XML-centric workflow offers benefits to scholarly publishers

Alexander B. **Schwarzman** <ssschwarzman@agu.org>

Hyunmin **Hur** <hhur@docsdoc.com>

Shu-Li **Pai** <spai@agu.org>

Carter M. **Glass** <cglass@agu.org>

Abstract

During the transitional paper–electronic period, a nonprofit STM publisher faces the challenge of publishing a scientific journal in both digital and analog formats while controlling costs and ensuring consistency between electronic and printed representations of an article. This must be achieved, as its sophisticated constituency expects a constantly expanding range of information products and services. In a few short years the American Geophysical Union (AGU) leapfrogged from the paste-up era, when authors prepared their own “camera-ready copy” to be pasted on boards for a printer, to the age of XML, when an article marked up in accordance with a custom-designed DTD serves both as a version of record and a source for generating PDF and HTML article representations. Bibliographic and reference metadata are then extracted from the XML article instance into a relational database, which serves as a basis for generating online and print access mechanisms/products, including various tables of contents and author and subject indices.

Maintaining metadata in a database has allowed AGU to offer its journal subscribers a number of innovative information products in electronic form, among them a “virtual journal” that cuts across the boundaries of a traditional printed periodical. The database also serves as a source of metadata exchanges with A&I services. As a result of ongoing interaction with the publishers' consortium CrossRef, the metadata database stores a continuously growing list of cited and citing publications' DOIs, thereby enabling implementation of dynamic linking of referenced materials as well as inbound and “forward” (i.e., “cited by”) linking.

As a result of implementing an XML-centric workflow and a suite of Java and XSLT applications, AGU has increased article production capacity and shortened publication time while reducing costs and labor. In addition, using the XML-tagged article as a source for all derivative formats, including print, coupled with automation of the production process, has enabled AGU to ensure consistency of style across its publications regardless of format and to improve accuracy of bibliographic and reference metadata. Such an approach has also allowed authors to concentrate solely on producing scientific content, leaving to the publisher the responsibility for presenting their papers in a variety of formats and for enriching the works' usability with value-added electronic features, such as reference and citation linking, multimedia files, article grouping by subsets, special sections, and index terms.

Architectural and workflow models are presented; rationales for selecting particular technologies (DTD, Relational DB, XSLT, Java) and representational formats (PDF, HTML) are explained; problems in converting legacy data from multiple sources are addressed; custom-written software to ensure strict datatyping and enforce dependencies, which cannot be provided for by the W3C

XML Schema nor by validating parsers is presented; and solutions for dealing with special characters, including math, are proposed. Lessons learned are shared, and benefits to a scholarly publisher are enumerated.

Table of Contents

1. Introduction

American Geophysical Union (AGU) publishes 14 high-impact English language journals in the area of geophysics, with approximately 4500 articles annually. A nonprofit multidisciplinary science organization with 41,000 members from 130 countries, AGU was established in 1919, and since then has supplied an organizational framework within which geophysicists have created the programs and products needed to advance their science. AGU now stands as a leader in the increasingly interdisciplinary global endeavor that encompasses the geophysical sciences.

AGU's activities are focused on the organization and dissemination of scientific information in the interdisciplinary and international field of geophysics. The geophysical sciences involve four fundamental areas: atmospheric and ocean sciences, solid-Earth sciences, hydrologic sciences, and space sciences.

AGU serves its diverse membership through a broad range of publications, meetings, and educational and other activities that support research in the Earth and space sciences. Many AGU members are involved in research of direct societal concern such as global warming, climate change, ozone depletion, natural hazards, water supply and quality, and other environmental factors. AGU's membership includes many of the world's foremost geoscientists from industry, academia, and government.

This paper will show how implementing an XML-centric solution brought AGU into the electronic publishing era. Each section will describe architectural or workflow decisions made by the e-publishing team at AGU, provide justification for those, and discuss alternatives that may be considered by publishers who wish to follow a similar route.

2. Manuscript life cycle

Figure 1 illustrates a typical manuscript life cycle. In the process of scholarly publishing, a typical manuscript goes through the following life cycle stages: submission, review, revision, acceptance, production, access/dissemination, and archiving

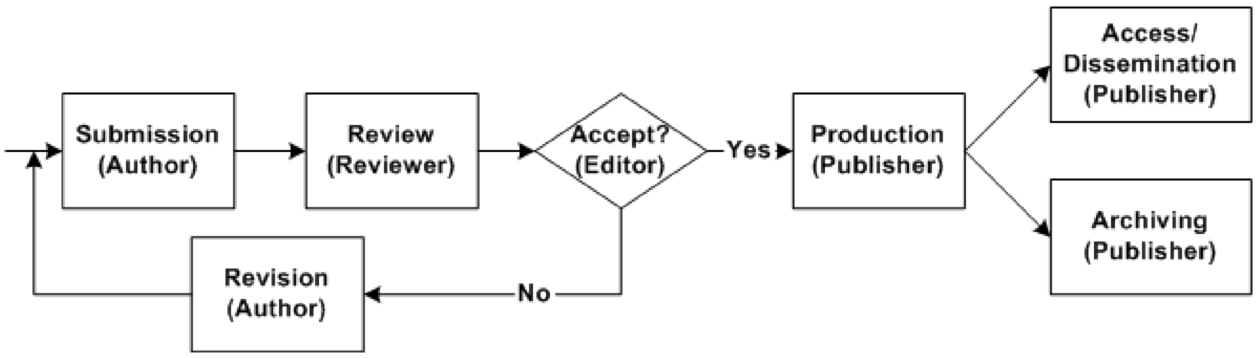

Figure 1. Manuscript life cycle.

- submission and review
    - submission (author)
    - review (anonymous reviewers)
    - revision (author)

  - acceptance (editor)
- production
  - copyediting (copy editor)
  - proofing (proofreader)
  - proofing (author)
  - correcting (vendor)
  - publishing (production coordinator)
- dissemination
- access
- preservation/storage

While AGU has introduced radical changes in how a manuscript is handled at all stages of its life cycle, in this paper we will concentrate on the post-acceptance part of the publishing process.

3. AGU manuscript production modes in 2001

In 2001, when the project fully commenced, AGU employed several manuscript production modes:

- pasting-up author-prepared camera-ready copy of the manuscript
- typesetting the manuscript
- SGML markup of the electronic manuscript file.

3.1. Camera-ready copy

The majority of manuscripts came as so-called "camera-ready copy" or CRC. The authors, upon acceptance and copyediting of their manuscript, would submit for production a CRC prepared using a word-processing system, such as MSWord, WordPerfect, or LaTeX, in accordance with AGU editorial style. Production staff pasted CRC onto boards and sent them to a printer, who produced pages and assembled printed issues.

The chief advantage of a CRC publishing model is an economic one: since it was the author's responsibility to prepare a production copy, the publisher incurred no typesetting expenses, which gave the nonprofit publisher a significant advantage over its commercial competitors. In the era of escalating journal subscription prices and shrinking library budgets, that was a very important consideration.

However, CRC advantages were far outweighed by its numerous drawbacks:

- electronic copy did not exist; as a result, scientific content could not be reused or repurposed
- the quality of the published product was inconsistent because the authors themselves prepared the final production files
- issue tables of contents, author and subject indices, and AGU's own bibliographic database Earth and Space Index (*EASI*) all had to be created manually; the process was laborious, long, and prone to errors since bibliographic metadata had to be rekeyed; and still *EASI* did not contain abstracts

- not having an article in electronic form meant that in order for various external abstracting and indexing (A&I) services, such as *INSPEC* or *SPIN*, to process AGU article metadata, printed issues of AGU journals had to be mailed, and metadata reentered. Not only was that process error-prone, it also resulted in substantial delay between issue publication and the time when A&I information became available.

3.2. Typeset manuscripts

Two AGU journals were wholly typeset. AGU sent accepted manuscripts to a vendor, who used a proprietary system *XyVision* to produce journal issues. Along with a PDF and HTML, one of the deliverables was an SGML article file marked up in accordance with a variation of the ISO 12083 journal article DTD. It is important to note that a manuscript typeset in *XyVision* served as a source for PDF, HTML, and SGML formats, and if a correction had to be made, sometimes it was made separately in all three formats, which created a potential for discrepancies among various article representations. Authors of other journals who did not want to prepare CRC had an option of having their manuscript typeset, but they had to pay extra for that service.

3.3. Odd cases

In 1997, AGU introduced an electronic-only journal, *Earth Interactions*, that was marked up in SGML in accordance with its own DTD. In 1999, AGU introduced another electronic-only journal, *Geochemistry, Geophysics, Geosystems*, which was produced by marking up its manuscripts with a variation of the ISO 12083 SGML DTD. For both journals, PDF and HTML representations were generated from the SGML source and posted online. In 1997, *Geophysical Research Letters* gave its authors an option to have, in addition to a printed format, online versions of their article: those choosing that route had to submit their manuscript in LaTeX, which then was converted to PDF and HTML using an in-house process. In 2000, *Global Biogeochemical Cycles* started using newly written AGU Article SGML DTD to markup some (but not all) of its accepted manuscripts; for those manuscripts, SGML was used as a source file to generate HTML, low-resolution (on-screen) PDF, and high-resolution PDF for inclusion into printed issues.

By the late 1990s it became exceedingly clear that CRC publishing model became a dead-end. In order to stay competitive in a dual paper–electronic publishing environment, AGU had to develop a unified electronic publishing process.

4. Project requirements

Accordingly, AGU leadership launched an electronic publishing program and formulated the following requirements:

- multiple outputs: journal article had to appear in multiple formats: print, PDF, and HTML
- search: both articles' metadata and their full text had to be searchable
- linking: bibliographic references' sources, external datasets, and components within an article had to be hyperlinked
- dynamic content: authors had to be able to include multimedia objects, such as videos or animations, into their articles
- cross-journal products: in the interdisciplinary world of Earth and space sciences, readers had to be able to create their own personalized collections of articles cutting across journals
- preservation of scientific content: scientific content had to be preserved for the foreseeable future in a readable, preferably nonproprietary, format.

Implementation of such an agenda required a radical overhaul of the cut-and-paste shop that AGU was then. An unexpected advantage of the CRC model was that AGU was not heavily invested into any typeset system, and therefore could leapfrog into electronic production easier than those who were. The situation was not unlike that of some of the third-world countries, which did not have a regular phone service and could therefore start installing wireless phone networks right away.

5. XML-centric architecture and workflow

After analysis, the AGU electronic publishing team concluded that the best way to satisfy the requirements was to design and implement an SGML-centric production system. An article marked up in SGML/XML would serve as the version of record; PDF, print, and HTML article representations would be derived from it, as would other products and services, such as online and printed tables of contents, metadata and full-text databases, cross-journal products, and author and subject indices.

Figure 2 shows the workflow and architecture of the implemented system.

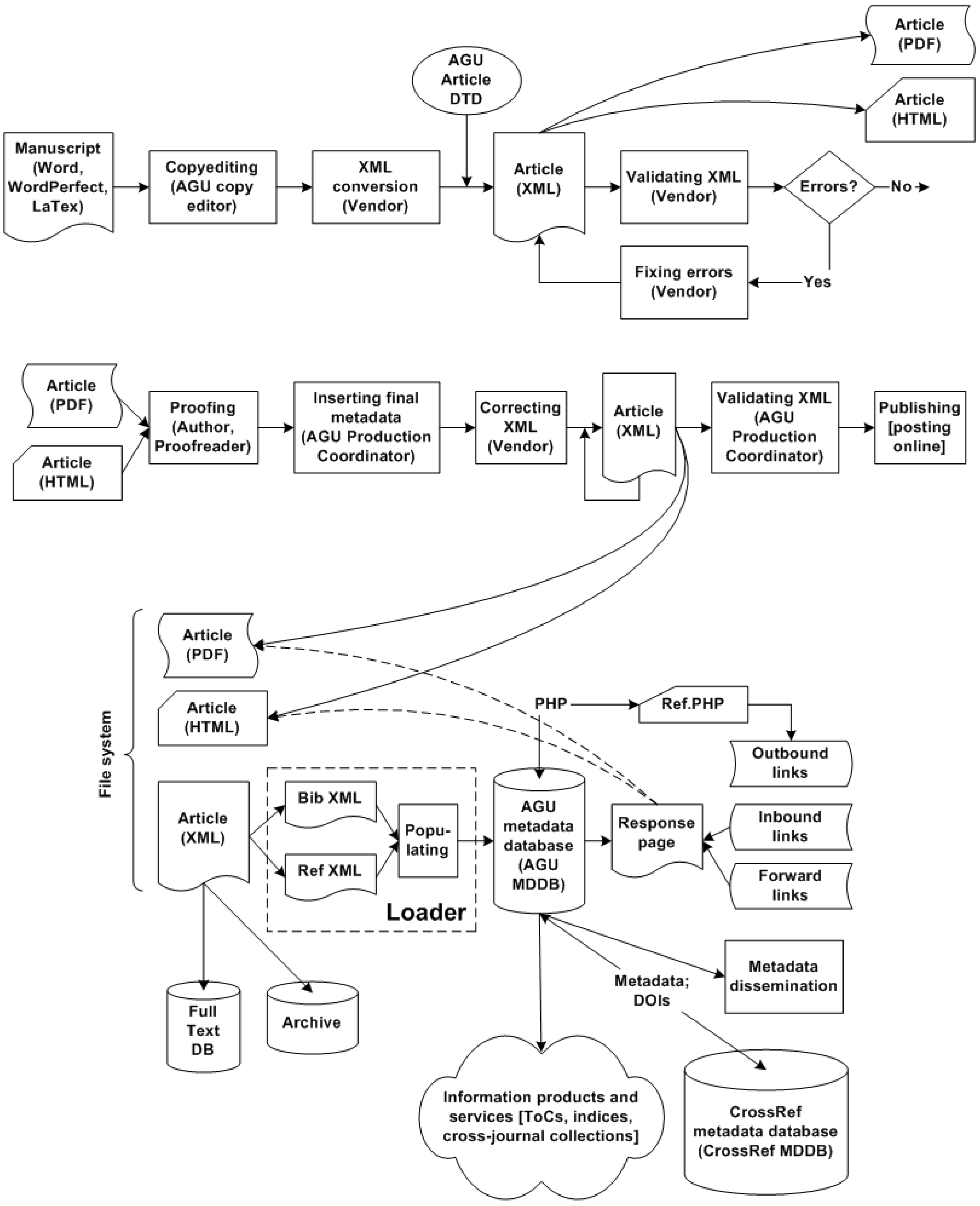

Figure 2. System architecture and workflow.

Workflow

- author submits a manuscript in allowed format (MSWord, WordPerfect, or LaTeX)

- copy editor edits in author-submitted format, not in XML (see Section 6.3, “Copyediting in author-submitted format”)
- vendor converts native format to the AGU Article DTD-compliant XML, runs additional checks using an AGU Validator (quality assurance tool, see Section 6.4, “Validator”), and uses composition system to generate high- and low-resolution PDF and HTML
- author and proofreader proof PDF and HTML, copy editor incorporates corrections, and production coordinator assigns final metadata
- vendor corrects XML, validates it, and generates new PDF and HTML
- proofreader checks final corrections, production coordinator validates final XML and uploads all files to the server; final quality control check is performed on PDF and HTML. At this stage, no additional XML check is required, since it had been validated previously and served as source of all content[1] .
- bibliographic and reference metadata are extracted from XML file into AGU metadata database (AGU MDDB), and full text is deposited into search engine repository
- metadata master file is constructed by pulling metadata out of MDDB. XSLT transformations are used to generate online tables of contents and article “response pages”, deposit bibliographic metadata to CrossRef, query CrossRef for reference resolution, and send metadata and abstracts to A&I services, etc.
- article files are deposited into the archive

Custom software

- AGU Article XML DTD
- AGU Validator
- XML conversion tool
- AGU Loader (bibliographic and reference metadata extractor and inserter)
- AGU metadata database (AGU MDDB)
- reporting, linking, and metadata dissemination modules
- full-text database and search engine

6. Design decisions: justification and alternatives

6.1. Article source

6.1.1. SGML vs. XML

The structure of a scholarly article could be adequately expressed in XML; it did not require any SGML-specific features, such as CONCUR, SUBDOC, inclusions, or exclusions. Therefore, from the very beginning the journal article DTD was written in XML-compliant SGML, and the switch to an XML DTD was trivial.

[1]If errors are discovered during the final check of PDF or HTML, they are usually so minor that they either do not affect XML at all, or minor adjustments to XML, HTML, and PDF could be made in-house. This happens very rarely, though. In an unlikely event of a major error, the article is pulled out of production and sent back to the vendor for the second correction cycle, where XML is corrected, and PDF and HTML are regenerated out of the corrected XML.

6.1.2. Schema language: DTD vs. W3C Schema and Relax NG

Having conducted a thorough analysis of AGU article types at the time of AGU's transition to electronic publishing, AGU e-publishing team could not find an adequate DTD. The industry-standard ISO 12083 SGML/XML DTD written in a monolithic, as opposed to a modular, fashion attempted to do everything for everybody but could not achieve that goal. Many publishers tried to adapt that DTD for their own needs but AGU chose to write its own AGU Article DTD on the basis of the Requirements. The situation is somewhat different now, in that there exists a well-documented journal article DTD for public use, which has been designed as a suite of XML DTD modules that could be easily customized for the needs of a particular publisher. The DTD was created and is maintained by the National Center for Biotechnology Information (NCBI), a center of the National Library of Medicine (NLM); see http://dtd.nlm.nih.gov/publishing/. Before attempting to write its own DTD, a publisher would be well-advised to explore the option of customizing the NLM journal article DTD.

AGU's transition to electronic publishing took place in 1999, and the W3C Schema did not become a W3C Recommendation until 2001. Therefore, there was no alternative but to use a DTD at the time. At present, a document designer has a choice among several XML schema languages, such as a DTD, the W3C Schema, Relax NG, and Schematron. In choosing a schema language suitable for its needs, a publisher must consider carefully both the structure of the documents to be marked up and the requirements of the processing system. There are many reasons why a DTD may still be an optimal choice for a scholarly publisher [Beck and Lapeyre, 2003], Section 6. Among the most compelling technical reasons to use a DTD is its parameter entity mechanism, which enables modular DTD design. This approach allows for inclusion and interchange of building blocks, such as CALS tables or MathML, some of which may exist only in DTD form, and makes it easier to maintain, customize, and scale a DTD. Other advantages of a DTD are its stability, the availability of processing tools, and consistency of validity checking implementation among them.

There are also practical reasons to use a DTD: the taggers and aggregators that work with the scholarly publishing industry are largely DTD-oriented, and their tools and processing systems are often DTD-driven. Many conversion vendors and custodians of archival repositories also use DTD-centric tools and processes.

Finally, there may be business reasons to continue using a DTD. Using XML character entities might be a nonnegotiable requirement for the purposes of archival readability (see Section 7.5.1, “XML character entities vs. Unicode”), and if a publisher has such a requirement, this cannot be done using any other schema language but a DTD. Some vendors still have their publishing systems/composition engines set up in such a way that they are not fully Unicode-compliant. If a publisher wants to maximize its choices in selecting the most competitive vendor, it may find its options somewhat limited if the XML instances it provides to the vendor do not contain XML character entities. For example, when AGU contracted an otherwise competitive vendor to print annual author and subject indices, AGU provided parsed XML output that contained Unicode points representing special characters. The vendor's composition system was not set up to support Unicode. Since AGU's policy is to use XML character entities to represent special characters in the XML source files, it was trivial to produce the XML file with entities, and the vendor generated the indices without any further problems.

On the other hand, there are some disadvantages to the DTD, chief among them are that a DTD is not in XML syntax and that it lacks strong datatyping. If you, as a publisher, face a decision which schema language to choose, ask yourself the following questions:

- is it a requirement that a publisher's or vendor's processing system deal with a schema in XML syntax?
- is it essential to have strong datatyping and full namespace support for elements and attributes?
- is it important to specify minimum and maximum occurrences of certain elements?
- are the content models similar enough to make use of the mechanism of derivation by extension or restriction?
- are the developers and vendors comfortable with inconsistency of validity implementation in parsers and other processing tools?

If you answer “yes” to some of these questions, then you should seriously consider using the W3C Schema. However, you must be acutely aware of the problems you might face: that there is no consistency in how different parsers check Schema validity, that the Schema is a more complex document, that it is not at all easy to modularize and scale it, to name just a few. Moreover, it is usually the case in scholarly publishing that content models are highly diverse and not derivable; therefore, for these models you will not be able to take advantage of the Schema's “derived by restriction” and “derived by extension” typing mechanism. Furthermore, many elements in scientific journals are of “mixed content” type, where almost all W3C Schema restrictions (e.g., min/max length, regular expressions) are not usable.

A more recent alternative to a DTD and the W3C Schema is Relax NG, which has both an XML and a non-XML (which is more human-readable) syntax. Relax NG combines the intelligibility of a DTD with the strong datatyping capabilities of the W3C Schema, brings back some SGML facilities, such as an “and” operator, and even introduces new operands, such as “interleave”. It is worth noting that in the past year, two major “text” DTDs—DocBook and TEI—have been converted to Relax NG. A publisher who considers using Relax NG must evaluate if its document model requires features offered by this schema language. Ask yourself the following questions:

- is having context-sensitive content models for the same element important? For example, is it essential that the same element (e.g., paragraph) used in the article abstract and body have a different content model? (Of course, you could simply use two different elements as an alternative.)
- is it a requirement to validate documents of several different types using one combined schema? Relax NG has an excellent facility for creating a schema describing the union of several document types
- is it desirable to use some W3C Schema features that Relax NG incorporates, such as strong datatyping?

If your answer is “yes” to some of these questions, then you may want to consider Relax NG. However, Relax NG is not yet as well established as the W3C Schema, not to mention a DTD, and even though the parsers' behavior is more consistent than that for the W3C Schema, the number of available tools is still somewhat limited. You should also be aware that Relax NG (as well as the W3C Schema) does not support XML character entities. Therefore, in order to use them, you must **also** have a DTD, and your processing gets more complex. Nor does Relax NG permit Formal Public Identifiers (FPI) in the XML instances. Hence, if you have a collection of SGML/XML documents that used FPIs and want to continue using FPIs, you will have a problem. You also have to be mindful of the fact that even though Relax NG introduces some useful and convenient features, they are specific to Relax NG; thus when they are used, the resulting schema may not translate neatly into either a DTD or the W3C Schema. As a consequence, you will not be able to utilize interoperability between Relax NG and a DTD or the W3C Schema.

The Schematron approach is still somewhat experimental at the time of writing, and one would be well advised to adopt a “wait-and-see” attitude before choosing Schematron.

To summarize, if a publisher has a valid need for features beyond those available in a DTD, it may consider using an alternative schema language, such as the W3C Schema or Relax NG. Publishing is a conservative industry, and—in our view—only if a publisher has a compelling need to use features not available in a DTD should it adopt a different schema language. Thus if we were making this decision today, we would still opt for a DTD. However, it is likely that when Relax NG gains adequate tool support, it will become a viable long-term schema solution for “text” (as opposed to “data”) content in general, and for scholarly publishing in particular.

6.2. Converting author-submitted manuscript to XML

6.2.1. Using a vendor vs. an in-house process

Since it is unlikely that in the near future authors will submit their manuscripts in XML, a publisher has to decide how the manuscripts will be converted from the author-submitted formats into the DTD-compliant and semantically correct XML that conforms to the publisher's rules and naming conventions. This crucial task can be either outsourced or performed in-house. Two main factors in making such a decision are: (1) how good is the quality of the resulting XML and (2) what are the costs of each alternative.

Not being a text-processor software development shop, AGU wisely chose not to attempt to design its own XML conversion processes. Since at the time of AGU's transition to electronic publishing, there were no off-the-shelf products that could provide ‘text-processor to XML’ conversion of adequate quality, AGU chose to use a vendor. The next section will elaborate on the vendor selection criteria.

At the present time, however, a publisher can choose to license any number of software products that aid in automating manuscript copyediting and redactory processes, and prepare a manuscript for input into a composition system or for electronic delivery in XML. The limitation of the most commercially available products of this kind is that even though they can deal with any format that MSWord can read, that, unfortunately, excludes LaTeX.

Ultimately, if the quality of the resulting XML product is the same, the publisher's choice between outsourcing XML conversion and performing it in-house is determined by what makes more sense economically.

6.2.2. Choosing a vendor

In choosing a vendor, a publisher should examine the process the vendor uses to produce XML. Some vendors use their proprietary composition systems to typeset a manuscript and then generate PDF, HTML, and XML out of it. Vendors with such a workflow must be rejected by publishers who designate XML as the version of record. Instead, only vendors with an “XML-first” workflow[2] must be chosen, so that after the quality of XML is assured, secondary formats, such as PDF, print, or HTML, can be derived from XML.

It must be noted that a tagger (who produces source XML from the author-submitted formats) and a compositor (who produces representational formats, such as PDF and HTML, from XML) are distinct roles, and do not have to be performed by the same actor. For example, a publisher may convert author-submitted manuscripts to XML and then transform it into HTML in-house, and employ a vendor who has a professional composition system only to produce a PDF representation. In AGU's case, a vendor performs both tagger and compositor roles.

Another factor is how well a vendor can deal with the manuscript formats accepted by a publisher. About 55% of AGU manuscripts come in MSWord, about 40% are in LaTeX, and the rest are in WordPerfect. Since AGU decided to mark up math as a LaTeX object within an XML instance, it selected a vendor whose conversion process transforms MSWord's MathType expressions into LaTeX automatically, while converting the nonmath parts of the manuscript into XML. For LaTeX manuscripts, the original math coding as done by the author is left intact, and the rest is converted to XML. A few manuscripts that come in WordPerfect are converted to MSWord first.

Lastly, in selecting a vendor, a publisher should assess how adaptable the vendor will be to the inevitable DTD changes. One would be well advised to pay attention to management principles, processes, and technology employed by the vendor before making a final determination.

6.3. Copyediting in author-submitted format

While some publishers feel that copyediting must be done in XML, AGU has decided to copyedit manuscripts in author-submitted formats. First, the software to edit in XML is very expensive; more expensive still would have been training to make good copy editors XML-savvy. Industry experience shows that it takes man-years of effort to customize XML editing software so that untrained copy editors could use it effectively. Second, copyediting in XML would mean an additional loop with the vendor—the accepted manuscript would have to go there for conversion, then the XML edited at AGU and returned to the vendor for creation of proofs for authors. After collecting authors' feedback, the manuscript would have to go back to the vendor yet again for producing low- and high-resolution PDFs, as well as HTML. While we could easily generate HTML in-house, producing print-quality PDF requires a professional composition system. Had AGU not had a requirement to produce a print-quality PDF, such workflow would have been feasible. As it stands

[2]We use the term “XML-first” to denote that representational formats, e.g., PDF and HTML, are generated from the XML, rather than from a proprietary composition software. This does not mean that a manuscript is converted to XML immediately after submission. As Section 6.3, “Copyediting in author-submitted format” indicates, AGU manuscripts are converted to XML only after they are copyedited in an author-submitted format.

right now, however, copyediting in author-submitted format versus in XML provides substantial savings of both money and time.

6.4. Validator

No matter how good the publisher's DTD is and how thoroughly it is documented, marked up instances must go through extensive checking to ensure quality of the XML markup. Errors in XML may range from DTD noncompliance to “tag abuse” to business rules violation. The fact that an XML instance conforms to its DTD does not necessarily mean that markup is correct or even makes sense. For example, how does one check the following rule: if an article is a commentary on a previously published one (i.e., ‘paper-type’ attribute value is “com”), then (1) the <related-article> element must be present and contain a digital object identifier (DOI) conformant to a publisher's DOI format, and (2) the title of the article must begin with “Comment on …”? If you use certain file, entity, and attribute value naming conventions, then these also must be checked using external means. For example, AGU file naming convention is that an XML article file name should be the same as the DOI suffix in lower case. Validation of that kind is, obviously, beyond the means of any XML parser. If a publisher uses a DTD, as opposed to a schema, then certain datatypes must be checked as well.

Unless a publisher implements an XML quality assurance procedure, it cannot be certain that the XML it receives is semantically correct. The problem is compounded if the publisher outsources all XML-related production issues to a vendor and/or an aggregator. If quality assurance steps are not in place, a publisher may realize all too late that the underlying source XML for thousands of articles, which looked fine in PDF, HTML, or print, is semantically wrong and cannot be reused or repurposed [Rosenblum and Golfman, 2001], p.61:

> 1. SGML that is archived and not put to immediate use online will probably have errors because there is no quality checkpoint.
>
> 2. No matter how complete the documentation, strict DTD interpretation can best be insured with software-based quality control tools.
>
> 3. Suppliers will not use quality control tools and fix reported problems without negative feedback loops.
>
> 4. Publishers must actively monitor and enforce quality standards, even after providing tools to suppliers.

In the beginning of electronic publishing at AGU, XML was checked visually, and since this task was ill-suited for a human eye, errors were not uncovered in many instances until later in the process, i.e., at the stage of checking HTML and PDF article representations. In such cases, an article would have to go through an expensive and disruptive “up-stream” workflow loop, whereby the article would have to be sent back to the vendor, who would correct the XML and regenerate the representational formats. AGU's solution was to implement a software application called the AGU Validator. The Validator performs three types of checks: it includes a validating parser that ensures DTD compliance, it checks selected datatypes, and it verifies AGU-specific business rules and dependencies. Examples of datatypes checked are received, revised, accepted, and published dates and DOI prefix and suffix formats. Examples of specific dependencies checked are that a special section[3] article must contain a special section code and title or that a correction must have a ‘paper-type’ attribute value “cor” and a title beginning with “Correction to…” In addition, the Validator is a database application: it connects to AGU metadata database and performs a number of checks, for example, whether a special section title or an index term (subject descriptor) in the XML instance matches the ones stored in MDDB.

All in all, the current implementation of the AGU Validator runs about 100 checks on each XML article instance. It is a Java application that contains a configurable rules file: as an XML article instance goes through production stages,

[3]A special section is a collection of articles on a particular topic. All originally proposed special section papers are supposed to appear in the same printed issue. As a result of electronic publishing, however, related articles published in the future, after the printed issue is released, can be linked electronically to the same special section via metadata stored in MDDB.

it acquires more and more metadata and is subjected to progressively more rigorous validation. The Validator is deployed both at a vendor's site and at AGU, and it is an AGU requirement that any vendor that wants to do business with AGU must use the Validator and deliver only error-free content.

6.5. Metadata Loader and metadata database

The centerpiece of the AGU publishing system is a metadata database. It contains bibliographic and reference information extracted from the article XML instance. After an article XML instance is ready for publication, the Loader harvests bibliographic and reference metadata from it and loads the metadata into MDDB. The database is used for producing all AGU publishing products and services, such as tables of contents, author and subject indices, cross-journal collections, etc. It is also used to deposit metadata to CrossRef citation linking service and to resolve reference metadata for outbound and forward (“cited by”) linking of references.

The Loader is a Java and XSLT application: first, a bibliographic and a reference instance are created from an article XML source using two XSLT stylesheets, the resulting instances are validated against their respective DTDs, and then bibliographic and reference metadata are extracted from the instances and loaded to MDDB. The application can operate in a batch or individual file selection mode.

We have chosen to implement MDDB as a relational database (Oracle on UNIX). Relational database technology in general and its query language, SQL, in particular have been around for a long time; they are reliable and well-established. There are, however, drawbacks: relational databases are very expensive and they cannot always adequately express XML structures, especially nested-ness. In AGU's case, a relational database is used only to store bibliographic and reference metadata; full text is placed into a MetaStar repository, which a user can search via the Web by means of a Fulcrum search engine.

As an alternative, one could consider using native XML databases. In principle, such an approach promises big advantages: one does not need a bridge, such as a metadata extractor/loader, between the XML text and the database; nor does one need to have two different databases, one for metadata and the other for full text. Furthermore, there is no problem of mapping XML structures into relational tables. One has to take into consideration, however, that native XML technology is just up and coming. Each vendor's definition of what exactly is meant by a “native XML database” is different, implementations are diverse, and XQuery—which is supposed to be a query language for the native XML databases—has not yet achieved the status of a W3C Recommendation. While the native XML database technology may hold great promise, we at AGU feel that it is not yet ready for implementation on a production scale. However, it may be worthwhile for a publisher to conduct a pilot project to investigate the feasibility of the native XML database solution and to keep an eye on the emerging market of the native XML databases.

6.6. The role of MDDB

6.6.1. Generating information products and services

As mentioned above, MDDB is the core of AGU information architecture: it serves as a source of providing most journal-related information products and services. The automated processes that generate AGU information products and services involve the following steps:

- depending on the desired output, a Java program extracts bibliographic information from MDDB for a specified collection of articles and constructs an XML metadata master file
- the XML master file is validated against the master file DTD[4]
- the master file is transformed into a number of HTML and XML outputs via appropriate XSLT stylesheets.

[4]There are four DTDs involved in the AGU production process: AGU article DTD; bibliographic and reference DTDs, which are part of the Loader and are used to extract bibliographic and reference metadata from the article instance and insert them into MDDB; and a master file DTD, according to which a master file is built by extracting bibliographic and reference metadata out of MDDB for various information products and services.

For example, in order to generate a year-to-date author or subject index, the Java program extracts bibliographic metadata from MDDB since the beginning of calendar year for a specific journal and creates an XML master file, which is then validated against its DTD. Then an XSLT stylesheet transforms the validated XML instance into the required HTML presentation format for posting on the Web.

Analogous processes are employed to produce various online tables of contents that are updated daily, e.g., by journal, subset[5] , category, theme, or a special section. The previously laborious and time-consuming processes of generating tables of contents, as well as author and subject indices, for printed issues have been made easy and fast by using the described process.

In addition to traditional products, storing metadata for all its journals in MDDB has enabled AGU to offer a number of cross-journal article collections, sometimes referred to as “virtual journals”: “Personal Choice” contains a selection of new articles on different topics based on index terms from across AGU journals. “Editor's Choice” contains selected new articles from across AGU journals on a specific topic.

MDDB also serves as a source of bibliographic metadata for various A&I services, such as NASA's *Astrophysics Data System (ADS)*, CrossRef, and AIP's *SPIN*. The same type of an XML metadata master file is used to deliver bibliographic metadata to an A&I service: the only difference is that in this case, the master file is transformed into a desired XML output instead of an HTML one.

6.6.2. Citation linking

AGU participates in the CrossRef citation linking service, and MDDB plays a central role in enabling various kinds of linking.

6.6.2.1. Inbound linking and response pages

“Inbound linking” means that when your publication is cited in another publication, the citation can be linked to your publication. In order to implement inbound linking, publishers create a so-called “response page” for their publication, assign a DOI to article content, and submit the article's bibliographic metadata, DOI, and response page's URL to CrossRef. A typical response page for a scholarly article usually includes its bibliographic information and abstract, and is linked to the article full-text representations, such as PDF and HTML. When your publication is cited, the citing publisher submits reference citation metadata to CrossRef, CrossRef matches the metadata to the DOI of your article, and the DOI maps to the URL of your response page.

It is very important that you be able to change the “look and feel” of your response page at any time. The only reasonable way to have this capability is to be able to autogenerate a response page according to requisite specifications. This is precisely what we have implemented at AGU using the process as described in a previous section: in order to create a response page, a Java program extracts from MDDB bibliographic metadata for a specific article, constructs an XML instance, and then an XSLT stylesheet transforms the XML instance into a required HTML representation of a response page. As a result, when readers click on a citation to an AGU article, they are taken to the MDDB-generated response page for the cited AGU article. If an article's abstract or title contain math or special characters coded as a LaTeX expression, the XSLT transformation uses an additional XML file where the mapping between the LaTeX expression and the corresponding image is defined. In the course of the XSLT transformation, an image link is substituted into the HTML representation instead of a LaTeX expression.

6.6.2.2. Outbound linking and metadata deposits

“Outbound linking” means that the references cited in an article could be linked to their sources, or—more precisely—to the response pages prepared by their publishers. In order to do that, references must be accurately marked up with a necessary degree of granularity and then a query submitted to CrossRef with the reference metadata. CrossRef will match the reference metadata against its own metadata database and return the DOIs of cited materials. These DOIs

[5]A subset is a part of a journal that encompasses a certain specialty area. It is a byproduct of the main journal. An AGU member can subscribe to a subset separately in print or electronic format.

can then resolve to the response pages' URLs. As mentioned earlier, AGU MDDB contains both bibliographic and reference metadata for its articles. It also has a field for storing DOIs of referenced materials. Having both bibliographic and reference information in MDDB enables AGU to deposit both sets of metadata into CrossRef MDDB, thereby taking full advantage of the latest CrossRef deposit schema.

Deposits of bibliographic and reference metadata to CrossRef are performed daily, and once more, the XML-centric processes are employed both to send the AGU metadata in and manage the resolved DOIs obtained from CrossRef. In order to deposit AGU metadata to CrossRef, a Java program constructs an XML metadata master file for the articles published today, which is then transformed into an XML instance conformant to the CrossRef deposit schema. The only difference in this case is that the master file here contains not only bibliographic but also reference metadata. CrossRef responds by sending an XML file with DOIs for the resolved references. An XSLT stylesheet transforms that XML file into a simplified XML instance, and then a Java program extracts the DOIs and inserts them into the reference table of MDDB.

AGU MDDB can also be used to send a query to CrossRef for any previously unresolved references. This is an important capability because publishers deposit metadata for their legacy materials all the time, and a reference unresolved today may become resolved tomorrow.

In AGU HTML, the page of references is implemented as a PHP application: the code for each reference citation has a PHP function with two parameters: article ID and reference citation ID. That PHP function simply checks whether AGU MDDB contains a DOI for the cited reference. If a DOI exists, then the PHP function constructs a link to DOI resolver, and the reader ends up at the cited publication's response page. If a DOI does not exist, then a link is not constructed. When depositing reference metadata with CrossRef, you can request a “query match alert” service: that is, CrossRef will send you a DOI of any unresolved reference when its publisher deposits it with CrossRef. At AGU there is a system that receives such notifications from CrossRef and inserts the references' DOIs into MDDB. As a result, a References page is created only once and does not have to be modified, but the number of linked sources it contains may increase every time the page is accessed.

6.6.2.3. Forward linking

“Forward linking” (citation indexing) means that you can create links to the publications that have cited your article since it was published. In order to do that, you have to submit a forward linking query to CrossRef, and it will send you DOIs and metadata of the citing publications. With a trivial addition of a few tables to store the citing publications metadata, AGU MDDB will be able to implement a “cited by” link in the HTML representation of its articles. The same XML-centric processes described in the previous section will be used both to send the metadata in and to populate MDDB with citing articles' metadata obtained from CrossRef.

6.7. Article representation: HTML vs. PDF only

While some publishers opt to provide PDF as the only representational format, it should be obvious from the foregoing why AGU's solution includes an HTML article representation as well. One reason is that implementing dynamic outbound linking requires inclusion of PHP code, and that would not be possible in PDF. Another reason is inclusion of dynamic content: PDF format cannot incorporate such article components as video, audio, or animation.

It is hoped that in the future, Web browsers will use XSL to adequately render schema-compliant XML instances that include embedded fragments marked up using standardized XML vocabularies, such as MathML, SVG, VRML, etc. Until that happens, it is useful to provide an HTML article representation.

6.8. Java vs. other programming languages

We have chosen Java as a programming language to implement both the Validator and the Loader because of its cross-platform interoperability. Since AGU chose to outsource article XML conversion and because it is a requirement that the vendor use the AGU Validator, it made sense to implement the Validator in a platform-independent language, so as not to restrict the choice of suppliers based on the platform they use.

The same issue of cross-platform interoperability was the main factor in writing the Loader in Java. AGU MDDB, initially implemented as an SQL Server on MS Windows, was later migrated to Oracle on UNIX, but we did not have to modify the Loader. As mentioned, the Loader has a batch version and an individual file selection mode. The batch implementation has no user interface, a minimal set of options, and runs in a production mode under UNIX. A single file selection version of the Loader has a user interface, is option-rich, and runs on Windows.

6.9. Retrospective conversion

One of the biggest challenges of any electronic publishing project is conversion of retrospective materials. If it is a requirement that your information repository include legacy data—as it indeed was for AGU—then sources of data must be identified, conversion specifications developed, and quality of resulting data assured. The AGU situation was difficult: its own bibliographic repository *EASI* did not contain abstracts, and it was a requirement that the new database would. AGU decided to find an A&I service that contained abstracts of AGU articles, purchase the abstracts, develop a conversion program that merged the records from its own bibliographic repository with the acquired abstracts, and populate the new bibliographic database with the resulting records. The main problem was that the AGU repository used a manuscript tracking number as an ID, while the acquired abstracts from the American Institute of Physics *SPIN* database utilized a different ID. In order to match the records, it therefore became necessary to develop a metadata matching algorithm and reconcile inevitable discrepancies between the two record sets. Another problem was that *EASI* existed in a Refer (EndNote) format, while the acquired external records were tagged in SGML. Additionally, in the AGU repository, special characters and math were marked up using LaTeX, while the *SPIN* records utilized SGML character entities and the ISO 12083 math tag set.

Faced with the prospect of conversion from multiple data sources, we implemented an XML-based conversion process: instead of writing source-specific procedures for populating a database, we chose to create interim XML records from each source, match the records, merge two XML instances into one XML metadata file, run it through a validating parser, and then extract the metadata and populate MDDB. A metadata matching program was written in OmniMark. It created bibliographic records that were then parsed against the Bibliographic DTD and loaded to MDDB using the Loader. Numerous records with metadata discrepancies were visually checked, corrected, and processed into the database. Figure 3 illustrates the process.

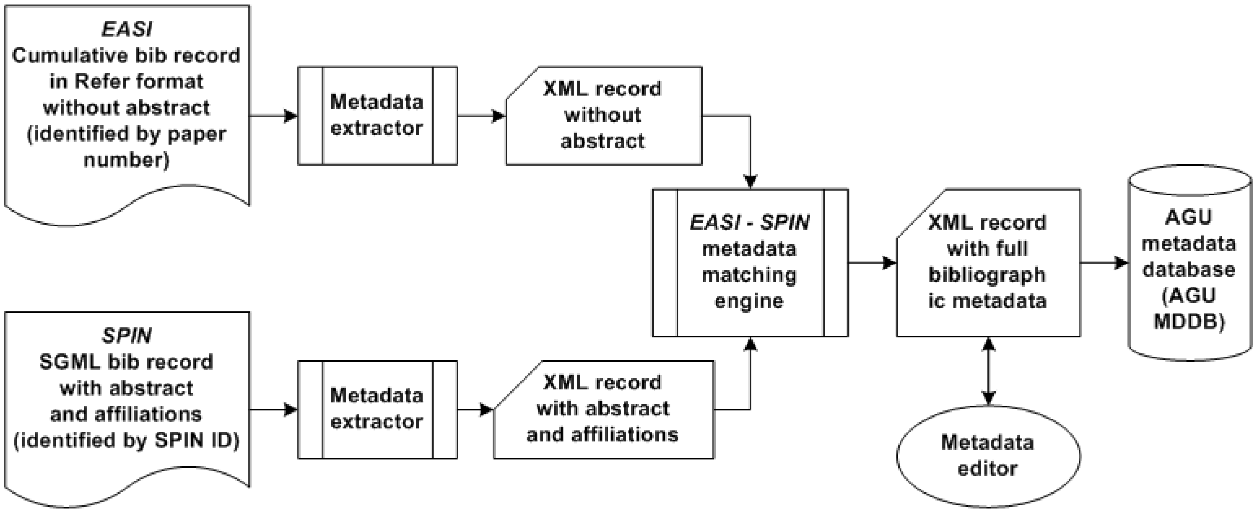
EASI
Cumulative bib record in Refer format without abstract (identified by paper number)
Metadata extractor
XML record without abstract
EASI - SPIN metadata matching engine
XML record with full bibliograph ic metadata
AGU metadata database (AGU MDDB)
SPIN
SGML bib record with abstract and affiliations (identified by SPIN ID)
Metadata extractor
XML record with abstract and affiliations
Metadata editor

Figure 3. Retrospective conversion from two bibliographic sources.

At a later stage, SGML entities and ISO 12083 math occurring in article titles and abstracts were converted into AGU DTD-compliant XML. However, we still have not converted LaTeX to XML for legacy metadata.

An alternative approach to obtaining legacy metadata would be to forego secondary sources altogether, and instead, have retrospective materials marked up in accordance with your current DTD. This can be done using OCR and pattern recognition software. Such an approach is not as radical as it may at first seem: while it will definitely incur upfront costs, it may ultimately save you time and money because you will not have to develop a conversion from multiple sources at all. Instead, you will be able to use the same tools you must develop anyway to extract and load metadata from currently published materials, if you are going to store metadata in a relational database. You will not have to deal with time-consuming quality control procedures of reconciling discrepancies among multiple metadata sources. Nor will you have to develop your own metadata matching algorithm or struggle with converting special characters from non-XML languages, such as LaTeX and SGML.

7. Lessons learned

7.1. DOI

7.1.1. What does your DOI identify?

In order to implement a system that operates with textual objects, they must be identified. In the journal publishing world, one of the most widespread identifiers is a digital object identifier (DOI). You must be very clear on precisely what your DOI identifies. The phrase “digital object identifier” should not be taken literally: what a DOI identifies does not necessarily have to be a digital object. A DOI can identify an abstraction of an intellectual work and/or its various manifestations, e.g., a PDF or HTML file; it can be even applied to a printed artifact.

You can apply different DOI suffixes based on what is known in the cataloging world as the “extent” of the record; that is, your DOI schema may account for a bibliographic record only, a bibliographic record plus an abstract, or for a full text. You may also want to assign different DOI suffixes based on the format of the item: thus XML, PDF, and HTML files of the same article may get different DOIs. Another factor to consider is the level of granularity: it makes sense to assign different DOIs to an article itself and to its components, such as figures or animations. It may be extremely important to be able to identify and register article components, because their intellectual property rights may differ from those for the whole article.

In practice, DOI usually resolves to a URL of a so-called “response page,” which typically contains bibliographic information, a free abstract, and links to full text protected by an intellectual property rights system. While this is what an article's DOI resolves to, a publisher would be well advised to have a precise understanding of what entity the DOI identifies, and build its electronic publishing system accordingly.

7.1.2. DOI format

By definition, the DOI is an “opaque” or “dumb” identifier. However, publishers like to imbue it with some intelligence. Such a decision may indeed have validity because—increasingly—a DOI, which was originally conceived as an internal identifier enabling machine processing, is being used as part of a printable and reusable citation. Since a DOI, once registered with the Handle System, cannot be changed, those who take this route must think about how they are going to deal with legacy data. For example, AGU uses an internal tracking number as a DOI suffix. However, such numbers became available as a result of using automated tracking systems, which appeared in the late 1960s. If a publisher has plans to digitize its legacy data, it will not be able to assign tracking number-based DOIs in a consistent format to the articles published before the automated tracking systems were used.

One may consider assigning DOI suffixes based on volume/year, issue, and page numbers. That approach will work if these categories are still applicable to your current publications. However, many electronic materials are published “as ready” and may not fall into such print-oriented “containers” as volume or issue. Indeed, even page numbers may not be relevant for an electronic publication. Thus you may need a different way to identify a publication: it may be a time stamp when an article becomes officially available online, an internal tracking number, or any other option. If a decision is made to use an “intelligent DOI,” then it may make sense to choose a “break point”: you use one DOI schema for the article published before it and another schema for the articles published after.

7.2. The version of record

In a multi-output journal production environment when the same intellectual work exists in numerous representations, such as XML, PDF, HTML, and print, the publisher has to decide what exactly constitutes the version of record. Many decisions will flow from that determination. If you decide to designate paper as the version of record, then you must realize that you do not really achieve the goal of separating content and presentation because paper combines both. On the other hand, there is a well-established practice of preserving paper, and if there is no requirement to preserve electronic versions in perpetuity, you avoid grappling with thorny issues of electronic file preservation.

AGU decided early on that the version of record would be XML, and that decision has had far-reaching implications.

7.2.1. Preserving scientific content vs. typographic appearance

A decision to designate XML as the version of record means that AGU is committed to preserving scientific content of the article, rather than its representation in a particular format or its appearance on a particular medium. That is, when PDF, HTML, and/or paper are no longer widespread, AGU will be able to present intellectual content of the article in whatever format is applicable at the time. Furthermore, we will not be limited to a particular device, such as a personal computer or a bound issue: AGU will be able to present the content of the published articles through a variety of devices and media, such as a personal digital assistant, mobile phone, DVD, etc. This need for scientific content to be independent of format and media makes it imperative to designate XML as a version of record.

7.2.2. Archiving

Some of the major advantages of XML are that it is a nonproprietary ASCII-based format and it is a W3C Recommendation, which affords it a status comparable to that of an international standard (SGML, of course, is an ISO standard). That makes XML ideal for preservation. The publisher, however, must decide in what formats it will preserve nontextual article components. There are no universal standards for graphics, video, animation, audio, and other nontextual files. The publisher must choose what formats will be accepted for publication and which ones will be preserved. AGU, for example, has decided to designate TIFF and EPS as archival formats for graphics, while providing GIF and JPG for view within HTML article representation.

7.3. “Published as ready”: journal model deconstructed

Separating content from presentation has had the effect of liberating scientific content from the print-bound notion of an issue; redefining the notion of a volume; and, to some extent, deconstructing the very notion of a journal.

When XML is designated as a primary format, each article can appear online as soon as it has completed its production cycle. Thus the articles are published continuously, rather than being released in discrete issues. The articles may be assigned to a specific issue and released as a printed collection later, but that print product assumes a secondary role.

Similarly, the notion of a volume, which in the world of print refers to a yearly collection of issues, has to be redefined in the electronic environment: it must be replaced either by a chronological year, or, as in AGU's case, it becomes the ordinal numeral denoting the number of years since the journal's establishment (i.e., nth year of publication).

The publisher should, however, be mindful of the “publish as ready” model's implications if during the dual paper–electronic period, XML is designated the version of record. Suppose a special section article is posted online in December and appears in a printed issue in January of the next year. What metadata would you assign to such an article, and how would you recommend citing it? If posting article representation(s) online manifests the fact of publishing, then in your DTD and MDDB you must differentiate between the year of publishing and the year of printing. Even more interesting is the question of which volume the article belongs to: if this piece of metadata is essential in your publishing model, as it is in ours, you may have to differentiate between the published volume and the printed volume numbers.

Finally, when an article is liberated from the constricting confines of issue and volume, the very notion of journal may be, if not discarded, then at least deconstructed. If your readers are interested in interdisciplinary research, why should they not be able to create their own collections of articles cutting across existing journals? Such collections can be statically or dynamically created based on user-defined criteria, such as subject descriptors, keywords, or author names, to mention just a few. “Journal” then becomes just one of many potential collections of articles, and such collections, constructed simply as a standing or an ad-hoc query executed against MDDB, become new information products or services that a publisher can offer to its reader audience.

7.4. Page numbers and article IDs

While the “published as ready” model allows the readers to see an article much sooner than they would if they had to wait until a printed issue is assembled and mailed, that model poses a challenge in terms of article identification and ordering within the issue. There are instances when it is not possible to predict the sequence of articles within an issue: in the case of special sections, even though it is an AGU requirement that all originally proposed special section papers be printed within one issue, each article appears online as soon as it is ready, and the final sequence of articles within the printed issue may not be known until the last article is accepted. Also, it does sometimes happen that an article must be removed from publication at the last moment or reassigned to a different unit within an issue, such as a subset. Obviously, such incidents impact continuous pagination within an issue.

One could argue that continuous page numbers within an issue could be assigned when the printed issue is assembled, and then reflected in the PDF representations of the article posted online. This, however, would require an alteration of the PDF, and that would run contrary to the principle that a representation of the version of record cannot be changed after the article is officially published. Even more importantly, if a journal is published infrequently, e.g., quarterly, it may take a long time between the article's publication and the time when an issue is assembled. Meanwhile, how would a published article be cited? If it is to be cited without page numbers, it is not a good idea to add them later because that would create inconsistency between how the same article is cited. Also, those who see an HTML representation of the article, have to be able to cite it, too; if the page numbers were to be added later, they would have to be added to HTML.

It seems then logical to abandon continuous page numbering altogether. If you do that, however, how would your readers locate an article within a printed issue? And, interestingly enough, the page number problem is not confined to print: as it turns out, “pages” are needed in the electronic world as well. Even if a publisher designates XML as the version of record and considers making an article available online as the fact of publication, it still must think about many bibliographic systems and services that require page numbers. For example, for any STM publisher it is vital that its publications be counted by Thomson ISI in its Science Citation Index® and Web of Science® for the purposes of individual citation tracking and calculating journal impact factor. If a publisher does not want to use page numbers, ISI can use any identifier… as long as it is no longer than four (or maximum six) alphanumeric characters! Therefore, you cannot hope to use a DOI as an identifier for ISI. CrossRef, one of the most prominent citation linking services in the STM world, accepts article IDs in its deposits but does not use them in its metadata resolution algorithm: it uses page numbers. Various A&I services that index your journal are likely to need page numbers as well.

To summarize, a publisher that decides to adopt the “published as ready” model faces the following challenges in terms of article identification:

- at the time of article publishing it is not always possible to predict accurately what its continuous pagination within the printed issue will be
- waiting until a printed issue is assembled and then adding page numbers to article representations creates discrepancy in how an article is to be cited and runs contrary to the principle that an article must not be changed after it is published
- abandoning page numbers altogether is not an option because many A&I services, most notably Thomson ISI, may need them for the purposes of citation tracking and metadata resolution

- as long as a printed issue exists, the reader needs a means of finding a particular article within it.

Faced with these challenges, AGU has devised the following approach. It did away with continuous page numbering within an issue and introduced a smart article identifier dubbed “citation number,” while providing page numbers within an individual article. “Citation number” is a concise identifier consisting of six alphanumeric characters. It contains information about a journal; issue number; a unit, i.e., subset, subject category, or special section, if any, an article belongs to; and the order in which the article is likely to appear within the issue or the unit. Page numbers within an article follow the “X of Y” format. When an article is published, it contains a recommended “how to cite” fragment, which includes authors, publication year, article title, journal, volume number, citation number, and DOI, e.g.,

> Citation: Holzworth, R. H., and R. A. Goldberg (2004), Electric field measurements in noctilucent clouds, *J. Geophys. Res.*, 109, D16203, doi:10.1029/2003JD004468.

D16203 is a citation number here, where

D	is part D (Atmospheres) of Journal of Geophysical Research (JGRD)
16	is an issue number
2	is the “Aerosols and Clouds” subset of JGRD
03	is the article sequence within the subset.

AGU's solution has allowed it to overcome the mentioned challenges:

- an article can be cited as soon as it is published, and all metadata necessary for the citation are contained in the version of record (XML), as well as in HTML and PDF representations
- A&I services can use either page numbers (each article begins with a number 1, though), or a citation number, or both
- citation numbers appear in print, which makes it easy for the reader to locate an article within a printed issue
- while citation numbers in most cases follow the physical sequence of articles within an issue or its unit, they may occasionally deviate from that sequence, thereby giving AGU the flexibility to deal with exceptions to a regular publishing flow.

Introducing citation numbers has also given the journal editors an option to create tables of contents that list issue articles in an order that is different from their physical sequence in the issue. Since the readers know that the articles are located within the issue in the order of citation numbers, the editor may decide to group the article in a customized way for the table of contents. In addition, in each issue AGU provides an author index that maps author names to their article's citation number.

There is still one potential problem in this approach, of which a publisher must be aware: if more than one article with the identical first author surname appears within the same issue, there may be a problem with article discovery for some A&I services, since each article within an issue begins with number 1 and their search algorithms usually do not take article titles into consideration. CrossRef has addressed this problem by providing for a multiple resolution, which, however, has to be explicitly requested in the query. One final caution: a publisher must never place more than one article on the same page, no matter how short the articles might be.

7.5. Special characters and math

7.5.1. XML character entities vs. Unicode

There are two approaches a publisher can take to code special characters: XML character entities or Unicode points. AGU has decided in favor of XML character entities for two reasons. First, XML is the version of record and has to be preserved. It has to be readable not only by machines but also by humans, and an XML character entity **&eacute;** is much more intelligible than a Unicode point **&#x000E9;**. Second, as long as a DTD contains mapping files from entity names to their Unicode values, an output with Unicode points can always be produced simply by running a validating parser.

7.5.2. Marking up math

There are two aspects of dealing with math: how to tag it in the source document and how to display it. We considered three approaches to marking up math in XML documents: using MathML, using LaTeX, or providing a link to an external graphic file. As for displaying math, there are two options: use browsers capable of displaying MathML or provide an image. The major advantages of MathML are that it is an XML vocabulary and it uses Unicode for special characters. However, those who decide to use MathML should bear in mind the following.

There are two flavors of MathML: Presentation and Content. Currently only MathML Presentation is developed enough to be used in a production mode. However, in our opinion, the concept of MathML Presentation is somewhat contrary to the XML philosophy of separation of structure and presentation: rather than express the semantics of mathematical expression, MathML Presentation describes how it looks (and LaTeX does a better job at that). MathML Presentation is extremely verbose; it takes dozens of lines to tag even a simple equation. Tagging complex expressions may require hundreds of lines of code, and it is rather difficult to find and correct an error, should one occur. MathML Content does strive to represent mathematical expression's semantics but it is not yet ready for prime time.

There are numerous problems with displaying MathML Presentation in Web browsers [Gaylord, 2004]. Some browsers, such as Firefox 1.0, Mozilla 1.7 and Netscape 7.1, can display MathML natively; others, such as IE 6.0, need a plug-in; and still others, such as Opera, neither display MathML natively nor allow for a plug-in. All browsers require additional fonts, and even when you install them, not all characters can be displayed: some browsers are more Unicode-compliant than others. In addition, MathML-containing file has to be an XHTML document.

In our view, given the present state of Web browsers, a publisher that serves a diverse reader community who uses a multitude of browsers on a variety of platforms would be taking a risk, should it make a decision to use MathML as a display format.

Another option is to produce an image of the math and link to it. This approach has the advantage of simplicity and complete control over the appearance of math symbols, but it has numerous drawbacks: the math cannot be reused, e.g., copied and pasted, or searched and retrieved.

AGU decided to use LaTeX for tagging math within XML and to present math as an image in HTML. In addition to the mentioned deficiencies of the MathML Presentation, a factor in AGU's decision was that about 40% of accepted manuscripts are supplied in LaTeX, and author tagging can be reused without recoding or alteration. It is trivial to produce GIF images out of LaTeX, and there is hope that when MathML Content is ready, it will be possible for AGU to conduct an unambiguous conversion from LaTeX to MathML Content.

In dealing with LaTeX, one has to be cognizant of the fact that it does not use Unicode. But how to deal with the situation when the same mathematical symbol is used inline and in a displayed formula? There are two approaches: code inline characters using an XML entity (which will resolve into a Unicode character for presentation in HTML) or use LaTeX (which will result in an inline GIF image in HTML). The safest way to ensure identicalness of the same math symbol occurring inline and in a displayed formula is to code both in LaTeX and present both as images in HTML. An additional argument in favor of such an approach is that different browsers may and do render the same Unicode characters differently. Thus your inline symbol will look different in different browsers, and discrepancy between the appearance of the same symbols inline and in a displayed formula may result in distortion of the author's intent and scientific meaning, which is not acceptable. To give an example of what makes matters even worse, **/varepsilon** in LaTeX looks disturbingly similar to Unicode's U03B5 (straight epsilon) as rendered by some browsers, while LaTeX's **/epsilon** looks like Unicode's U025B (curly epsilon). The Unicode pair U03D5/U03C6 (straight phi and curly phi) are confusing too, in part because Unicode reversed the glyphs of these two between versions 2 and 3.

On the other hand, using LaTeX to code all inline characters, even such “noncontroversial” ones as Greek alpha, that appear reasonably similar across various browsers when represented by a Unicode point, may run contrary to the text reuse and accessibility requirement, when they are to be represented by GIF images in HTML. Each publisher has to decide for itself what the optimal approach should be.

AGU's solution was to identify special characters that (1) look substantially different when rendered by various browsers (essentially, all Greek letters that have a “var-” counterpart), and (2) cannot be rendered by the IE6 browser (which the majority of AGU readers use), and mark up those in LaTeX. We created a clear special character markup policy in writing, providing strict and precise specifications to the vendor as to which inline symbols are to be tagged in LaTeX and which are to be represented by XML entities. The policy is periodically revisited to take into consideration the state of MathML and that of browsers.

A publisher who faces a decision on how to tag and display math can now consider several options. For example, some publishers have alternate representation of math in the XML, e.g., MathML and LaTeX, and/or a link to an image, and comment MathML out until a future time. If a decision is made to use MathML only, then a publisher may decide to generate an image from the MathML on the server side for the HTML, rather than to rely on the browsers. Of course, if you do not provide an HTML rendition of your materials, you only have to decide how to code math in the source document and you do not have to grapple with the issues of math display at all.

8. Results

In a few short years the American Geophysical Union leapfrogged from the paste-up era, when authors prepared their own “camera-ready copy” to be pasted on boards for a printer, to fully electronic XML-centric production of journal articles and associated information products. This transition has required a major software development effort, reengineering of the workflow, and—most importantly—a cultural shift in the organization: the staff's mindset had to adjust to the new world where publishing was not the same as printing, content could be separate from presentation, and a multitude of information products with a different “look and feel” could be generated from the same source. Even though the transition has not been without growing pains, it has allowed AGU to reap a number of strategically important benefits.

8.1. Improved productivity

- since introduction of the XML-centric workflow, the number of published articles has increased, while in the journal production department two full-time positions have been eliminated and 25% of staff positions have been downgraded
- production time from acceptance to publication has been substantially reduced (it is now the fastest since 1984—when records began to be kept), and the trend continues
- management reporting has improved significantly.

8.2. Automated production of publishing products

Producing issue tables of contents, as well as end-of-year author and subject indices, used to be a labor-intensive, time-consuming, and error-prone process. Having XML publishing metadata verified and validated prior to populating MDDB has enabled AGU to generate all metadata-based products automatically. A software application creates these products on a predefined schedule (usually in the nighttime to decrease the network load), and after a visual quality control they are posted on the Web.

It is instructive to include herein the recollection of an AGU managing editor involved in the process of preparing an issue table of contents before an XML-centric workflow was introduced:

> When the guts of an issue was prepared, copies of the first pages of the articles were compiled. The production coordinator marked up the first pages for capitalization and subject category, for example, “Corrections” or “Commentaries,” and the group was released to the printer along with the mechanical boards. The printer then set a Table of Contents according to AGU specifications, and in about 4-5 days we received back galley proofs, which went through the proofing cycle. We would then return the marked up galleys for the printer to correct. Three days later, we would receive page proofs for checking. The proofreader compared the page proofs against copies of the galleys to make sure that all corrections had been made. If everything was okay, the production coordinator, would approve the Table of Contents for print. In most cases, however, small changes needed to be made, which necessitated another few days and another round of proofing. In all, issue Table of Contents production could take as much as a week and the cost for such handling was dear.
>
> With the advent of the current electronic workflow, the time-consuming, tedious task of producing the Table of Contents has been substantially reduced to one, or at absolute most, two days. Now the list of articles in the issue is output from MDDB, and then is formatted by the division secretary, who passes her output to the proofreader. The proofreader marks up changes and returns it to the secretary for correction. The Table of Contents is then released to the print team. The print team makes minor adjustments and produces a PDF, which is released to the printer at the same time as the laser proofs and the electronic files for the issue are released. This new process has cut at least 5 days from the production cycle and greatly reduced the costs of preparation.

8.3. Improved quality of published product

Prior to introduction of the XML-centric workflow the quality of the published articles was not consistent: even though the manuscript went through a copyediting step, the final production copy was prepared by the author, who may or may not have incorporated all of the copy editor's suggestions. Besides, there was always a possibility of a human error on the part of even the most conscientious copy editor. An introduction of XML quality control procedures has improved accuracy of the end-product by reducing human error: the authors are now responsible only for producing the content of their articles, while the Validator checks the accuracy and consistency of the article's structure by automatically applying more than 100 validation rules to each article. After this rigorous checking, various outputs are produced automatically from a single source of information. Such a workflow ensures both content consistency among various outputs and presentation consistency among article representation within the same format, such as PDF/print or HTML.

Introducing reference resolution has made it possible to do something that previously was simply not feasible: namely, to check the accuracy of the cited references' metadata. This has increased the quality and value of the published product, and made it easier for the reader to find the cited sources of information.

8.4. Automatic production of the Web search repository

When AGU readers search for an article through the Web-based form, they in fact perform a search against a full-text repository. Articles' metadata and full text come to the repository automatically, by way of daily extractions from XML files. While this procedure seems natural and almost routine now, the process of creating an AGU bibliographic repository *EASI* was much different just a few years ago. We can now look back with a mixture of amusement and amazement at the 2001 documentation of how *EASI* was produced as recently as in 2001:

> When a journal issue is finalized, a production coordinator photocopies the first page of each article, writes in some additional data, such as the AGU index terms (which do not appear in print), and sends this piece of paper to the Publications Administration department. The staff there manually

input the data into the fields set up in a WordPerfect template. The way the template is set up is inconvenient, in that it requires keying-in placeholders for the empty fields and rekeying the same information if the number of authors exceeds a preset limit. If any field contains a diacritical mark, a Greek letter, or mathematical or chemical formulas, the staff codes them in LaTeX. When the records for all the articles in the journal issue are input, the record set is printed out, and the printout is sent to the Editorial Services department for proofreading against the photocopies of the first pages, which are also sent there. After receiving the corrected printout a Publications Administration staff member runs a WordPerfect macro on the source file to merge the individual records. A different macro has to be used for each journal. Then a clean-up macro has to be run on the merged file to eliminate the placeholders. After that, yet another macro has to be applied (twice!) to each individual index term code to add a first-tier and second-tier index term description to the code. Then the file is sent to the South-Western Corporation that runs an error check, and returns an error report to the Publications Administration. The staff there does necessary research and supplies missing or incorrect information. Then the South-Western Corporation converts the file to the delimited Refer format required by EndNote software and returns it to the Publications Administration, which passes this file to the Secondary Services department. The Secondary Services department actually produces an online *EASI* database: the received file is appended to the rest of the source file, and the updated file is reindexed in its entirety by the WAIS software.

8.5. Automatic daily archiving

AGU maintains an archive populated with the XML source, PDF representation, and metadata for the daily published articles. The archive also contains nontextual article components, such as graphics, videos, and animations. The daily archiving process is fully automated.

8.6. Direct data feeds to A&I services

As mentioned earlier, MDDB and the XML-centric workflow have made it possible to implement direct feed of bibliographic metadata into various A&I services, such as NASA's *Astrophysics Data System (ADS)*, AIP's *SPIN*, and CrossRef. This is a long way from the not-so-distant past when journal issues had to be printed, mailed, and metadata manually keyed in. The delay between article publication and metadata appearance in A&I services was sometimes as long as half a year. Now the process is fully automated, electronic, and instantaneous.

8.7. Reference linking implementation

Introduction of MDDB and XML-centric workflow has enabled AGU to take full advantage of CrossRef citation linking services. Because both AGU and CrossRef use XML as a data format, all AGU needs to do is to generate an XML file from MDDB and transform it into CrossRef's XML format by using XSLT. After sending the data feeds and receiving the XML-formatted response email from CrossRef, AGU generates a simplified XML file from it using XSLT, and updates MDDB with the extracted DOIs for the resolved references. AGU has already implemented inbound and outbound linking, and will soon launch forward linking. As part of inbound linking, AGU has implemented fully automatic generation of response pages, so that their format and content could be easily and uniformly changed at any moment.

8.8. Introduction of new information products and services

Many products and services that would not have been possible in a printed world, such as cross-journal collections, inclusion of multimedia content, and immediate access to datasets that are underlying scientific research, have already been offered to AGU constituents, while others, such as RSS data distribution, can easily be implemented. One of the most exciting offerings is a “virtual journal,” which is a collection of articles sharing a set of index terms. MDDB provides an easy way to retrieve articles' metadata using their index terms and publication date. Using an XSLT stylesheet, a “virtual journal” can be updated on a daily basis.

Looking back, we can now definitely conclude that reengineering the AGU production process so that it would become XML-centric was the right move that has allowed AGU to bring its constituents the results of scientific research of the highest quality in the quickest and most cost-efficient manner.

Acknowledgements

The authors would like to express their gratitude to AGU's Jane Seebold and Barbara Major for valuable contributions to this paper. We are also greatly indebted to Inera™ Inc.'s Bruce Rosenblum and Mulberry Technologies Inc.'s Tommie Usdin for kindly agreeing to review this paper and for providing their insightful comments. We would like to thank AGU's Nathaniel Pendleton and SPI Publisher Services's Villanar Villanueva for stimulating discussions. Finally, we would like to acknowledge AGU management for providing the time and resources for writing this paper.

Bibliography

[Beck and Lapeyre, 2003] Beck, Jeff and Deborah Lapeyre (2003), *New Public Domain Journal Article Archiving and Interchange DTDs*, XML Conference & Exposition 2003, Philadelphia, Pennsylvania, USA, 7–12 December 2003, XML 2003 Conference Proceedings, http://www.idealliance.org/papers/dx_xml03/papers/04-01-02/04-01-02.html

[Gaylord, 2004] Gaylord, Tonya R. (2004), *Getting MathML to display isn't a cakewalk yet…*, poster presented at the 2004 Extreme Markup Languages Conference, Montreal, Quebec, Canada, 2–6 August 2004.

[Rosenblum and Golfman, 2001] Rosenblum, Bruce and Irina Golfman (2001), *E-Journal Archive DTD Feasibility Study* prepared for the Harvard University Library Office for Information Systems E-Journal Archiving Project by Inera™ Inc., 65 pp., http://www.diglib.org/preserve/hadtdfs.pdf

Biography

Alexander B. **Schwarzman**
Information Systems Analyst
American Geophysical Union [http://www.agu.org]
Washington
District of Columbia
United States of America
sschwarzman@agu.org

Alexander (‘Sasha’) Schwarzman is information systems analyst at the American Geophysical Union (AGU). Since 1999, he has been involved in AGU's transition from a paper-based to an XML-oriented publishing model. He developed and maintained AGU Article SGML and XML DTDs, formulated requirements for AGU metadata database and suite of XML/XSLT tools that work with it, designed specifications for conversion of legacy metadata into the database, and implemented an XML-centric production workflow for currently published journals that includes AGU staff and outside vendors. Before reengineering AGU's journal publishing process, Sasha took part in developing an innovative Internet-based system for submission, management, and publishing of abstracts for meetings and conferences organized by AGU. Prior to joining AGU, Sasha worked at the National Agricultural Library's Text Digitizing Program. He holds a Master of Science degree in Mechanical Engineering from the State Technical University of St.Petersburg, Russia, and a Master of Library Science degree from the University of Maryland, College Park, USA.

Hyunmin **Hur**
Founder and Principal Software Engineer
DocsDoc, Inc.
Seoul
Republic of Korea
hhur@docsdoc.com

Hyunmin Hur is the founder and a principal software engineer of DocsDoc, Inc., based in South Korea. He has been involved with XML for more than five years and has developed professional XML-based information systems in various business areas.

Shu-Li **Pai**
Electronic Publications Systems Specialist
American Geophysical Union [http://www.agu.org]
Washington
District of Columbia
United States of America
spai@agu.org

Shu-Li Pai is the electronic publications systems specialist at the American Geophysical Union. His primary interest is to facilitate the AGU's daily operation through XML. He has been involved in developing software systems, including credit authorization, desktop publishing, fax store-and-forwarding and soft switch. He holds a Master of Science degree in Computer Science from the Indiana University at Bloomington.

Carter M. **Glass**
Manager, Electronic Publications Systems Development
American Geophysical Union [http://www.agu.org]
Washington
District of Columbia
United States of America
cglass@agu.org

Carter Glass is the manager of electronic publishing development at the American Geophysical Union. His mission is to build tools to help scientists collaborate in research and publish their results. Before working at AGU, Carter was the manager of the Internet Publishing and E-commerce unit of The Thomson Labs. He conducted research into all aspects of large–scale electronic publishing and publishing commerce. His areas of research interest were XML, SGML, online payment systems, dynamic database publishing, internet advertising and content protection technologies. He has served as internet strategy consultant to senior publishing executives and hosted numerous workshops on internet publishing. Carter has been building database systems for over fifteen years and was drawn to online publishing because it has the most interesting challenges in the years ahead.