Extreme Markup Languages 2002

Montréal, Québec
August 6-9, 2002

A Decade of DTDs and SGML in Scholarly Publishing
What Have We learned?

Bruce Rosenblum
Inera Incorporated

Irina Golfman
Inera Incorporated

Abstract

With the growth of computer use, the invention of the World Wide Web, and the increased pace of scientific research, publishers realized that making research available electronically was vital to both researchers as well as their own business interests.

Over the past decade, scholarly publishers have designed DTDs and implemented SGML systems. Despite efforts to create a common DTD for scholarly publishing, publishers have developed proprietary DTDs and SGML implementations that diverge from efforts toward standardization on a single DTD because publishers required solutions that best suited their business requirements.

The recognition that individual organizations within an industry segment may have unique business requirements is important for all parties involved in the design and implementation of SGML systems, especially those who work on content or data interchange projects.

We examine differences between DTDs and SGML of ten publishers, explore some reasons for these differences, and discuss issues that are raised by these differences, especially when content must be transformed from one proprietary DTD to another.

A Decade of DTDs and SGML in Scholarly Publishing

What Have We learned?

Table of Contents

A Decade of DTDs and SGML in Scholarly Publishing

What Have We learned?

Bruce Rosenblum and Irina Golfman

§ 1 Introduction

From 1665, with the advent of Oldenburg's *Philosophical Transactions*, until recently, the primary method of scholarly communication was the printed journal [Eisenstein, 1979]. Easily read on paper and archived on long library shelves, the printed journal serves as the "social registry of scientific innovations" [Guédon, 2001].

With the growth of computer use, the invention of the World Wide Web, and the increased pace of scientific research, publishers realized that making research available electronically was vital to both researchers as well as their own business interests. To fulfill this requirement, publishers of scholarly journals turned to two formats, SGML and PDF [Bide, 2000].

Many journal publishers recognized in the early days of SGML that because journal content is highly structured, consistently styled, and has unique archival requirements, SGML is ideally suited to encode it. These publishers were among the first to implement large-scale SGML systems. Over the past decade, scholarly publishers have designed DTDs and implemented SGML systems that take advantage of SGML's strengths [Owens, 1996].

Even before publishers were making daily use of SGML, there were efforts to create a common DTD for scholarly publishing by the Electronic Manuscript Project of the AAP [American Association of Publishers] [Goldfarb, 1993] and the ISO 12083 working group [NISO, 1995]. However, publishers have developed proprietary DTDs and SGML implementations that diverge from efforts toward standardization on a single DTD because publishers required solutions that best suited their business requirements.

In this paper, we will examine differences between DTDs and SGML of ten publishers, explore some reasons for these differences, and discuss issues that are raised by these differences, especially when content must be transformed from one proprietary DTD to another. Finally, we will discuss some issues when developing industry-standard and interchange DTDs.

§ 2 Journal Publishing and SGML

2.1 *DTD Family Tree*

Most journal publisher DTDs owe their legacy, directly or indirectly, to the work of the AAP from 1983 to 1987 (Z39.59), and the ISO 12083 Serial DTD that was derived from this work. The 12083 DTD was released as an ANSI standard in 1988 and became an ISO standard in 1993. The 12083 DTD was last updated in 1995 [NISO, 1995].

Table 1 shows the legacy of ten publishers DTDs. In some cases, DTDs were directly derived from ISO 12083 and only minimal changes were made. In other cases, the DTD author(s) reviewed the structural foundation of 12083, but built a DTD from scratch using a unique set of elements and attributes.

Even those publishers who use 12083 with only minimal modification were unable to use it verbatim. In a 1998 survey conducted by the ISO 12083 working group, the comments of one publisher captured the feelings of many: "It is way too complicated, yet it is not flexible enough to represent the things I need to have in the journal I publish on the Internet" [Kennedy, 1998].

Table 1 DTD Legacy by Publisher.

DTD	Legacy
AIP	Derived from 12083 Serial DTD
BioOne	Derived from 12083 Serial DTD
Blackwell	Developed by Blackwell Science. Significant earlier versions are 2.2 and 3.0
Elsevier	Developed by Elsevier Science. Earlier versions are 1.1.0, 2.1.1, 3.0.0, 4.1.0, and 4.2.0
Highwire	Derived from Elsevier DTD 4.1
IEEE	Derived from 12083 Serial DTD
Nature	First developed by Alden Press
PMC	Derived from the Keton Full Text DTD, which is based on the CJS [Cadmus Journal Services] 2.1 DTD. The CJS DTD is derived from the Elsevier 3.0.0 DTD
UCP	Derived from AAP Article DTD Z39.59 with modifications based on 12083 Serial DTD
Wiley	Based on the Elsevier DTD 3.0.0, as modified from analysis of Wiley journals

Publishers did not adopt ISO 12083 verbatim because it is too generalized. It was designed to work for everyone, but it does not really work for anyone. Every organization has specific needs that must be met in a DTD, and each publisher made changes (major or minor) to meet their requirements.

2.2 SGML vs. XML

Most of scholarly publishers started to archive their content in a structured form prior to the creation of XML. As a result, most publishers today use SGML DTDs rather than XML (see Table 2).

Table 2 DTD type, version and last revision date.

DTD	Type	Definition	Version	Last Revised
AIP	SGML	DTD	3.0.2	August 14, 2001
BioOne	SGML	DTD	1.0.1	October 16, 2000
Blackwell	XML	DTD	4.0	October 2000
Elsevier	SGML	DTD	4.3.1	April 2001
Highwire	SGML	DTD	4.2.14	July 2001
IEEE	SGML	DTD	2.0	February 2, 2000
Nature	SGML	DTD	3.29	July 27, 2001
PMC	XML	DTD	1.13	Sept 10, 2001
UCP	SGML	DTD	Version 6	Sept 19, 2001
Wiley	SGML	DTD	3.4	July 10, 2000

Over time, we expect most publishers to switch to XML. This change will occur in part because there is a wider array of tools for XML than SGML. The time frame for the SGML to XML transition will vary by publisher.

We are not familiar with any publisher that plans to use a different model than a DTD. This may be in part because XML Schema only became a W3C recommendation in February 2001 and other DTD alternatives are still being developed (RelaxNG, Schematron, DSDL). The fact that the XML DTD models for CALS tables and MathML have not been converted to W3C XML Schema (or other

models, e.g., RELAX NG and Schematron) in widely accepted form may also account for some lack of movement away from DTDs by journal publishers. Alex Brown [Brown, 2002] provides a more complete overview of the issues in the DTD versus alternatives discussion in his paper presented at XML Europe, 2002.

Please note that because most publishers still use SGML today, we use the term SGML generically to refer to content tagged in a markup language (either SGML or XML). In cases where the distinction is important, specific reference may be made to XML.

2.3 *SGML and PDF*

Some publishers only create PDF because, relative to the cost of creating SGML, PDF creation is relatively inexpensive and can typically be created as a by-product of the traditional print process. SGML and PDF each have technological advantages and disadvantages. Some key differences are:

- PDF retains the full visual presentation of the original article. While it is possible to create PDF files directly from SGML input, they may not match the original PDF without some manual typesetting. If one's only need is to read the article content, PDF is a much handier and easier format to use than SGML.
- SGML retains structural information about content. Because elements are semantically identified in a precise and granular manner, new information can be discovered. For example, publishers can tag elements in citations, which allows creation of links to indexing databases (e.g., Medline, CrossRef). Without this semantic markup files, such links cannot be established with current databases, and links cannot be created to new databases that may be developed in the future.
- SGML is a "human-readable", non-proprietary format. By human readable, we mean that a person can open an SGML file and see understandable content and structure rather than completely unreadable computer code. By non-proprietary, we mean the format is a publicly documented international standard. In contrast to SGML, PDF is a "binary" format that is impossible for a person to read without Adobe's software. The PDF format is proprietary, owned and controlled by Adobe. Because SGML is human-readable and non-proprietary, there is a higher probability that files will be accessible from an archive 75 or more years in the future. There is no guarantee that the Acrobat Reader will outlive acid free paper

Because PDF and SGML solve different problems, publishers often create both SGML and PDF manifestations of their content. PDF allows all files to be viewed with the same application, independent of the publisher that created the PDF file.

However, in SGML, the same content will be tagged differently by each publisher because each one uses a different DTD. While this strategy serves the needs of individual publishers, it creates a Tower of Babel rather than a consistently accessible repository when SGML is viewed by anyone outside of the publisher's organization. A unique application must be built to render each publisher's SGML.

§ 3 DTD Designs

Different philosophies arose as publishers began to design DTDs and implement SGML systems around their unique business and technical requirements.

3.1 *Style Sheets, Generated and Boilerplate Text*

SGML enables publishers to separate format instructions, which are often proprietary, and structural information by tagging content for semantic meaning rather than format. Format can then be applied to the structural elements with a style sheet when the content is rendered. Steve DeRose comments, "Strong separation of formatting from structure is the hallmark of good SGML use" [DeRose, 1997].

In an ideal SGML application, a complete separation of formatting and structure will be preserved. In journal publishing, the degree of separation varies by publisher. To discuss these differences, we need to define two terms, *generated text* and *boilerplate text*.

We define *generated text* as inconsequential, formulaic, or stereotypical text and formatting omitted from an SGML file, which is applied to content by a style sheet when an SGML file is rendered. The style sheet generates text based on the structure information provided by the markup elements and attributes. For example, generated text includes spacing, punctuation and face markup (e.g., emphasis in the form of bold, italic, etc.) added with a style sheet to the presentation when the `<author>` element is rendered from SGML according to the 12083 DTD.

Boilerplate text is inconsequential, formulaic, or stereotypical text and formatting that has been included in the SGML file even though it could have been omitted.

Some publishers, such as Elsevier, follow DeRose's philosophy:

> In order to separate structure and presentation one applies the concept of generic markup: generic codes (or tags) are placed around most – or all – elements in a document. These elements could be a paragraph, a title, an abstract etc. The tags usually indicate the structure of the document. They do not indicate the style or format of the document, such as fonts, column widths etc. For each different style a style sheet is required to translate the logical structure into a presentation on paper, for example. The set of tags and their mutual relations comprise the 'generic markup language' [Poppelier, et al., 1997].

SGML created according to Elsevier standards excludes almost all boilerplate text and face markup. To render content and apply generated text, a sophisticated style sheet, which is separate from the SGML document instance, is applied to the SGML content. This separation allows the style of presentation to be modified easily, meeting a key goal of Elsevier's electronic workflow requirements. However, because Elsevier does not archive style sheets with SGML files, the style information must be recreated to render SGML in a new environment.

Other publishers, such the University of Chicago Press, take a different view on the issue of generated text:

> Our overriding concern in our SGML implementation was to accurately preserve the entire text as published.
>
> As an example of this design philosophy at work, consider the issue of generated text. Many DTDs, including ISO 12083, either assume or allow for the possibility that the formatting system will generate text such as counters, labels, or the punctuation and connecting text around a list of author names. However, if one uses generated text, then one must also archive the generation rules with the text in order to accurately recreate the original text. We know from experience that journal styles evolve over time; it seemed to us a much better solution to dispense with generated text entirely [Owens, 1996].

The University of Chicago Press approach shows a keen insight into the problems faced by archivists such as libraries. By including more boilerplate text and format information in the SGML file and avoiding generated text, the print version of the article can more easily and faithfully be reproduced from an archive. Blackwell, in the context of explaining their format for structured references, gives further insight into this issue:

> Many SGML and XML DTDs consider punctuation to be generated text i.e., the punctuation required is generated by stylesheet rules and is not stored in the document. The disadvantages of relying on stylesheet rules to create generated text are:
>
> 1. The XML document is no longer a 'standalone' document
>
> 2. The generation rules need to be stored along with the document throughout its life

> 3. The document cannot be read without applying a process which applies the punctuation rules to the XML document
>
> 4. It can be quite inefficient if different rules/templates have to be created to reflect differing punctuation styles across a store of documents
>
> Storing the generated text in an x element in the XML document means that the XML fragment can easily be converted to, for example, simple text, a typesetting format or HTML without the need for complicated templates or rules. If existing rules for generating punctuation are already in place, or if more 'abstract' XML is required, then the contents of the <x> element can be ignored and the rules applied [Blackwell Publishing, 2001].

The range of boilerplate text and formatting publishers include in SGML varies widely. However, inclusion or exclusion is based on publisher policy rather than DTD design because DTDs are not powerful enough to enforce most of these policies. In many cases, even schemas may not be able to enforce them.

3.2 DTD Structural Framework

Even if policies for structure/format separation could be enforced in a DTD or schema, a tightly defined structural framework would become unnecessarily restrictive for some publishers:

> A narrowly targeted DTD can enforce some of these restrictions, but a broadly targeted one cannot, since it must be adaptable to different house styles if required [Megginson, 1998].

The problem is multiplied when a DTD must be broadly targeted to accommodate different house styles:

> The problem is much more complicated than simply choosing names: authors at the two companies are accustomed to thinking about the *structure* of their information in very different ways, and a DTD that is well suited to one will work very poorly for another. If you are a DTD designer, there are four broad approaches that you might chose in this situation:
>
> 1. you can create a DTD that uses a new, neutral structure, different from either of the existing ones
>
> 2. you can impose one of the two structures arbitrarily
>
> 3. you can create a DTD that allows either of the two structures as alternatives
>
> 4. you can create a less-restrictive DTD that can be adapted to any appropriate structure
>
> Most industry-standard DTDs use the fourth method more often than the others, because the DTDs need to be useful for a wide range of applications within a single industry; but, as a result, the DTDs provide a lower level of guidance for authors, validation for processing, and context-sensitivity for searching [Megginson, 1998].

Elsevier Science production specifications require most journals to apply one of five standard style sheets to all content. As a reflection of this production model, the Elsevier DTD is somewhat restrictive. Content must conform to Elsevier's structural requirements so that it can be rendered in one of the standard styles. Content does not conform must be changed so that it conforms to the DTD.

Blackwell Science takes a different approach. Individual journals are permitted to retain their unique styles. In order to accommodate these variations, the DTD is correspondingly less restrictive than the Elsevier DTD, and a greater portion of boilerplate text and formatting is retained in XML files.

The University of Chicago Press chose to retain all boilerplate text and formatting in SGML files. This was a conscious decision driven in part by their requirements to archive all text, *exactly* as it

appeared in print, without the need to archive an accompanying style sheet. As a result, the University of Chicago Press DTD is the least restrictive of these three.

These differences in the DTD structure are not necessarily dictated by technical choice. These decisions have been made for pragmatic reasons rather than philosophical reasons. Publishers' production requirements and business imperatives often drive them. In effect, each journal publisher has incorporated their specific production and business requirements into their DTD structure and their SGML files.

§ 4 SGML Implementations

All of the DTDs we surveyed use a similar high-level structural approach for major article elements, in part because of the shared AAP/12083 heritage. However, the implementation details of SGML document instances vary widely. The largest variance is found in items that have the greatest granularity, such as article history and reference citations. Often the differences revolve around the use of boilerplate versus generated text and formatting. The following sections highlight varying methods of different publishers to tag the same textual content.

4.1 *Six Degrees of Granularity*

The degree of granularity that is required by publishers for certain similar structural elements varies significantly. Article history (which typically includes the received, revised, and accepted dates for the manuscript) is an example where publishers differ widely on how the content should be tagged.

The publishers we surveyed use several models to tag the article history:

- Both parsed and unparsed article history (Blackwell 4.0): `<history><p>Accepted for publication 27 July 1999</p></history> <trackinghistory> <trackingdate type="accepted" date="1999-07-27"/> <trackingdate type="paginated" date="1999-09-12" by="Typesetters Ltd"/> </trackinghistory>`

  Note that Blackwell uses `<history>` for print and `<trackinghistory>` to track production internally, e.g., dates at which the XML was created, edited, published online etc.

- Dates parsed into very granular attributes with boilerplate text included (Blackwell 3.0): `<hst><re year="1997" month="12" day="30">Received for publication December 30, 1997 and <rv year="1998" month="07" day="21">in revised form July 21, 1998 <acc year="1998" month="07" day="27">Accepted for publication July 27, 1998</hst>`

- Dates tagged separately with boilerplate text included (UCP): `<HISTORY>Received <RECEIVED>2001 February 12</RECEIVED>; accepted <ACCEPTED>2001 May 22</ACCEPTED></HISTORY>`

- Dates parsed as attributes with no generated text (Elsevier, Nature): `<re day="20" mo="7" yr="2000"><acc day="18" mo="12" yr="2000">`

- Dates parsed into date type elements with no generated text (AIP): `<history><received><date>26 July 1999</date></received><accepted><date>23 November 1999</date></accepted></history>`

- Dates parsed into very granular elements with no generated text (PMC): `<history><rec><date><day>13</day><month>12</month><year>2000</year></date> </rec><acc><date><day>09</day><month>2</month><year>2001</year> </date></acc><pub><date><year>2001</year></date></pub><epub><date> <day>09</day> <month>2</month><year>2001</year></date></epub></history>`

Each of these models represents similar data in different ways, based on the requirements for how each publisher will use the data.

4.2 Label Text

Figure numbers, which appear at the start of figure captions, are typical of the range of implementations for generated text. Table 3 shows the differences in tagging figure numbers by surveyed publishers.

Table 3 Examples of figure caption numbers. Note the varying degrees of generated versus boilerplate text and formatting in the SGML.

Publisher	Print Example	Corresponding SGML
AIP	FIG. 1.	`<figgrp id="F1">`
BioOne	Fig. 1.	`<TITLE>Fig. 1. Two recent…</TITLE>`
Blackwell	**Figure 1.**	`<num id="leg-f1">Figure 1.  </num>`
Elsevier	**Figure 1.**	`<no>Figure 1</no>`
Highwire	**Figure 1.**	`<no><b>Figure 1.<b> </no>`
IEEE	Fig. 1.	`<title just="just" autonum="off">Fig. 1. The…</title>`
Nature	Fig. 1	`<fig id="f1" entname="figf1">`
PMC	**Figure 1**	`<title><p>Figure 1</p></title>`
UCP	Figure 1:	`<LABEL>Figure </LABEL><NO>1: </NO>`
Wiley	Figure 1.	`<FIG ID="fig1" LOC="FLOAT"><GRAPHIC NAME="fig001"></GRAPHIC><NUMBER>1</NUMBER>`

Several different approaches have been used:

- Some publishers include the label and number as part of the caption (BioOne, IEEE).
- Some publishers include only attribute information (AIP, Nature). The rendering application generates the figure label and number from the attributes. One publisher (Wiley) uses a variation of this model in which only the text of the number is provided.
- One publisher tags the label and number in different elements (UCP).
- Several publishers include the label and number in one element (Blackwell, Elsevier, Highwire and PMC), but there are differences in the degrees of generated text.
    - Highwire includes punctuation after the number and bold face markup.
    - Blackwell and PMC include the punctuation after the number but exclude the bold face markup. The face markup is applied by a style sheet when the SGML is rendered.
    - Elsevier excludes the punctuation after the number and the face markup. A style sheet is used to apply both.

  This overall strategy retains the original text in case anything unusual was done by the author or editor (e.g., "Figures 1a and 1b").

The result in this case is that ten different publishers have ten different ways to tag substantially identical content for a relatively simple element.

4.3 *Citation Text*

The variances in implementations become greater and more complex with a review of citation links. Table 4 illustrates the variety of tagging used for numbered ("Vancouver" style) citations by surveyed publishers.[1]

Table 4 Examples of Vancouver-style citations in different DTDs. Note the varying degrees of automatic text generation and the handling of IDREF attributes for the middle and last numbers of a range.

Publisher	Citation	SGML Example
AIP	superscript.[1,6] superscript.[3–5]	`superscript.<citeref rid="r1" style="superior">1</citeref> <citeref rid="r6" style="superior">6</citeref> superscript.<citeref rid="r3" style="superior">3</citeref> <citeref rid="r4" style="superior">4</citeref><citeref rid="r5" style="superior">5</citeref>`
BioOne	(1,6) (3–5)	`<CITEREF RID="i0031-8655-071-01-0001-b1">&lpar;1,6&rpar;</CITEREF> <CITEREF RID="i0031-8655-071-01-0001-b3">&lpar;3&ndash;5&rpar;</CITEREF>`
Blackwell	[1,6] [3–5]	`&lsqb;<link rid="b1 b6">1,6</link>&rsqb; &lsqb;<link rid="b3 b4 b5">3&ndash;5</link>&rsqb`
Elsevier	[1,6] [3–5]	`<cross-ref refid="bib1 bib6">[1,6]</cross-ref> <cross-ref refid="bib3 bib4 bib5">[3&ndash;5]</cross-ref>`
Highwire	(1, 6) (3–5)	`(<cross-ref refid="bib1" type="bib">1</cross-ref>, <cross-ref refid="bib6" type="bib">6</cross-ref>) (<cross-ref refid="bib3" type="bib">3</cross-ref>&ndash; <cross-ref refid="bib5" type="bib">5</cross-ref>)`
IEEE	[1], [6] [3]–[5]	`<citegrp><citeref rid="ref1" type="ref"></citeref></citegrp>, <citegrp><citeref rid="ref6" type="ref"></citeref></citegrp> <citegrp><citeref rid="ref3" type="ref"></citeref><citeref rid="ref4" type="ref"></citeref><citeref rid="ref5" type="ref"></citeref></citegrp>`
Nature	superscript[1,6]. superscript[3–5].	`superscript<bibr rid="b1 b6">. superscript<bibr rid="b3 b4 b5">.`
PMC [a]	[1,6] [3-5]	`[<abbr bid="B1">1</abbr>,<abbr bid="B6">6</abbr>] [<abbr bid="B3">3</abbr>-<abbr bid="B5">5</abbr>]`
UCP	[1,6] [3–5]	`[<CITEREF RID="rf1">1</CITEREF>,<CITEREF RID="rf6">6</CITEREF>] [<CITEREF RID="rf3">3</CITEREF><CITEREF RID="rf4"></CITEREF>&ndash;<CITEREF RID="rf5">5</CITEREF>]`

Publisher	Citation	SGML Example
Wiley	[1,6] [3–5]	`<BIBR HREF="bib1">1</BIBR><BIBR HREF="bib6">6</BIBR> <BIBR HREF="bib3">3&ndash;5</BIBR>`

[a] PMC allows ranges to be tagged as shown above or as `[<abbr bid="B3">3</abbr>,<abbr bid="B4">4</abbr>,<abbr bid="B5">5</abbr>]`. The style is determined by the publisher who submits SGML content to PMC

Several points should be noted in the tagging of citations:

- For citations to a simple list of references, every publisher includes links to both references 1 and 6.
- For citations to a range of references, publishers may include:
  - A link to only the first reference in the range (BioOne, Wiley)
  - Links to only the first and last reference in the range (Highwire, PMC)
  - Links to all references in the range (AIP, Blackwell, Elsevier, IEEE, Nature, UCP)
- Publishers may:
  - Include all of the original citation text (BioOne, Blackwell, Elsevier, Highwire, Wiley)
  - Keep the original text, but create multiple linking elements around the text (PMC, UCP)
  - Exclude the original citation text (AIP, IEEE, Nature)
- Formatting (parentheses, brackets, superscript) is handled in a variety of ways:
  - No formatting information is retained (IEEE, Nature). Formatting is applied by a style sheet
  - Formatting information is found in an attribute (AIP)
  - Parentheses/brackets are found inside the linking element (BioOne, Elsevier)
  - Parentheses/brackets are found outside the linking element (Blackwell, Highwire, PMC, UCP)
- Several publishers do not retain the range (AIP, IEEE, Nature). The rendering software must include an additional layer of processing to reconstruct the range if the citations are to appear online as a range rather than as discrete links.

The approach taken by Highwire and PMC is designed to allow easy rendering while retaining the general look of the original text. Both exclude links to the middle references in the range. The resulting user interface in HTML is simplified by creating hyperlinks for the first and last numbers as hyperlink text.

This approach causes no loss of significant functionality when linking to a reference section because references in the middle of the range are easily accessible if the reader clicks on the first or last number. However, if links to middle objects of the range are excluded when this model is used for figure or table citations (e.g., "see Figures 1-4") it may be difficult or impossible to link to some objects.

Some publishers recommend a more sophisticated rendering engine that presents a drop down menu with a list of possible targets [Pepping & Schrauwen, 2001], pp. 234-235. A model where text and

links to all objects are preserved, similar to that used by Blackwell and Elsevier, facilitates this rendering style. However, this model requires the use of IDREFS in the DTD rather than IDREF. IDREFS may allow to SGML that is closer to the printed text, but IDREF is easier to render when converting the SGML to simple HTML web pages.

4.4 References

> The devil can cite Scripture for his purpose.[2]

Without doubt, structuring reference sections is one of the most difficult aspects of SGML production for journal publishers. Table 5 summarizes the types of references for which different publishers create structured SGML.

Table 5 Reference types for which structured SGML is created. Book includes monographs and edited books. Conf is a conference proceeding. Standard is a publication by a standards organization such as ANSI, ASTM, etc. An E-Ref is a reference to a web page.

DTD	Journal	Book	Conf	Report	Patent	Thesis	Standard	E-ref [d]
AIP	F	F	F	F	N	N	N	F
BioOne	F	F	F	N	N	N	N	F
Blackwell	F	F	N	N	N	F	N	F
Elsevier	F	F	F	N	N	N	N	F
Highwire	P [a]	N	N	N	N	N	N	F
IEEE	F	F	F	F	F	F	F	F
Nature	F	F	N	N	N	N	N	F
PMC	F	F [b]	F [b]	N	N	N	N	F
UCP	P [c]	N	N	N	N	N	N	F
Wiley	F	F	N	N	F	N	F	F

Key: F: Full structure support; N: No structure support. Reference is unstructured text inside a single SGML element; P: Partial structure support

[a] The Highwire DTD has elements to tag the journal title, volume, date and first page. Other reference components are not tagged because they are not required for reference linking to databases today.

[b] PMC allows fully structured book and edited book references. Some publishers, however, may only partially tag the reference content in their own DTD. The result, when converted to the PMC DTD, may be a partially tagged reference. For example:

```
<bibl id="B12"><title><p>Arenaviruses (Chapter 50).</p></title> <aug><au ca="no" ce="no"><snm>Peters</snm><fnm>CJ</fnm></au> <au ca="no" ce="no"><snm>Buchmeier</snm><fnm>MJ</fnm></au> <au ca="no" ce="no"><snm>Rollin</snm><fnm>PE</fnm></au> <au ca="no" ce="no"><snm>Ksiazek</snm><fnm>TG</fnm></au></aug> <source>In Virology; third Edition. Edited by B.N. Fields, et al. Lippincott-Raven,</source> <pubdate>1996</pubdate> <fpage>1521</fpage><lpage>1551</lpage></bibl>
```

In order to establish a link from this data, it would be necessary to sub-parse the `<source>` data: book name, edition, editors, and publisher.

[c] UCP does not include structural sub-elements in references. Reference linking information to a variety of databases (e.g., ADS, Medline) is stored in reference attributes. When queried, UCP replied they could tag sub-elements based on pattern matching to produce archive SGML.

DTD	Journal	Book	Conf	Report	Patent	Thesis	Standard	E-ref [d]

[d] All surveyed DTDs support tagging of external links in references, allowing the link to be completed if the data is correctly tagged. However, in the case of electronic references, many of the other elements may not be fully structured due to the variety of styles used by authors in citing web resources. As a result, E-refs are fully structured in some cases and partially structured in others.

Typically, journal references have a somewhat predictable structure. Most publishers tag these references for production purposes, including the ability to establish external links. However, the tagging approach varies in several aspects from publisher to publisher:

- Structure:
  - All elements are structured with significant granularity (AIP, BioOne, Blackwell, Elsevier, IEEE, Nature, PMC, Wiley)
  - Only enough content is structured to establish database links (Highwire)
  - No elements are structured (UCP - see footnote c of Table 5)
- Element order:
  - All elements appear in their print order (Blackwell, Highwire, Nature, UCP, Wiley)
  - All elements are ordered in a fixed manner prescribed in the DTD (AIP, Elsevier, PMC)
  - Some elements are ordered in a fixed manner prescribed in the DTD, and some elements appear in their print order (BioOne, IEEE)
- Punctuation:
  - Punctuation between elements is retained (AIP, Blackwell, Highwire, Nature, UCP, Wiley)
  - Punctuation between elements is not retained (BioOne, Elsevier, IEEE, PMC)
- Face markup applied to whole elements:
  - Is not retained (AIP, BioOne, Blackwell, Elsevier, IEEE, Nature, PMC, Wiley). A style sheet applies it when the content is rendered
  - Is retained (Highwire, UCP) eliminating the need for journal-specific style sheets. For example: `<it><title>Eur. J. Immunol.</title></it>`
- Database links (e.g., Medline PMID):
  - Are retained (AIP, BioOne, Blackwell, PMC, UCP)
  - Are not retained (Elsevier, Highwire, IEEE, Nature, Wiley). They are created when the content is rendered

Most of these structuring approaches also apply to non-journal references — if they were parsed. Many publishers do not structure non-journal references because they are more difficult to parse than journal references, they appear less frequently, and databases to which these references can be linked are less common. As a result, some publishers do not include support for non-journal references in their DTDs (see Table 5).

A closer examination reveals that the DTD structures and SGML tagging are related to each publisher's production processes and business requirements.

The Highwire DTD only has elements to structure journal references and the model only includes those elements required to link citations to articles or abstracts. Because Highwire only establishes journal links, this minimalist tagging satisfies their requirements without placing any extra burden on those who create the SGML. Because Highwire does not create links to non-journal content when they place articles online, they have chosen not to include tags to structure these references in their DTD.

The University of Chicago Press takes a different minimalist approach. Reference attributes contain link IDs to databases such as PubMed. However, within the reference text, only face markup elements are included, not semantic markup elements. During SGML creation, the University of Chicago Press parses each journal reference automatically to look up database link attribute values, but then the granular information is discarded. This decision was made in part because the editorial staff does SGML-based copy editing, and it was decided that editing references would be more difficult if the editorial staff had to edit heavily tagged content.

4.5 *Tables*

With one exception, all surveyed DTDs use either the CALS or Elsevier table models, although some publishers still handle tables as graphics rather than as full-text SGML.[3] However, all publishers support scanned images for the table body because some tables are too complex to be captured in SGML. Table 6 summarizes table models used by different publishers.

Table 6 Table model by DTD. Most DTDs use either the CALS or Elsevier table model.

DTD	Model
AIP	CALS
BioOne	CALS
Blackwell	CALS
Elsevier	Elsevier 4.0
Highwire	Elsevier 4.0, although some features have been removed. Highwire also allows submission of CALS tables even though the CALS table model is not part of the Highwire DTD
IEEE	ArborText CALS, although by convention IEEE includes only the table title in SGML, and the body of the table is always scanned.
Nature	CALS
PMC	Elsevier 3.0
UCP	ArborText table model
Wiley	CALS (OASIS version) [a]

[a] The most significant difference between the OASIS CALS model used in the Wiley DTD and the CALS adaptation used by AIP, BioOne, Blackwell and Nature is that table footnotes in the latter DTDs are handled with an element `<tfoot>` that is part of `<tgroup>`. Wiley handles table footnotes in a manner similar to regular article footnotes.

The CALS model is based on the MIL-M-38784B 910201 DTD originally developed for the US Department of Defense. Over the years a large number of organizations have adopted it. OASIS adopted a simplified version of the SGML CALS table model in the mid 1990's and later modified it for XML use. The OASIS version was created based in part on polling software vendors about which features they supported and potential users about which features they most needed. Because the CALS table model has been widely adopted, a significant number of SGML applications have built-in support for it.

The Elsevier table model was introduced in DTD 3.0. It was modified in DTD 4.1 to handle certain complex tables and table embellishments that were unsupported in the earlier DTD.

The Elsevier and CALS table models can structure most tables found in journal articles. The models are not completely parallel. So some structures may be tagged in the Elsevier DTD (e.g., multiple alignment points within a single cell) that may be difficult to replicate in the CALS DTD.

In some cases, however, neither DTD can adequately represent a table. The most common case is when a graphic appears within a table, and the alignment of surrounding cells to specific parts of the graphic must be carefully setup (such alignment typically requires typographic commands that are the antithesis of good SGML markup and are therefore unlikely to be supported in a DTD). In such cases, most publishers recommend that the table content be incorporated in the SGML as a scanned image rather than tagged SGML. When tables are scanned, most publishers follow the protocol of tagging the table number and table caption in SGML and scanning the table body (including heading cells and table footnotes).

All of the table models support left, right and center alignment of text in table cells. Most DTDs support alignment with a specific character (sometimes called "decimal alignment" because it's most commonly used to align a column of numbers by placing the decimal points in a vertical line).

The PMC DTD and early versions of the Highwire DTD do not support character alignment because HTML lacks this capability. This DTD design decision made by Highwire and PMC directly reflects their business requirements, which are focused on the online presentation of content based on today's technologies.

4.6 *Equations*

Published math appears in two basic forms: inline math and display math. Inline math describes simple equations that appear in the running text flow. Display math describes more complex equations that appear on their own line or in their own visual block.

Most inline math is quite simple (e.g., "$x_2 + y_2 = z_2$"). It can usually be represented with Unicode character values and face markup (italic, bold, superscript and subscript). Many publishers do not tag such equations as mathematical expressions, although some publishers (Elsevier) request that these expressions be tagged.

In a few cases, inline math may be more complicated. For example stacked elements, $a_j T^i_j = b$, a simple summation, $\sum_{i=1}^{n} b_i = z$, or a radical, $\sqrt[n]{x+y} = z$ can appear inline. In these cases, the equation must be tagged using SGML markup rather than simple face markup.

All surveyed DTDs include a model for encoding display math and complex inline math with a stream of text commands.[4] Four primary encoding models are used by the surveyed publishers: 12083, Elsevier, MathML, and TeX.[5] These models are summarized in Table 7.

Table 7 Math Models by DTD.

DTD	Model
AIP	ISO12083:1993
BioOne	ISO12083:1993
Blackwell	MathML (W3C, 7 April 1998)
Elsevier	Elsevier Math
Highwire	Elsevier Math

DTD	Model
IEEE	TeX
Nature	ISO12083:1994
PMC	TeX
UCP	Based on ArborText's implementation AAP Math with further changes by UCP.
Wiley	TeX or LaTeX in external file

Because 12083, AAP and Elsevier math are structural cousins, there are actually three primary math models: 12083, MathML, and TeX. In addition, most publishers (but not all, e.g., UCP) support scanned images for math because some equations may be too complex to be captured in SGML.

AAP Math is the original foundation of SGML math markup for most journal publication [van Herwijnen, 1993]. 12083 math, although not directly derived from the AAP DTD, was developed in part based on a review of the AAP DTD. Elsevier's math model is more closely related to AAP although certain semantic constructions, such as explicit integrals and products, have been dropped.

MathML is a newer math model, developed for XML rather than SGML. Unlike 12083 math, which is strictly concerned with presentation markup, MathML can be used for presentation or content markup. "The intent of the content markup in the Mathematical Markup Language is to provide an explicit encoding of the *underlying mathematical structure* of an expression, rather than any particular rendering for the expression" [Diaz, et al., 2001].

MathML first became a W3C recommendation in February 1998, and version 2.0 became a W3C recommendation on February 21, 2001. Most surveyed publishers do not use MathML because they developed their DTDs prior to the original MathML recommendation, and they have not converted their DTDs to XML. Only one surveyed publisher uses MathML (Blackwell). Several publishers have indicated their long-term intent to migrate from 12083 math to MathML [Pepping & Schrauwen, 2001].

Neither 12083 math nor MathML can be natively displayed in most current browsers.[6] As a result, when publishers prepare full-text SGML for online presentation, the equations are converted to an image, usually in GIF format.

TeX [TeX Users Group, 2000] is a powerful ASCII-coded typesetting system created by Donald Knuth of Stanford University in 1981. Leslie Lamport developed LaTeX [LaTeX Project, 2000], a 'dialect' of TeX in 1985. LaTeX is particularly suited to the production of long articles and books, since it has facilities for the automatic numbering of chapters, sections, theorems, equations etc., and also has facilities for cross-referencing.

Because TeX and LaTeX have been around so long, and because they provide tremendous typesetting facilities for complex expressions, they are widely used in the mathematical and physics community. Many publishers have chosen to retain math in TeX or LaTeX rather than convert it to SGML. In addition, some publishers have chosen TeX rather than SGML because all of the subtle presentational nuances of an equation can be completely preserved when the equation is rendered, nuances that might require processing instructions if the equation were tagged in SGML.

There are many tools that convert TeX to GIF images for web presentation. These tools have encouraged some publishers to stick with TeX and bypass SGML for tagging of math. In fact, some organizations prepare SGML for the web by converting SGML equations to TeX and then using a TeX to GIF converter to create a graphic file for each equation.

Most surveyed publishers also permit equations to be captured as scanned images rather than SGML or TeX encoding. While the scanned image route is usually intended for equations that cannot be captured with the available encoding, some SGML suppliers use scanned images for all equations

because they are easier to create than text-encoded equations. This approach preserves the exact visual appearance of an equation, but it precludes the possibility of re-formatting equations at a later date.

§ 5 SGML Interchange

If publishers worked completely in isolation and never had to share their SGML files with anyone outside of their own organization, the differences in DTDs and implementation practices would not be important. However, journal publishers do share their SGML files with other organizations, most commonly with content aggregators and content archives. In some cases, the aggregators and archivists can be one and the same organization.

In this section, we examine issues encountered by Highwire Press at Stanford University in their work with different DTDs. The experience of Highwire Press over the past six years is typical of those faced by any organization that must take SGML designed and implemented for the internal needs of the supplying organization and reuse it under a different set of requirements.[7]

Highwire Press started to place full-text journal content online in May 1995 with the Journal of Biological Chemistry. Initially, Highwire worked with providers of SGML to develop or modify DTDs that would allow online presentation and full-text indexing. For each DTD, Highwire built a custom parser that converted the SGML to HTML for online presentation.

After several years of building custom parsers for SGML to HTML transformation, Highwire developed their own DTD. Ideally, all content is delivered in this DTD, however, if a customer already creates content in another DTD, Highwire converts the customer's content to the Highwire Press DTD.[8]

The Highwire DTD was designed to satify the following key goals:

1. Highwire's main goal is to present full-text content online. In addition, Highwire provides archive services to their publisher partners.
2. Online presentation provides rich linking opportunities unavailable in print publications. Where possible, Highwire creates electronic links that will enrich the research experience of the full-text user.

Highwire derived their DTD from the Elsevier 4.1.0 DTD.[9] The DTD was changed to address the specific requirements that Highwire faced in placing content online. They simplified some element models that did not affect online presentation and linking while they added other elements that aided in tracking articles and creating links to related content. The changes made include:

1. Elimination of some features that Highwire did not need (e.g., keyword classifications, appendices)
2. Simplification of some DTD features, most notably the reference section where references are encoded as formatted text with only the minimal tagging required to create external links
3. Addition of some new features that Highwire requires for online presentation and linking (e.g., the `<addart>` element used to create links between articles in a journal via the volume and page number).

All content delivered to Highwire is now converted into the Highwire DTD[10] , and Highwire encourages new customers without a DTD to adopt the Highwire DTD from the start.

This new approach for Highwire has generally been successful; however, it has not been without problems. The following sections examine some of the issues Highwire has faced.

5.1 *The Startup Process*

When a journal first submits SGML to Highwire, a formal validation process is conducted. At least two issues of the journal receive careful proofing. Problems are reported to the publisher, and Highwire works directly with the publisher or SGML provider to facilitate changes that may be required in their SGML production processes. The goal of validation is to ensure that deliveries will be consistent and correct.

Tables and math receive special attention during the startup process. Tables, which may be submitted in either CALS or Elsevier format, sometimes have problems with column headers and spanning cells. Graphics embedded in tables may cause alignment problems, and Highwire recommends scanning those tables with graphics that require alignment to specific cells. In some situations, publishers submit all tables as scans rather than full-text SGML. Highwire discourages this practice because the scan files are large, slow to load in browsers, and cannot be indexed for searching.

Unfortunately HTML has more limited presentation capabilities than CALS or Elsevier SGML. For example, decimal or character alignment is unsupported in HTML. As a result, Highwire sometimes sacrifices a degree of table formatting in online presentation. Their primary goal is to ensure the table is readable.

Math is usually tagged according to the Elsevier or MathML DTDs, or encoded as TeX. Because most browsers are unable to render math, the equations are converted to GIF images. Highwire converts all equations from their SGML format to LaTeX, and then converts the LaTeX to GIF images. Sometimes math presents formatting problems, especially in long equations that require line breaks. As a result, some equations receive hand massaging even during the regular production process.

When validation is complete and all problems have been resolved, Highwire moves the journal to a more automated process for regular issues. The importance of this validation process for quality of the final presentation cannot be overstated. Without this step, the overall quality of journals hosted by Highwire would be much lower.

5.2 *Content Transformation Issues*

Highwire faces a number of challenges when converting content from other DTDs to the Highwire DTD. Some of these challenges include:

1. The format of reference citations in the source DTD may be difficult to convert into the Highwire DTD. Ranges of numbered citations (e.g., "[3-5]") can be especially problematic (See Table 4).
2. Highwire often finds appendix section headers, table titles, or figure captions are incorrect in SGML. As a result, correct placement and linking of appendix sections, figures, and tables can be difficult.
3. Many journals now include magazine-style news content in the front section. In many cases, the design and layout of this material is a freeform style that differs significantly from the standard article layout. This content may have uncited or uncaptioned figures and extra design elements that make them difficult to reproduce online.
4. Many publishers do not tag content required for links to external databases.
5. Special issues present some extra challenges, and supplements in particular, because of inconsistent or unwieldy numbering standards. Publishers may restart page numbering for supplementary issues at page 1, creating ambiguous situations that today's reference linking applications cannot resolve.

5.3 *Quality and Consistency Problems*

All of the problems discussed up the to this point would exist even in a perfect high quality SGML production system. Unfortunately, real world production of SGML has shown that consistency and high quality are not always achieved [Bide, 2000].

Highwire has found several kinds of quality and consistency challenges:

1. Files are submitted that fail to parse.

   All organizations that create SGML parse the files. However, sometimes they fail to re-parse files after making final edits. In addition, sometimes SGML files become corrupt in transmission, which may cause parse errors. In such cases, the files are returned directly to the provider for appropriate repairs.

2. Use of SGML elements may be inconsistent.

   Usually Highwire resolves consistency problems during the startup process. Because some SGML files are created by manual keying rather than software-driven processes, and the people who create them may change over time, the new people may not always apply designated tagging standards.

   As a general rule of thumb, the more people who create SGML according to a DTD, the more variation you will find because each person, in the absence of specific rules (or even in the presence of such rules), will find a unique way to tag the same article.

   Inconsistent tagging can be benign, but in other cases it may effect either presentation or searching of content. Consistency can be enforced with constant manual checking, or a custom application can be developed to report consistency errors.

3. The content between the tags is incorrect.

   This issue is the most difficult one that Highwire and most other publishers working with SGML face.

   When SGML files are created, the degree of quality assurance conducted varies widely. Some organizations check that the file parses but little more. More sophisticated organizations may have developed manual or automated systems for quality validation.

   These problems arise from several sources:

   a. The content may have been incorrect in print as a result of author, editorial or production errors. The most common errors appear in the reference section, resulting in failed links to databases such as Medline.

   b. The SGML supplier may misinterpret the content, possibly because it is ambiguous. For example, when applying given name and surname tags, it may be unclear how to handle some non-Western names, or names from societies where surnames are not commonly used. For example, the names "Ho Chi Minh" and "Leonardo da Vinci" would likely receive inconsistent tagging.

   c. Suppliers may fail to tag certain elements that Highwire feels are essential for online presentation. Typically, this information is used to create links.

   d. In some cases, SGML suppliers create sub-standard work due to internal issues.

Minimization of quality and consistency problems requires startup validation and ongoing quality assurance processes. Because content sent to Highwire is put online almost immediately, most of these problems are caught at some stage of the production process. Furthermore, because Highwire is organizationally focused on quality and customer satisfaction, they have created feedback loops to customers to ensure that systemic problems with SGML quality are addressed.

5.4 *Linking Problems*

One of the most compelling reasons for electronic journals is linking. The ability to rapidly follow hyperlinks through research materials allows new forms of discovery. For this reason, Highwire pays special attention to creation of links when placing journals online.

The SGML Highwire has received from customers shows that there are consistent issues with linking:

1. SGML from some suppliers may not provide links between floating objects (e.g., figure, tables) and their citations. In some cases, links might be missed because they are in unusual places. For example, if Table 4 was scanned and Figure 5 is cited only from the body of Table 4 there is no place in the SGML file to create a link.[11] Missing links make it difficult to correctly place floating objects.
2. Links from citations to references may not be created if the citation was incorrectly formatted. A common example is a name-date (Harvard style) citation with a typo in the author's name or the year that was unchecked during copy editing. In such cases, no link is created from the citation to the reference.
3. Linking to full text available at Highwire, Medline, ISI, or other external databases fails. One of the most common quality problems encountered, reference link failures are most commonly the result of author error. For example, if the author incorrectly transcribed a first page number or date, the reference will fail to link. Mistakes by SGML suppliers also cause linking failures, most often because a supplier did not tag the elements in a reference required to create the link.
4. Links to the Genbank, PDB, or Swisprot databases fail for one of the following reasons:

   a. The database number was untagged because the DTD did not support it.
   b. The DTD supports database links, but the SGML supplier failed to tag it.
   c. The link was tagged in the SGML, but it fails to resolve because of a typographical error. One of the most common errors is using a capital letter 'O' rather than a digit '0', or using a lower case letter 'l' rather than a digit '1'.

   Highwire uses regular expression pattern matching to automatically identifying untagged Genbank numbers with a relatively high success rate. PDB and Swisprot numbers are harder to automatically tag.

   Links to web addresses (HTTP and FTP) and email addresses fail for one of the following reasons:

   a. The supplier failed to tag the information in the SGML file.
   b. The information was tagged as a link in the SGML file, but the type of link was unidentified.

   In cases where the link was untagged, regular expression pattern matching may work. This automatic tagging is frustrated by problems such as spaces in the middle of web addresses. Typesetters often insert spaces in web addresses to create better looking text flow in narrow print columns. The spaces are not removed during the process that creates SGML from the typeset file, and the result is a visually pretty but semantically incorrect web address.
5. Links to related articles may fail. Every publisher has a unique standard for creating links between related items [Rosenblum & Golfman, 2001]. For example, an erratum should have a link back to the article to which it is related. These links may fail for the following reasons:

a. The volume and page information to create a link was untagged because the DTD did not support it.

b. The publisher failed to tag the appropriate volume and page information in the item.

c. The publisher uses a system to create the link that is not easily converted to HTML. For example, The Elsevier DTD maintains these relationships in the refers-to attribute of the `<art>` element. The location of this information in the SGML file does not conveniently create a location in the article for a logical hyperlink to the related content.

Highwire tries to compensate for incorrect or missing link information whenever possible. However, they deliberately try to undershoot rather than overshoot when automatically creating links because they believe it's better to miss a link than to create an incorrect link. In some cases, they limit their regular expression pattern matching to specific sections of the document. For example, they only search for untagged email addresses in the front matter of an article.

5.5 *Transformation Location*

Highwire encourages customers to deliver content in the Highwire DTD, but they cannot require publishers who have their own DTD to deliver in Highwire format. When a DTD transformation is required, Highwire does the transformation, rather than the publisher, for several important reasons:

1. Publishers may not want to switch to the Highwire DTD as their primary DTD because they use their SGML for additional purposes.
2. Most publishers are not interested in bearing the burden or expense of creating SGML in a second DTD.[12]
3. Highwire charges publishers a fee associated with setup of the transformation to the Highwire DTD; however, this fee is less than the cost of having the publisher build or buy a system to convert the SGML themselves. It costs Highwire less than other organizations to build this transformation because:

    a. They are familiar with their own DTD

    b. They can reuse pieces of other transformations (e.g., regular expression pattern matching to identify linking elements)

    c. They know the mission of their transformation intimately, and can create the most efficient transformation to achieve exactly that mission.

4. Highwire has greater control over the quality of the transformation if they do it rather than someone else. Highwire uses a small team to create software-driven transformations. By using a small team under one roof, there is likely to be less variance in the SGML created ("fewer hands, fewer errors"), allowing for a smoother flow of the resulting SGML into Highwire's online presentation systems.

While there are many advantages to converting SGML at Highwire rather than at the publisher, there is one significant disadvantage: if a publisher significantly upgrades their DTD, a full-fledged parser update, coupled with extensive integrity testing, is required at Highwire.

Communication of DTD upgrades to Highwire, whether major or minor, is critical. In many cases, Highwire has only learned of a DTD upgrade through the failure of a file to parse, rather than through proactive communication from a publisher.

§ 6 SGML Quality Control

The Highwire experience illustrates that high quality results can be maintained through clear standards, continual monitoring, feedback mechanisms, and appropriate levels of investment. However, Highwire is not the only organization to have addressed this problem directly. Elsevier Science provides another valuable case study.

When Elsevier Science first required full-text SGML, they provided a DTD and documentation for the DTD. Even with a 90-page reference manual, Elsevier found that the quality of the SGML they received from suppliers was not as high as they had hoped. Files parsed (most of the time), but high quality SGML required more than parser validation.

Elsevier found that the interpretation of the DTD was inconsistent, sometimes because desired interpretations were undocumented, and sometimes because those people creating SGML had not memorized the documentation. Certain policy decisions that could not be encoded in a DTD needed clarification. Some common author mistakes appeared in SGML because editors did not catch them; sometimes editors made mistakes that affected SGML as well.

When these problems were encountered in 1996, Elsevier was archiving a lot of the SGML - they were not yet placing most of their SGML online. In this regard, the problem Elsevier faced with deferred-use of the SGML was similar to the problems that archivists will face.

Elsevier realized that these errors would cause long-term problems if they were left unchecked and unfixed. In order to catch these and a wide range of other errors, Elsevier started to build a quality control application, known simply as "QCTool".[13]

QCTool validates that a file parses, and then it reports three classes of problems: errors, warnings, and notifications. Errors are problems that indicate misuse of the DTD or violations of Elsevier policies (for example, a non-EMPTY element that has no content). Warnings are lesser issues that probably indicate incorrect SGML (for example, the text "Smith (2001)", if untagged, will cause a warning to be issued that a possible citation needs tagging). Notifications are warnings that might indicate a problem, but are just as likely to be a false warning.

Initially QCTool was used inside Elsevier to examine SGML files for errors. As more errors were caught with the tool, a small team known as "the repair shop" started to fix the most egregious ones. It did not take long to realize that the capacity of this team to fix the errors reported by QCTool would soon be exceeded. In addition, Elsevier decided that the responsibility to fix these errors lay with the supplier, not with Elsevier.

In mid-1997, Elsevier distributed QCTool to suppliers and asked them to run SGML files through it in the hope that suppliers would fix the problems that were reported. In some cases, though, the number of problems reported was so large that suppliers found it more expedient to ignore QCTool. In addition, some errors were the result of author or editorial mistakes, and sometimes it was unclear how they should be fixed.

By mid-1998, most of the early questions about how to use and respond to the QCTool were resolved. But because a large number of errors were still being fixed in the repair shop, Elsevier no longer *requested* that suppliers run QCTool and fix the problems. New policies were instituted that *required* suppliers to run QCTool and fix all errors. Warnings and notifications were to be heeded, but they need not be fixed.

To reinforce the importance of the QCTool, Elsevier announced that any files with errors would cause the entire issue submission to be rejected by Elsevier and returned to the supplier for repair and resubmission (we call this a negative feedback loop). To insure that suppliers adhere to these policies, Elsevier runs QCTool on all files when they are received by Elsevier Science. Elsevier trusts suppliers to run QCTool, but they verify that they have run it.

§ 7 Lessons Learned

There is much that we can learn from a decade of DTDs and SGML in scholarly publishing. Some of the lessons are applicable strictly to those in scholarly publishing; however, most of the lessons apply to anyone who works with SGML or XML, regardless of whether they use DTDs, W3C XML Schemas, or any other document schema definition language.

7.1 *Lessons for DTD Developers*

The most important lesson for DTD developers is to make sure you know your short term and long term business requirements. These requirements include knowledge of how SGML will be created, used, and re-purposed. It is important to remember that SGML systems are more than technology; they are a business investment, and that investment can only be justified with a reasonable return on the investment.

DTD designers must understand the organizational workflow that will be used to create SGML. Some DTDs have elements that are very labor-intensive, i.e., the cost to tag them is high in either ongoing labor costs or development of software-based pattern recognition systems. If SGML is too expensive to tag according to the DTD, the SGML project will probably not be successful. Many groups that use SGML are long past version one of their DTD, and revisions were often made based on the cost to tag elements of marginal business value.

Once the business requirements are well understood, the DTD developers can start to wrestle with more detailed issues as part of their document analysis. Key issues include how loose or restrictive the DTD should be and how much text is generated by the style sheet.

These two issues reach to the core of DTD design and SGML implementation philosophy. Design decisions around these issues have not been without controversy in some organizations. However, the decisions surrounding these issues should be driven by the business imperatives.

Business imperatives should drive DTD designers toward standard DTD fragments when they match the business objectives. The movement toward CALS tables and MathML, which have increasing support from tool vendors, are examples of good use of standard DTD models.

When the DTD is complete, it must be carefully and completely documented. Sample files that illustrate all features of the DTD are an essential part of the documentation because DTD (or Schema) interpretation often cannot be controlled through validation. When a DTD is well designed according to the business requirements of the client organization(s), the business requirements will be evident from a review of the DTD and sample document instances.

7.2 *Lessons for SGML Implementers*

SGML system implementers face the daunting task of turning the DTD design into reality. Execution first requires that they understand the design goals and implementation requirements of the DTD. The DTD documentation and sample files are critical for acquiring this knowledge.

Armed with this knowledge, the implementers must produce SGML in a cost-effective yet high quality manner. Once the SGML production system has been implemented, whether the SGML is created in-house or outside of the organization, and whether it is done by hand tagging or with automated tools, quality assurance is an essential part of the process.

As many publishers have discovered, no matter how complete the documentation, strict conformance to the DTD designer's interpretation can only be ensured with software-based quality control tools. However, SGML producers will not use quality control tools and fix reported problems without negative feedback loops. Publishers must actively monitor and enforce quality standards, even after providing tools to SGML producers.

The two most important lessons for implementers are 1) demand excellent documentation and 2) develop and use automated quality assurance tools.

7.3 Lessons for SGML Consumers

SGML "consumers" are users of tagged content, not the knowledge consumers. SGML consumers include content aggregators, derivative content publishers, online providers and library archives. They are often outside of the organization that created the SGML. A common characteristic of SGML consumers is that they must transform SGML from one or more DTDs into their own SGML or XML DTD, or HTML.

SGML consumers are in an unenviable position. By the time the SGML arrives at their desk, the supplier's business requirements have been determined, the DTD developed, and the SGML systems implemented; and chances are good that none of this bears any relation to the business requirements of the SGML consumer.

So, how does the SGML consumer go about mapping similar information from multiple DTDs that was created independently and without coordination to a single target DTD?

First, make sure you understand *your* organization's business requirements. Then review as many source DTDs as possible. Study the DTDs, sample files and documentation to learn each designer's philosophy. Learn about the business requirements of each organization supplying SGML and determine how closely they are aligned with the business requirements of your organization.

Next, create a mapping of your business requirements and how they correlate to your business partners' requirements. Ask yourself some questions. Are my semantic requirements the same as my supplier? Are they supplying all of the elements that meet my requirements, or do I need to add some markup to their files? Do I need all of their elements? Can I afford to drop some semantic elements (we call this a transformation with "tolerable loss" because it does not retain the complete semantic markup of the source documents, but it meets the SGML consumer's requirements)? Only when you understand how these requirements mesh can you build the requirements for your DTD.

Based on your organization's business requirements, you can evaluate which elements of each DTD are important and which elements can be dropped. The tradeoff between tolerable loss and DTD complexity should be evaluated for each element that of the new DTD.

This pragmatic approach is more feasible than creating a superset DTD that includes every element from every supplier. Your DTD will most likely fall between the intersection and the union of structural elements found in the source DTDs, and it will probably need to have a looser content model. Making the DTD too restrictive will make transformations from DTDs with looser models difficult if not impossible.

In summary, the following lessons apply to SGML consumers:

1. Successful transformations can be completed only if the business requirements of both the supplier and consumer are clearly understood before the DTD development and transformation process is started. If the business requirements are not similar, SGML transformations will take more work to complete.
2. Some content may be more difficult to convert from one publisher's DTD to an archiving DTD due to how the same structural element may be handled differently in DTDs.
3. The quality and consistency of incoming SGML cannot be ensured. However, it can be significantly improved through an initial validation process and ongoing feedback loops with the SGML provider.

4. Significant publisher DTD upgrades can cause costly rework of the SGML transformation system. Additionally, one cannot rely on publishers to communicate information about DTD upgrades.
5. Typically the lowest cost and highest quality transformation results when the SGML consumer controls the transformation from other DTDs to their own DTD because the SGML consumer can ensure that the transformation meets their business requirements.

§ 8 Conclusions

The AAP and 12083 DTDs were important projects. They laid the structural foundations for subsequent DTDs used in journal publishing. They did not succeed, however, in their goal of becoming industry-standard DTDs. This goal was not reached because, while these DTDs were generalized for the needs of the industry, they did not meet the specific business requirements of individual organizations within the scholarly publishing community.

The recognition that individual organizations within an industry segment may have unique business requirements is important for all parties involved in the design and implementation of SGML systems, especially those who work on content or data interchange projects.

Industry-standard DTDs work best for data interchange, but only when they have been designed with a loose enough model to accommodate the business requirements of all groups within the industry segment. However, when data interchange is not one of the SGML business requirements, DTDs can be more narrowly targeted to meet the specific business requirements of an organization.

Notes

1. We have chosen only a close review of Vancouver-style citations here. A comprehensive review of name-date ("Harvard" style) citations would show even more variations in style because of the how these citations are formatted for print by different publishers.
2. William Shakespeare. *The Merchant of Venice*. Act I. Sc. 3.
3. The authors are familiar with scholarly publishing DTDs outside of the surveyed group that do not have any support for tables. These publishers scan all tables.
4. Some publishers outside of the scope of the surveyed DTDs do not feel it is necessary to invest in support of tagged equations because their content includes them infrequently. These publishers treat all display equations as scanned images.
5. We carefully avoid the word "tagging" here because TeX and LaTeX are not SGML or XML.
6. W3C's Amaya browser can be used to render MathML. See http://www.w3.org/Amaya/.
7. The information in this section is based on a conference call with Highwire Press representatives Maureen Phayer and Diana Robinson on September 24, 2001. It is supplemented by the author's experience working with providers of SGML content for online publication by Highwire.
8. Some publishers do not provide SGML to Highwire. In most cases, the non-SGML files are typeset files from systems such as XyVision, Quark, FrameMaker or Miles 33. Highwire arranges for conversion of these files to SGML.
9. At the time Highwire developed their DTD, almost every Highwire partner was using one of the Elsevier DTD versions or the Keton DTD (an Elsevier derivative). This experience with the Elsevier DTD and its derivatives led Highwire to stick with this model rather than chose a significantly different model.

10. There are a few exceptions, notably content for which custom parsers had been previously built. Highwire decided that the existing systems to handle those DTDs did not need to be updated. At the current time, Highwire converts content into the Highwire DTD from approximately a dozen DTDs including: Blackwell, Elsevier, Lippincott Williams and Wilkins, New England Journal of Medicine, Oxford University Press, OvidBase, and Radiological Society of North America.
11. Many editors would submit that this is a case of incorrect authoring or editing; however, we have seen examples of such cases in SGML instances.
12. This rationale is consistent with the experiences noted in [Bide, 2000]. Most publishers have limited technical staffs, and the cost of compliance with an additional DTD is not sufficiently low to make it more attractive than the cost of Highwire converting the content.
13. Elsevier is not the only publisher to create an integrated quality control tool. UCP is another publisher with tools to check content inside elements.

Acknowledgements

We extend our thanks to Dale Flecker at Harvard University for permission to extensively cite material from the Harvard-Mellon E-Journal Archive DTD Feasibility Study, and to MacKenzie Smith, Stephen Abrams and Marilyn Geller, at Harvard University for many stimulating conversations about the mission of an E-Journal archive. We are grateful to Maureen Phayer and Diana Robinson at Highwire Press for taking significant time to discuss Highwire's experiences, and to G. Paul Bozuwa at Capital City Press for permission to reference our experience implementing their SGML systems for scholarly publishing.

We would like to thank the following publishers that participated in this study, provided materials, and responded to our queries: Bob Hollowell (American Institute of Physics); Kristine Schnebly (BioOne); David Sommer, Richard O'Beirne (Blackwell Science); Karen Hunter, Jos Migchielsen (Elsevier Science); John Sack, Maureen Phayer, Diana Robinson (Highwire Press); Stephen Cohen, Ken Rawson (IEEE); Y Kathy Kwan, Ed Sequeira (PubMed Central); Howard Ratner, Heather Rankin (Nature Publishing Group); Evan Owens, John Muenning (University of Chicago Press); and Margaret Wallace (John Wiley & Sons)

Bibliography

[Bide, 2000] Bide, Mark. *Standards for Electronic Publishing*. 2000. http://www.kb.nl/coop/nedlib/results/e-publishingstandards.pdf. Accessed on August 28, 2001.

[Blackwell Publishing, 2001] Blackwell Publishing. *Blackwell Publishing DTD 4 documentation*. 2001. http://www.blackwellpublishing.com/xml. Accessed on September 18, 2001.

[Brown, 2002] Brown, Alex. *XSD Schemas in Book and Journal Publishing*. 2002. XML Europe 2002 Conference, Barcelona, Spain. http://www.idealliance.org/papers/xmle02/dx_xmle02/papers/03-01-02/03-01-02.pdf. Accessed on June 14, 2002.

[DeRose, 1997] DeRose, Steve. *The SGML FAQ Book*. 1997. Kluwer Academic Publishers: Norwell, MA.

[Diaz, et al., 2001] Diaz, Andrew, et al. *W3C's Math Home Page*. 2001. http://www.w3c.org/Math/. Accessed on October 31, 2001.

[Eisenstein, 1979] Eisenstein, Elizabeth. *The Printing Press as an Agent of Change*. 1979. Cambridge University Press: Cambridge.

[Goldfarb, 1993] Goldfarb, Charles. *A Brief History of the Development of SGML*. 1993. http://www.oasis-open.org/cover/sgmlhist0.html. Accessed on April 2, 2002.

[Guédon, 2001] Guédon, Jean-Claude. *In Oldenburg's Long Shadow: Librarians, Research Scientists, Publishers, and the Control of Scientific Publishing*. 2001. http://www.arl.org/arl/proceedings/138/guedon.html. Accessed on April 2, 2002.

[Kennedy, 1998] Kennedy, Dianne. *ISO 12083 Survey*. 1998. http://www.xmlxperts.com/12083.htm and http://www.xmlxperts.com/survey98.htm. Accessed on October 4, 2001.

[LaTeX Project, 2000] LaTeX Project. *LaTeX: A document preparation system*. 2000. http://www.latex-project.org/. Accessed on April 2, 2002.

[Megginson, 1998] Megginson, David. *Structuring XML Documents*. 1998. Prentice Hall: Upper Saddle River, NJ.

[NISO, 1995] NISO. *ISO 12083–1995: Electronic Manuscript Preparation and Markup*. 1995. NISO Press: Bethesda Maryland.

[Owens, 1996] Owens, Evan. *SGML and The Astrophysical Journal: A Case Study in Scholarly and Scientific Publishing*. 1996. http://www.journals.uchicago.edu/sgml96.html. Accessed on October 11, 2001.

[Pepping & Schrauwen, 2001] Pepping, Simon, and Schrauwen, Rob. *Tag by Tag*. 2001 Elsevier Science: Amsterdam.

[Poppelier, et al., 1997] Poppelier, Nico, et al. *Document Type Definitions For Serial Publications Part I: Article DTD 4.1.0*. 1997. http://support.sciencedirect.com/sgml/nov1997/refman.pdf. Accessed on April 2, 2002.

[Rosenblum & Golfman, 2001] Rosenblum, Bruce, and Golfman, Irina. *E-Journal Archive DTD Feasibility Study*. 2001. http://www.diglib.org/preserve/hadtdfs.pdf. Accessed on April 2, 2002.

[TeX Users Group, 2000] TeX Users Group. *TeX Users Group Home Page*. 2000. http://www.ams.org/tex/. Accessed on April 2, 2002.

[van Herwijnen, 1993] van Herwijnen, Eric. *ISO 12083 Math DTD*. 1993. http://www.ams.org/html-math/iso12083.html. Accessed on December 3, 2001.

The Authors

Bruce Rosenblum
Inera Incorporated
815 Washington Street, Suite 3
Newton
MA
02460
USA
tel: (617) 969-3053
fax: (617) 969-4911
bruce@inera.com
http://www.inera.com

Bruce Rosenblum, CEO of Inera since 1997, has spent more than twenty years developing electronic publishing systems. He frequently consults to scholarly publishers on the application of XML in journal publishing. At Inera, he leads the design and development of eXtyles™, a suite of editorial and XML tools used to produce more than 125 scholarly journals. Prior Inera, Mr. Rosenblum was Vice President at Turning Point Software where he led the design and development of software products for companies such as Microsoft, Word Perfect, Broderbund, Houghton Mifflin and Funk Software.

Irina Golfman
Inera Incorporated
815 Washington Street, Suite 3
Newton
MA
02460
USA
tel: (617) 969-3053
fax: (617) 969-4911
irina@inera.com
http://www.inera.com

Irina Golfman founded Inera Incorporated in 1992 to provide SGML-related consulting services. Under her guidance, Inera has successfully completed SGML and XML-based projects for clients in the printing, publishing, manufacturing, computer, and financial services industries. Prior to founding Inera, Ms Golfman was Director of Product Development for Kurzweil Computer Products, a division of Xerox. In that role, she managed the design and development of OCR/ICR products for Macintosh, Windows and Unix. Prior to Xerox, Ms. Golfman held positions in software development at Wang Laboratories, Linkware, Codex Corporation, and Prime Computer.

Mr. Rosenblum and Ms. Golfman have presented papers and tutorials at SGML and XML conferences in the United States and Europe, Seybold Seminars, and have been featured speakers at a number of local SGML and XML related events. They are active members of OASIS, GCA's Independent Consultant Cooperative, and SGML Forum of New York.

Extreme Markup Languages 2002
Montréal, Québec, August 6-9, 2002
This paper was formatted from XML source via XSL
Mulberry Technologies, Inc., August 2002