (e.g. formatting preferences, target media, usage mode) at this point.

To bridge the gap between web server and XML content, we developed a *S*ervlet *I*nterface for *O*nline *U*ser-adapted *X*ML (SIOUX). Servlets [15] are plug-in extensions written in Java that add functionality to a HTTP daemon. To serve a request, SIOUX

a) reads in the requested XML file,
b) applies a XSL transformation to the target medium and mode,
c) inserts the content into a page template,
d) resolves server-side includes and other placeholders,
e) returns the HTML page to the learner's client and
f) logs user id, URL and learning context into a relational database.

Our setup builds on an Apache HTTP daemon [1] and the MySQL RDBMS [16] running on an RS/6000 workstation. Servlet support is provided by the JServ servlet engine, connecting to the SIOUX servlet on a Windows PC (Windows NT4.0, Pentium II-350, 256 MB, Sun Java Runtime Environment 1.2.2). The response times vary between 0.2-1.0 seconds per page request depending on complexity of the XSLT style sheets. This is acceptable compared to the 1-5 minutes it takes a learner to work through the page contents. Fig. 4 shows how a request is handled by Apache/SIOUX and which resources are used.

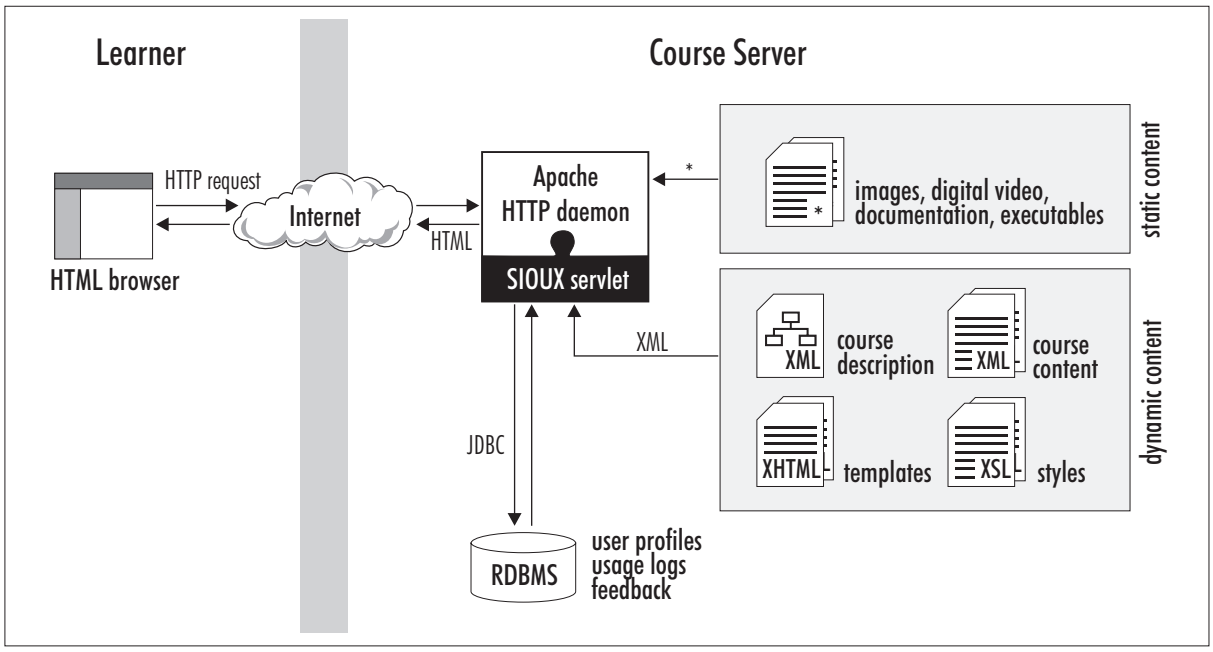

Fig. 4. Resources used within the SIOUX architecture. The SIOUX servlet generates dynamic content from XML structure, content, style and template files. Static content is handled by the Apache server without SIOUX intervention.

4.6. Tracking content usage

The system contains a learner database that allows restricting access to course contents, personalized tracking of content usage and persistent storage of user preferences. After a request has been served, the user's ID, the requested URL and all session parameters (e.g. current mode, target media, individual interface settings, date and time of access, referring URL, agent, ...) are logged into a relational database.

5. Conclusion

We presented how structuring content supports the online learning process in creating, deploying and using course content. Examples have been given, how existing DTDs can be modified to address the specific needs of online educators. Using this XML compliant DTD offers dedicated semantics needed in didactical processes while avoiding getting trapped in a proprietary format. Publishing time of new course releases could be cut down from 1-2 days of manual labor to some minutes batch processing time.

Using one single source which is varied on-thy-fly avoids complicated and error prone maintenance of several parallel versions. Rendering is acceptable fast for online learning purposes. Several modes (tutorial, script) match the main usage scenarios closer than a single all-purpose version. The handling of equations has been improved by providing high resolution versions for screen and printout. The equation itself remains in a editable textual representation which might use MathML in the near future.

Preserving learning object identifiers in the resulting HTML pages makes way for applications like fine-grained annotation mechanisms on sub-page detail level.

The flexibility in displaying content in different flavors provides also a test bed for evaluation of different user interfaces or navigation. Equally important is the ability to measure the usage of these experimental variations. While it is obvious that web interface design plays an important role in online education process, it is arguable which is the *best* way. Due to extended tracking possibilities we can offer an approach that helps to collect hard facts. In the likely case we'll learn that there is no single ideal interface for *all* learners, the presented workflow is already powerful enough to serve highly personalized educational content. In any way, precise usage data will help WBT authors to understand how students are using their course contents as well as web interface designers to improve their work based on learners acceptance of interface innovations.

Acknowledgements

The site description file format has been originally developed by Igor Fischer. The research described in this papers was conducted within the project "Bioinform@tik", funded by the state of Baden-Württemberg and the Deutsche Telekom AG.

References

[1] Apache XML Project, "Xalan-J XSLT," http://xml.apache.org/xalan/index.html

[2] Apache Software Foundation, "Apache HTTP Server Project," http://www.apache.org/httpd.html

[3] S. Arneil, M. Holmes, and H. Street, "Hot Potatoes," University of Victoria Language Centre, http://web.uvic.ca/hrd/halfbaked/

[4] B. Barquero, U. Creß, F.W. Hesse, "BioInform@tik. Evaluation der multimedialen Lehrveranstaltungen im Sommersemester 99," internal report, Deutsches Institut für Fernstudienforschung an der Universität Tübingen, University of Tübingen, Tübingen, 1999.

[5] T. Bray, J. Paoli, and C.M. Sperberg-McQuee, eds., "Extensible Markup Language (XML) 1.0," W3C Recommendation 10-Feb-1998, http://www.w3.org/TR/1998/REC-xml-19980210.html

[6] D. Brickley, "Tutorial Markup Language (TML)," Institute for Learning and Research Technology, University of Bristol, http://www.ilrt.bris.ac.uk/netquest/about/lang/

[7] S. DeRose, D. Orchard, and B. Trafford, eds., "XML Linking Language (XLink)," World Wide Web Consortium Working Draft 26-Jul-1999, http://www.w3.org/1999/07/WD-xlink-19990726

[8] P. Hoschka, ed., "Synchronized Multimedia Integration Language (SMIL) 1.0," W3C Recommendation 15-June-1998, http://www.w3.org/TR/REC-smil/

[9] "Draft Standard for Learning Object Metadata," IEEE Standards Department, Learning Technology Standards Committee, Piscataway, 2000. (IEEE P1484.12/D4.0).

[10] "ISO 8879: Information processing – Text and office systems – Standard Generalized Markup Language (SGML)," International Organization for Standardization (ISO), Genf, 1986.

[11] R. Kraußе, "Exercise Format," TU Dresden, http://linus.psych.tu-dresden.de/Stupla/ef/

[12] H. Müller, J. Deponte, "Distributed Multimedial Presentation and Conferencing," Proc. Workshop "Die Virtuelle Wissensfabrik", GMD, St. Augustin, 1999.

[13] Organization for the Advancement of Structured Information Standards (OASIS), "The DocBook DTD," http://www.oasis-open/docbook, 1998.

[14] A. Ram et al., "PML: Adding Flexibility to Multimedia Presentations," *IEEE Multimedia*, Vol. 6, No. 2, April 1999, 40-51.

[15] Sun Microsystems Inc., "The Java Servlet Specification v2.2,", December 1999, http://java.sun.com/products/servlet/

[16] T.c.X, DataConsultAB, "MySQL," http://www.mysql.com

[17] World Wide Web Consortium (W3C), "Extensible Stylesheet Language (XSL) Version 1.0," W3C Working Draft 27-Mar-2000, http://www.w3.org/TR/xsl