XML: Using an Evolving Standard in Electronic Publishing

Ann Apps and Ross MacIntyre

MIMAS, Manchester Computing, University of Manchester,
Oxford Road, Manchester, M13 9PL, UK
ann.apps@man.ac.uk, ross.macintyre@man.ac.uk

Abstract. XML is the proposed electronic publishing and data interchange format of the future. Currently XML is immature with little tool support, particularly for end-user World Wide Web browser display. At present many journal publishers typeset their articles, or at least their article metadata, in SGML which is converted to HTML for end-user display. The SGML article metadata is used for electronic data interchange for supply to external aggregating and abstracting agencies. This paper discusses the change to the electronic journal publishing production process implied by an adoption of the World Wide Web standard format of XML, the problems associated with using and displaying XML now, and the future benefits of adopting a standard interchange format. Real work experience gained by working with journal publishers in actual electronic publishing scenarios is used as illustration.
Keywords. XML, electronic journals, metadata, DTD, electronic data interchange.

1 Introduction

XML (Extensible Markup Language) [1], [2] is the proposed electronic publishing and data interchange format of the future. Currently, although being billed as 'tomorrow's solution today', XML is immature with little tool support, particularly for end-user World Wide Web browser display. At present many journal publishers typeset their articles, or at least their article metadata, in SGML (Standard Generalised Markup Language) [3]. This is converted to HTML for end-user display. The SGML article metadata is used for electronic data interchange for supply to external hosting, aggregating and abstracting agencies. The adoption by the World Wide Web of the standard format of XML implies a change to the electronic journal publishing production process. There are problems associated with using and displaying XML now, but there could be future benefits in adopting a standard electronic publishing and data interchange format.

2 An Electronic Journals Application

The Manchester University Press (MUP) [4] electronic journals application hosted by MIMAS ('epub@mimas') [5], at the University of Manchester, UK, holds the article header information in XML format. The bespoke XML Document Type Definition (DTD) used for this electronic journals application uses elements which are based on the Dublin Core Metadata Element Set [6] with local enhancements and qualification schemes. Using Dublin Core elements for the article header information recognises that this is essentially metadata for the journal articles. XML was chosen as the syntax to hold the base article metadata to allow for possible future developments of the electronic journals application where a flexible format would be an advantage. Using XML also provided a testbed for research into its use within a practical, 'real' application. The design of this electronic journals application was developed from work done at MIMAS on the *Nature* Digital Archive project [7], [8], and from previous work on the SuperJournal project [9]. Brief extracts given below are from a sample article header [10] published in the January 2000 issue of the 'International Journal of Electrical Engineering Education'. This complete example article header [11] may be viewed via the 'epub@mimas' web site.

2.1 The Data Handling Process

Journal article metadata is supplied to MIMAS in SGML format, using the 'Simplified SGML for Serial Headers' (SSSH) [12] Document Type Definition (DTD) for the majority of the journals but a typesetter's proprietary DTD for one journal. The full articles are supplied as PDF files and are displayed in this format to the end-user.

During the data handling process, the SGML metadata for articles in a journal issue is transformed into a common XML format using a bespoke program written in the OmniMark [13] programming language. At the same time an XML 'Table of Contents' file for the issue is generated, and the higher-level 'Tables of Contents' files, listing issues available for the journal and the current issue in the application journals list, are updated. All the XML generated is validated by parsing against the appropriate DTD using the XML parser, *sx* [14], which is part of James Clark's SGML parsing suite, *SP* (which also includes the SGML parser, *nsgmls*). This parser is used for validation because it is stricter than the current OmniMark XML parser, which is SGML-derived. For instance, basic character data, *#PCDATA*, is not

allowed in a nested model group in XML, a construct which is faulted by *sx* but not by OmniMark. An example of this construct is:

```
<!ENTITY  % data "(#PCDATA | i | b)*"> <!ELEMENT jtl - - (%data;)>
```

which is valid SGML, but in XML should be changed, so that #PCDATA is not nested, to:

```
<!ENTITY  % data "#PCDATA | i | b"> <!ELEMENT jtl - - (%data;)*>
```

The former definition is correctly faulted by *sx*, but not by the XML parser within the current version of OmniMark (5.3).

As a result of this data handling process, which is run as a single program, all the information about the journals, issues and articles is held in XML format, capturing just the information content and hierarchies. No display formatting information is contained within the XML files, with the exception of simple bold and italic tags within text strings. Also the ordering of the elements within the article metadata XML file does not dictate their ordering in the final display to the end-user.

For example, the title and authors of an article are captured in the XML as:

```
<DC.Title LANGUAGE="en">Specifications and standards for learning technologies:
the IMS project</DC.Title>
<MUP.Creator SCHEME="MUP.AU" IDS="AFF1">
    <MUP.FNMS>Bill</MUP.FNMS><MUP.SNM>Olivier</MUP.SNM></MUP.Creator>
<MUP.Creator SCHEME="MUP.AU" IDS="AFF1">
    <MUP.FNMS>Oleg</MUP.FNMS><MUP.SNM>Liber</MUP.SNM></MUP.Creator>
<MUP.Creator SCHEME="MUP.AU" IDS="AFF2">
    <MUP.FNMS>Paul</MUP.FNMS><MUP.SNM>Lefrere</MUP.SNM></MUP.Creator>
<MUP.Creator SCHEME="MUP.AFFS">
    <MUP.AFF ID="AFF1">UK IMS Centre, University of Wales, Bangor, UK</MUP.AFF>
    <MUP.AFF ID="AFF2">UK IMS Centre, The Open University, Milton Keynes, UK
        <EMAIL>o.liber@bangor.ac.uk</EMAIL></MUP.AFF>
</MUP.Creator>
```

The authors are linked to their affiliations using identifiers captured as XML attributes. An author may have more than one affiliation indicated by a list of identifier references within its '*IDS*' attribute.

Further down the XML file, following the keywords, abstract, publisher name, some date, type and format information and identification of the full article file, is the bibliographic citation information for the article and the copyright:

```
<DC.Identifier SCHEME="DCCITE">
    <JournalTitleFull>International Journal of Electrical Engineering
Education</JournalTitleFull>
    <JournalTitleAbbreviated>IJEEE</JournalTitleAbbreviated>
    <Chronology>January 2000</Chronology>
    <JournalVolume>37</JournalVolume>
    <JournalIssueNumber>1</JournalIssueNumber>
    <JournalPages>26-37</JournalPages>
</DC.Identifier>
<DC.Relation.IsPartOf SCHEME="SICI">
    0020-7209(20000101)37:1</DC.Relation.IsPartOf>
<DC.Rights>Copyright &copy; Manchester University Press 2000</DC.Rights>
```

2.2 Article Metadata Display

The MUP electronic journals application processes the base XML files dynamically to display 'Tables of Contents' and article metadata to the end-user as HTML. When an end-user requests viewing of an article's metadata, including its abstract, the XML article metadata file is converted into HTML 'on-the-fly' using a bespoke OmniMark program. For the above example extract the end-user would see, in Arial/Helvetica font, with the title and bibliographic citation information in blue:

> **Specifications and standards for learning technologies: the IMS project**
>
> Author(s): Bill Olivier[AFF1], Oleg Liber[AFF1], Paul Lefrere[AFF2]
> [AFF1]UK IMS Centre, University of Wales, Bangor, UK
> [AFF2]UK IMS Centre, The Open University, Milton Keynes, UK
>
> Journal: International Journal of Electrical Engineering Education
> ISSN: 0020-7209

This is followed by the abstract and keywords, thus changing the display ordering from the order of the elements in the XML file, and finally the copyright:

> Copyright © Manchester University Press 2000

The HTML page displayed contains a link to the PDF file of the full article. This is available for viewing by journal subscribers only, authentication being by IP-address checking. The article metadata and abstracts may be freely viewed.

Changes to the layout or formatting of this HTML display for all articles within the electronic journals application require simply a change to the OmniMark conversion program. If the article metadata files were stored as HTML a change to the display requirements would necessitate editing all the existing article metadata files within the application.

2.3 The XML Display Method

At the time this electronic journals application was developed few web browsers made any attempt to display XML. Now the latest versions of Microsoft Internet Explorer (IE5) and Netscape (6) will display XML, but many end-users are still using older versions of web browsers which require HTML. Definitions of the XML stylesheet and formatting languages have only recently been stabilised. For these reasons, it was decided to use a bespoke program to convert XML files to HTML 'on-the-fly' for web browser display. OmniMark was chosen as a programming language in preference to Perl, or a more general programming language, because it is SGML/XML aware with an in-built parser, allowing rules to be written specific to elements within the XML DTD.

2.4 Multiple Languages

Although all of the articles within this electronic journals application are written in English, one journal, the 'International Journal of Electrical Engineering Education', provides the article titles and abstracts additionally in French, German and Spanish. These are captured in the article metadata XML using a '*language*' attribute for the *Title* and *Abstract* elements, with multiple instances of these elements allowed. For example:

```
<DC.Title LANGUAGE="en">
Specifications and standards for learning technologies: the IMS project
</DC.Title>
<DC.Title LANGUAGE="fr">
Sp&eacute;cifications et normes pour technologies de formation: le projet IMS
</DC.Title>
<DC.Title LANGUAGE="de">
Spezifikationen und Normen f&uuml;r Lerntechnologien: das IMS Projekt
</DC.Title>
<DC.Title LANGUAGE="es">
Especificaciones y est&aacute;ndares para las tecnolog&iacute;as de la
ense&ntilde;anza: el proyecto IMS
```

```
</DC.Title>
```

This would be seen by the end-user as:

> **Specifications and standards for learning technologies: the IMS project**
> **Spécifications et normes pour technologies de formation: le projet IMS**
> **Spezifikationen und Normen für Lerntechnologien: das IMS Projekt**
> **Especificaciones y estándares para las tecnologías de la enseñanza: el proyecto IMS**

At present, non-keyboard characters such as 'é' or 'α' are maintained in the base XML files as SGML entities, 'é' and 'α'. Those which cannot be displayed directly in HTML are suitably converted when the HTML is generated. The advantage of keeping special characters as SGML entities in the base XML files is that this is a neutral form from which they may be converted as necessary for any other, including future, applications which make use of the data. Future versions of this electronic journals application will use Unicode [15] for the display of non-keyboard characters. This change will require a change to the display program only and not to the XML metadata files for all the articles.

3 Data Interchange Formats

Increasingly journals publishers are supplying their article metadata, and in some cases also the full articles, to external agencies such as aggregators and abstractors. Much of this data interchange currently is in SGML format and generally requires the SGML to be conformant with prescribed DTDs. In a utopian future, the data conversion work required for this data interchange could be simplified if XML were used as a standard electronic data interchange format, particularly if interoperable XML schemas and DTDs were standardised within the journals publishing sector. XML, which was originally conceived as a means of exchanging SGML documents over the World Wide Web, is becoming the standard data interchange format for electronic business transactions. MIMAS have worked with two publishers to format their article metadata for supply to *ingenta* [16], and also with a publisher to deliver metadata to PubMed (National Library of Medicine) [17].

3.1 A Common DTD

Where a publisher has data produced by several typesetters, different journals may have article metadata which conforms to different DTDs. Some journals may have the article header information only in SGML, with the full articles in another format such as PDF, whereas others may be typeset as full article SGML from which the article metadata may need extracting for supply to an external agency. For Oxford University Press (OUP) [18] to supply their article metadata, including some back data, to *ingenta*, a common DTD, chosen by OUP, was required. OmniMark programs were written by MIMAS which converted SGML article headers, originally supplied according to several DTDs, into SGML which conformed to a new common DTD. Some of the original DTDs were skeletal, so information such as journal identifiers, ISSN numbers and standard format cover dates had to be added to the SGML during conversion. Following the recommendation by MIMAS to use the OmniMark programming language for this task, and following OmniMark familiarisation sessions held by MIMAS for two OUP staff, this method of SGML conversion was adopted by OUP and further development was continued by them in-house.

3.2 A Data Supply Index

CABI *Publishing* [19], an applied life science publisher of books, journals and electronic products, supply the SGML article metadata of their primary full text journals, with the full articles in PDF and HTML, to *ingenta* for hosting. The standard CABI DTD is used for this data supply, but it required some updating to include necessary information for web publishing by *ingenta*, with the consequent updating of the SGML articles. *ingenta* require article headers only in SGML, so these are extracted from the full article SGML files for supply. An *ingenta* 'manifest' file, an SGML file which provides an index of the delivered files, is required with a supplied journal issue. This 'manifest' file lists the articles contained in the issue, in page number order, along with a prescribed-format *ingenta* identifier. For each article, the 'manifest' file indicates the PDF file for the full article. If the article is also supplied as HTML, this file is detailed in the 'manifest' file along with all its associated graphics files. The 'manifest' file for a journal issue is generated by an OmniMark program which processes all the SGML articles in that issue, generating the 'manifest' file entries for each one, updating the SGML article metadata with any missing bibliographic citation information, and extracting the SGML header as a separate file. An example extract of the data interchange *ingenta* index file, listing only one article is:

```
<!DOCTYPE MANIFEST SYSTEM "manifest.dtd">
```

```
<MANIFEST>
  <ISSUE NAME="infobike://CABI/BJN/1999/00000081/00000005">
    <ARTICLE NAME="infobike://CABI/BJN/1999/00000081/00000005/art00005">
      <HEADER NAME="BJN81349hd.sgm">
      <BODY NAME="BJN81349.pdf" TYPE="application/pdf">
      </BODY>
      <BODY NAME="BJN81349.htm" TYPE="text/html">
        <ADDFILE NAME="images/BJN81349f1.gif" TYPE="image/gif">
        <ADDFILE NAME="images/BJN81349f2.gif" TYPE="image/gif">
      </BODY>
    </ARTICLE>
    <ARTICLE NAME="…">…</ARTICLE>
  </ISSUE>
</MANIFEST>
```

Although this 'manifest' index file is in SGML rather than XML format, because it is of a simple structure its translation to XML would not be difficult. The main change would be to add end tags for empty elements, so the element specifying the SGML header file would become in XML:

```
      <HEADER NAME="BJN81349hd.sgm"/>
```

3.3 An Aggregated Delivery File

CABI *Publishing* additionally supply their article metadata to National Library of Medicine, PubMed. PubMed require a single SGML file containing the article metadata for all the articles in a single journal issue. This SGML must conform to the PubMed DTD. The PubMed supply file is generated by an OmniMark program which processes all the SGML articles in the issue to extract the required article metadata elements for each article and aggregate them into a single delivery file, in page number order, within the issue. Again, the PubMed supply file is SGML rather than XML, but its structure is simple enough for future direct translation into XML, as can be seen from the following abbreviated extract.

```
<ArticleSet>
  <Article>
    <Journal>
      <PublisherName>CAB International</PublisherName>
      <JournalTitle>Bull Entomol Res</JournalTitle><Issn>0007-4853</Issn>
      <Volume>89</Volume><Issue>1</Issue>
      <PubDate><Year>1999</Year><Month>02</Month></PubDate>
    </Journal>
    <ArticleTitle>The biology and ecology of the large pine weevil…
    </ArticleTitle>
    <FirstPage>3</FirstPage><LastPage>16</LastPage>
    <AuthorList>
      <Author><FirstName>S</FirstName><MiddleName>R.</MiddleName>
        <LastName>Leather</LastName>
        <Affiliation>Silwood Centre for Pest Management</Affiliation>
      </Author>
      <Author>…</Author>
    </AuthorList>
    <Abstract>
      The biology and pest status of Hylobius abietis Linnaeus in Europe…
    </Abstract>
  </Article>
  <Article>…</Article>
</ArticleSet>
```

4 XML for Full Articles

Developing a full article XML DTD appears to be a more involved task than developing an article header DTD such as the one for the Manchester University Press electronic journals application described above. CABI *Publishing* required their full article DTD to be XML-compliant, in addition to other enhancements. Because of syntax differences, SGML will not parse against an XML DTD and vice versa. In order to make the CABI DTD XML-compliant, an XML version of the DTD was produced and validated using James Clark's *sx* parser. The indicated changes were then transferred to the SGML version of the DTD.

4.1 DTD Changes

Most of the changes required to make a full article DTD XML-compliant are 'nitty-gritty' syntax changes. The changes made to the DTD include:

- All end tags are mandatory and tag minimisation is not allowed in the DTD.
  Thus: `<!ELEMENT pnm – o (#PCDATA)>`
  would become in an XML-compliant DTD: `<!ELEMENT pnm - - (#PCDATA)>`
  and in an XML DTD: `<!ELEMENT pnm (#PCDATA)>`
- Comments inside declarations are not allowed. So
  `<!ELEMENT pnm - - (#PCDATA)--this is the publisher name-->`
  should become:
  `<!ELEMENT pnm - - (#PCDATA)> <!--this is the publisher name-->`
- Basic character data content, *#PCDATA*, must occur at the extreme left of mixed content declarations. Thus
  `<!ENTITY % data "i | b | #PCDATA">`
  must become `<!ENTITY % data "#PCDATA | i | b">`
  Nested entity definitions need adjusting so that *#PCDATA* is always at the left after expansion. Brackets and repetition operators within entity definitions are problematic, *#PCDATA* not being allowed within a nested model group. So
  `<!ENTITY % data "(#PCDATA | i | b)*"> <!ELEMENT jtl - - (%data;)>`
  had to be changed to:
  `<!ENTITY % data "#PCDATA | i | b"> <!ELEMENT jtl - - (%data;)*>`
- Some SGML attribute types, such as *NUTOKEN* and *NUMBER*, are not allowed in XML. These were changed to *CDATA*.
- Literal defaults for attributes needed quoting:
  `<!ATTLIST table position (fixed | float) "float">`
- Exclusions are not allowed in XML, so the following, which indicates that a table may not contain another table anywhere, is not valid in XML: `<!ELEMENT table - - (title?, tgroup*) – (table)>`
- The repetition operator was changed from '+' (one or many) to '*' (zero or many) for repeatable elements which could contain just plain *#PCDATA*, for example: `<!ELEMENT collab - - (%data; | orf)*>`
- *PUBLIC* identifiers indicating sub-DTDs are not allowed in XML, so these were changed to *SYSTEM* identifiers indicating system file paths in the XML version of the DTD. This seems to make the XML less easily portable.
- A standard default declaration is used in XML in place of the SGML Declaration. This shouldn't cause a problem, but some maximum values had to be increased in the SGML Declaration for the XML-compliant SGML version of the DTD to account for changes to entity nesting and repetition operators.

4.2 Data Changes

Most of the above changes do not affect the marked-up article data, but some changes would also be required to the data files if a move were made from SGML to XML. These include:

- XML is case-sensitive so all case must be consistent. This is not valid XML: `<PNM>MIMAS</pnm>`
- Empty elements require end tags, for example `<orf rid="a1">` should become `<orf rid="a1"/>`.
- All end tags are mandatory, but this requirement will already be imposed by the XML-compliant version of the SGML DTD.
- All attribute values should be quoted, for example '`align=center`' should become '`align="center"`'.

From this experience it seems that converting a full text SGML DTD to an XML-compliant one is a simpler task than maybe envisaged. Most of the changes were syntactical ones found using an XML parser.

5 Data Production Workflow

As journals publishers supply their data to increasing numbers of external agencies in different formats, managing workflow during production becomes more important. The data for the articles in a journal issue has to undergo several transformations to provide these different supply formats, each step involving quality checking and possible repeat. This probably follows tasks such as initial, preflight quality checking, and some valuing adding, for instance including links from the bibliographic references in the articles to the corresponding full text articles or to abstracting services such as Medline [17]. Some of the steps in the process may be performed automatically, but some, such as quality checking, will be manual operations. This process must usually be done within a tight timescale to ensure that journal issues are published electronically in a timely fashion.

As the production process becomes more complex workflow management becomes a necessity so that the progress of a journal issue through this system can be tracked, without reliance on individual staff knowledge. Maintaining journal article data in XML will assist in managing this workflow process, because XML being a neutral, non-proprietary format is suitable for the data interchange between different states in the workflow process, for maintaining back-up data at each stage, and as a medium-independent format for content archiving [20]. XML-based tools are appearing in the market-place which provide a complete electronic publishing environment including workflow management.

It is probable that, as electronic journals publishing moves towards being XML-based, there will be a requirement for retrospective conversion to XML of existing data currently held in HTML. Existing electronic journals applications which operate using static HTML files will not lend themselves to future developments where capture of the information content rather than just its display format is necessary. Retrospective conversion of existing HTML data to XML is not trivial, and is potentially labour intensive. Although there are some current tools which claim to be able to perform this conversion, most are Microsoft Windows based, and would not be suitable for accurate batch conversions of large amounts of data.

6 Future Developments

There have been many recent developments in the definition of the accompanying standards to the basic XML language which make up an integrated XML framework. These include the XML stylesheet standard [21] made up of a transformation language (XSLT) to prepare XML objects for display and a display formatting language (XSL-FO) for visual styling, the XML linking (XLink) [22] standard, and XML Schemas [23]. Until now the definitions of these elements of XML have not been stable enough for practical use, hence the lack of available XML tools, but this situation appears to be improving.

The XML stylesheet transformation and display formatting languages allow a web publisher to specify and customise the required layout for documents. Different stylesheets may be easily employed when displaying on different types of devices, providing multiple views of an XML document. For instance, less information would be displayed on mobile devices with limitations on both screen size and bandwidth. New versions of web browsers which understand XML stylesheets will allow for direct display of XML documents and thus distributed style processing. Extensions to web servers are appearing, such as the Apache Cocoon project [24], which will process XML stylesheets on the server-side, allowing XML documents to be displayed by older web browsers. But using a bespoke program for XML display, such as the one used within the Manchester University Press electronic journals application will still be a viable alternative, especially if an XML-aware language such as OmniMark is used.

XML linking (XLink) will be used to provide linking between documents. XML linking provides more functionality than the simple, single HTML hypertext link. It provides a means to specify any point in a document as a link target, this part of the language (XPath) [25] also being utilised by the XML stylesheet standard. XLink allows linking to multiple places with dynamic resolution. XLink specifications are held in separate files, also in XML format, thus separating the links indicated within a document from the addresses. Adopting XML as a format for web publishing will open up future linking possibilities unavailable, or difficult to implement, in HTML.

The original definition of XML was based on SGML, with document structure imposed by a DTD, although 'well-formed' XML documents, with no DTD, are also legal. More recent developments of the XML standard have introduced XML Schemas. An XML Schema defines data typing information as well as providing a template for the document structure. So it is possible, for instance, to insist that the content of a particular element is always an integer, such that '`<StartPage>12</StartPage>`' would be legal but '`<StartPage>Twelve</StartPage>`' would not (assuming page numbers are always integers in a particular application), or that a number is always within a defined range. As another example, it may be a requirement that a date field is always of the form '*yyyy-mm-dd*' where '*yyyy*' is a four digit year, '*mm*' is a two digit month and '*dd*' is a two digit day.

7 Why Use XML?

XML provides a simple but flexible, standards-based format which captures the information content of a document separately from any styling and display instructions. This means a document creator can focus on its content, with display decisions made later, the actual display to an end-user being dynamic. Because the document does not include display information it is suitable for use in many different contexts and media, using simply created profiles. Visual display possibilities and web publishing functionality far exceed those provided by HTML. Many electronic journals publishers are already using SGML, experience which will still be valuable if migration to XML is chosen. The conversion of a full article SGML DTD to XML was relatively unproblematic.

A requirement to use a standard interoperable format for electronic publishing will become increasingly important as academic publishing moves towards becoming 'joined-up' with the global linking of content through the World Wide Web implied by initiatives such as CrossRef [26], developed from the DOI-X project [27], which utilises Digital Object Identifiers [28]. Increasing requirements for publishers to supply their article metadata to several external agencies necessitate using a standard electronic data interchange format. XML, with its endorsement by the World Wide Web Consortium who are now developing an XML Fragment Interchange standard [29], appears to provide the future standard format for many such applications. The open, non-proprietary standard of XML, with its underlying format neutral data, provides flexibility for use across multiple applications. However, to enable interoperability within the journals publishing sector a co-operative agreement on a common DTD or schema will become necessary.

Moving towards using XML for electronic journals publishing should provide the flexibility to allow for future applications. Tools which process XML, covering the whole publishing production process from content creation to web publication, are starting to appear, of varying cost. Because of the current interest in XML by the 'e-business' community, and because XML is simpler to understand and process than SGML, it seems that more tools will become available than exist to process SGML. Although this is apparently good news, many of these tools are not interoperable making them difficult to integrate into a production environment, some do not have constituent parts purchasable separately, and it is difficult to predict which tools will survive in the longer term. Some very useful XML tools are free or shareware, but their future maintenance or quality is unknown. So care would be advised in choosing XML tools at the present time while standards are still developing.

The significant disadvantage to using XML is the relative instability of the standards, some of which are still at the working draft stage. Some earlier suggested extended features of XML have now been dropped, making adopting XML sometimes feel like trying to 'hit a moving target'. So care must be taken in adopting such features while they are immature. However, the basic XML language, which is derived from the more complex SGML, is stable.

XML appears to be the future for electronic journals publishing. Although some shortcomings of XML have been highlighted, these should be overcome as standards stabilise. As the examples given in this paper have shown, it has been possible to successfully implement a 'live' electronic journals application using XML as a format to hold the article metadata, while the XML standard has been evolving.

References

1. XML. http://www.w3.org/XML
2. XML information collected by the DIFFUSE project. http://www.diffuse.org/xmlguide.html
3. Goldfarb, CF.: The SGML Handbook. Oxford University Press.
4. Manchester University Press. http://www.manchesteruniversitypress.co.uk
5. Electronic Publishing at MIMAS. http://epub.mimas.ac.uk/
6. Dublin Core Metadata. http://purl.org/dc
7. MacIntyre, R., Tanner, S.: Nature – a Prototype Digital Archive. *International Journal on Digital Libraries* (2000), Springer-Verlag, (Scheduled for 2000(2))
8. Apps, A., MacIntyre, R.: Metadata for the Nature Digital Archive. http://epub.mimas.ac.uk/natpaper.html
9. Apps, A., MacIntyre, R.: The SuperJournal Project: Data Handling Using SGML. *Proceedings of the Third ICCC/IFIP Conference on Electronic Publishing, Ronneby, Sweden, 10-12 May 1999*, ICCC Press
10. Olivier, B., Liber, O., Lefrere, P.: Specifications and standards for learning technologies: the IMS Project. *International Journal of Electrical Engineering Education* 37(1) (January 2000), Manchester University Press
11. Example XML Article Header. doi://10.1060/IJEEE.2000.003 (http://dx.doi.org/10.1060/IJEEE.2000.003)
12. SSSH, Simplified SGML for Serial Headers, DTD. http://www.oasis-open.org/cover/gen-apps.html#sssh
13. OmniMark Technologies. http://www.omnimark.com
14. sx (XML parser and validator). http://www.jclark.com/sp/sx.htm
15. Unicode. http://www.unicode.org
16. ingenta. http://www.ingenta.com
17. National Center for Biotechnology Information, National Library of Medicine, PubMed and Medline. http://www.ncbi.nlm.gov/PubMed/
18. Oxford University Press. http://www.oup.co.uk
19. CABI *Publishing*, the publishing division of CAB International. http://www.cabi.org
20. Beebe, L., Meyers, B.: Digital Workflow: Managing the Process Electronically. *The Journal of Electronic Publishing* 5(4) (June 2000). http://www.press.umich.edu/jep/05-04/sheridan.html
21. XML stylesheets. http://www.w3.org/Style/XSL
22. XML Linking (XLink). http://www.w3.org/TR/xlink
23. XML Schemas. http://www.w3.org/XML/Schema.html
24. Apache Cocoon Project. http://xml.apache.org/cocoon
25. XPath language for addressing parts of an XML document. http://www.w3.org/TR/xpath

26. CrossRef. http://www.crossref.org
27. Atkins, H., Lyons, C., Ratner, H., Risher, C., Shillum, C., Sidman, D., Stevens, A.: Reference Linking with DOIs. *D-Lib Magazine*. 6(2) (February 2000). doi://10.1045/february2000-risher
28. Digital Object Identifier. http://www.doi.org
29. XML Fragment interchange standard. http://www.diffuse.org/xmlguide.html#fragments