Qualified Dublin Core using RDF for Sci-Tech Journal Articles

Thomas G. Habing, Timothy W. Cole, and William H. Mischo
Grainger Engineering Library Information Center
University of Illinois at Urbana-Champaign, USA
thabing@uiuc.edu, t-cole3@uiuc.edu, w-mischo@uiuc.edu
http://dli.grainger.uiuc.edu/idli/idli.htm

Abstract

As a participant in the D-Lib Test Suite project, the University of Illinois maintains a full-text XML testbed containing over 65,000 scientific and technical journal articles. For this project, we have developed techniques to generate and utilize Qualified Dublin Core (DCQ) metadata using RDF. In this short paper we will provide a brief overview of our digital collection, demonstrate DCQ metadata creation and usage for items in our collection, and finally summarize some of the issues that arose as a consequence of our decision to use DCQ.

Keywords: *Dublin Core, Metadata, Electronic Publishing, Digital Library, XSLT, XML, RDF.*

1 History and description of the testbed

In 1994 a digital library testbed containing SGML-formatted journal articles was created as part of a Digital Library Initiative I project at the University of Illinois at Urbana-Champaign (UIUC) [1]. In 1998 the testbed was converted to XML and continued as part of the ongoing Corporation for National Research Initiative's (CNRI) D-Lib Test Suite program. During this time various professional society publishers have contributed testbed content. Current participants include: the Association for Computing Machinery (ACM), the American Institute of Physics (AIP), the American Physical Society (APS), the American Society of Civil Engineers (ASCE), Elsevier Science, and the Institution of Electrical Engineers (IEE). ASM International also has contributed several volumes of their ASM Handbook. The Testbed currently contains some 65,000 articles from approximately 50 journals. A production Web-based search and retrieval interface, DeLIver (Desktop Link to Virtual Engineering Resources) has been implemented as part of the testbed [2]. Contributing publishers and other collaborating partners, including the Naval Research Laboratory and NTT Learning Systems, supplement the CNRI grant with additional monetary and in-kind support.

2 Usage of Metadata in Illinois Testbed

Since the project's inception, metadata issues have been an active concern. Metadata records are used to fulfill several important functions.

2.1 Normalization

Metadata provide normalized information across key fields extracted from otherwise heterogeneous DTDs. This is required not only to facilitate search and discover, but also to provide common and easily displayable search results. For example, the markup used for bibliographic data about an article (including referenced articles) varies widely from publisher to publisher. However, our metadata generation processes extract these various metadata into a common scheme based on DCQ and RDF. In addition to the typical bibliographic metadata, we also normalize and maintain metadata such as figure and table captions, author affiliations, subject keywords, abstracts, and tables of contents.

2.2 Value-added

Metadata also facilitate inclusion of value-added information beyond what is available in the marked-up full-text articles. For example, links to referenced and referencing works can be maintained. These can be links to the locally maintained full-text, to surrogate records in various Abstracting and Indexing databases, such as INSPEC or Compendex, or to Digital Object Identifier (DOI) links obtained from the CrossRef publisher initiative [3] during the metadata generation process.

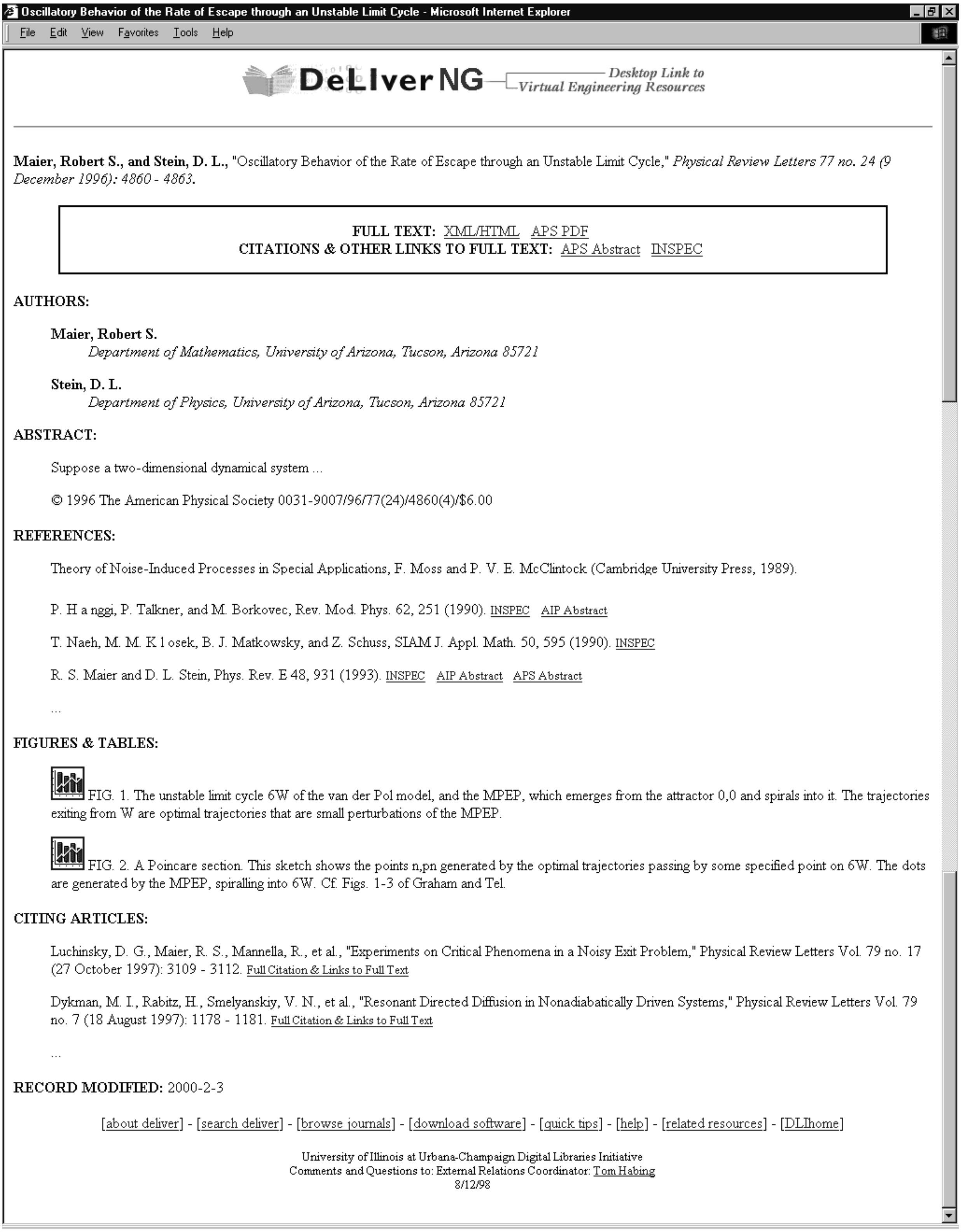
2.3 Presentation

Metadata are used for display of intermediate search results, extended citations, and for tables of contents. Metadata also are dynamically merged with the article full-text at render time. This allows access from the full-text article view to the various value-added features. It also allows us to provide these features to the user while simultaneously

https://doi.org/10.23106/dcmi.952106655

maintaining the archival integrity of the original full-text files as provided by the publishers. (Metadata files are stored separate from article full-text.)

Figure 1, showing an extended citation for one of the Testbed articles, illustrates metadata maintained.


Oscillatory Behavior of the Rate of Escape through an Unstable Limit Cycle - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
DeLIver NG
Desktop Link to Virtual Engineering Resources

Maier, Robert S., and Stein, D. L., "Oscillatory Behavior of the Rate of Escape through an Unstable Limit Cycle," *Physical Review Letters 77 no. 24 (9 December 1996): 4860 - 4863.*

FULL TEXT: XML/HTML APS PDF
CITATIONS & OTHER LINKS TO FULL TEXT: APS Abstract INSPEC

AUTHORS:

Maier, Robert S.
Department of Mathematics, University of Arizona, Tucson, Arizona 85721

Stein, D. L.
Department of Physics, University of Arizona, Tucson, Arizona 85721

ABSTRACT:

Suppose a two-dimensional dynamical system ...

© 1996 The American Physical Society 0031-9007/96/77(24)/4860(4)/$6.00

REFERENCES:

Theory of Noise-Induced Processes in Special Applications, F. Moss and P. V. E. McClintock (Cambridge University Press, 1989).

P. H a nggi, P. Talkner, and M. Borkovec, Rev. Mod. Phys. 62, 251 (1990). INSPEC AIP Abstract

T. Naeh, M. M. K l osek, B. J. Matkowsky, and Z. Schuss, SIAM J. Appl. Math. 50, 595 (1990). INSPEC

R. S. Maier and D. L. Stein, Phys. Rev. E 48, 931 (1993). INSPEC AIP Abstract APS Abstract

...

FIGURES & TABLES:

FIG. 1. The unstable limit cycle 6W of the van der Pol model, and the MPEP, which emerges from the attractor 0,0 and spirals into it. The trajectories exiting from W are optimal trajectories that are small perturbations of the MPEP.

FIG. 2. A Poincare section. This sketch shows the points n,pn generated by the optimal trajectories passing by some specified point on 6W. The dots are generated by the MPEP, spiralling into 6W. Cf. Figs. 1-3 of Graham and Tel.

CITING ARTICLES:

Luchinsky, D. G., Maier, R. S., Mannella, R., et al., "Experiments on Critical Phenomena in a Noisy Exit Problem," Physical Review Letters Vol. 79 no. 17 (27 October 1997): 3109 - 3112. Full Citation & Links to Full Text

Dykman, M. I., Rabitz, H., Smelyanskiy, V. N., et al., "Resonant Directed Diffusion in Nonadiabatically Driven Systems," Physical Review Letters Vol. 79 no. 7 (18 August 1997): 1178 - 1181. Full Citation & Links to Full Text

...

RECORD MODIFIED: 2000-2-3

[about deliver] - [search deliver] - [browse journals] - [download software] - [quick tips] - [help] - [related resources] - [DLIhome]

University of Illinois at Urbana-Champaign Digital Libraries Initiative
Comments and Questions to: External Relations Coordinator: Tom Habing
8/12/98

Figure 1. Sample Extended Citation

https://doi.org/10.23106/dcmi.952106655

3 Metadata extraction process

The XML metadata files are created by transforming the source full-text XML using Extensible Stylesheet Language Transformations (XSLT). These transformations are managed by a driver program written in Microsoft Visual Basic which also manages other aspects of the overall document processing flow.

Depending on the publisher's markup, in some cases there are required bibliographic metadata elements that cannot be derived explicitly from the source full-text. For example, some publishers omit volume, issue, and page numbers, or issue dates from their full-text markup. In these cases, the required values are derived by the VB program, possibly from operator inputs or by parsing file names, and they are then passed to the XSLT as parameters. Except for these parameters, all other processing is done entirely by the XSLT.

3.1 Metadata mappings

In many cases, the transformation of full-text XML elements into DCQ RDF properties is straightforward. For example, the following source XML:

```
<titlegrp>
  <title>This is the Long Title</title>
  <alttitle>Short Title</alttitle>
</titlegrp>
```

is easily transformed into:

```
<dc:title>This is the Long Title</dc:title>
<dcq:alternative>Short Title</dcq:alternative>
```

by a simple one-to-one mapping.

However, the derivation of other properties may require complex processing. For example,

```
<dcq:tableOfContents>
  1 Introduction;
  2 Outline of the algorithm;
    2.1 The discretization error estimate ;
  3 Concluding remarks;
</dcq:tableOfContents>
```

requires extracting section and sub-section headings from throughout an article.

Another example requiring more complex processing is mathematics. Because mathematical markup needs to be preserved in the metadata, many of the RDF property elements require the rdf:parseType='Literal' attribute. For example:

```
<dc:title rdf:parseType='Literal'>
  <math xmlns='&MathML;'>
    <msup><mi>x</mi><mn>2</mn></msup>
  </math> is x squared
</dc:title>
```

Deriving this is complicated because there may be additional markup that needs to be discarded, while the math markup is preserved. This means that a simple <xsl:copy-of> cannot be used. Instead, for each node in the source element an XSLT template is recursively invoked. The template has code to determine whether a sub-element must be preserved or whether it can be discarded, only saving its textual content.

3.2 Advanced XSLT techniques

There are also a few cases when XSLT extension mechanisms must be invoked to perform processing which is not possible using only native XSLT. One example is date conversion. Whenever possible, dates are converted into the W3CDTF encoding. However, XSLT has no built-in date conversion functions. Instead an XSLT Extension Function, written in JavaScript, is invoked.

In addition, much of the value-added metadata, such as links to cited or citing articles, is obtained from sources outside the article full-text XML. This is accomplished by using the XSLT document() function. This function takes the URL of an XML document and returns a node-set representing the complete XML tree for that document. Nodes or fragments from this tree can then be used in the XSLT. The URLs are constructed by concatenating various fixed strings and metadata elements together from the source XML document. For example, the following XSLT snippets:

```
<xsl:variable name="XREF_URL" select="
concat($XREF_BASE_URL, '&amp;qissn=', $ISSN,
'&amp;qyear=', substring-before($IssueDate, '-'),
'&amp;qvolume=', $Volume, '&amp;qpage=',
/artcl/jart/jafm/pubfront/fpage )"/>
…
<xsl:variable name="XREF"
select="document($XREF_URL)"/>
…
<xsl:element name="dc:identifier"  >
   <xsl:element name="uiLib:DOI">
      <xsl:element name="rdf:value">
       <xsl:value-of select="$XREF/kernel/doi/string"/>
      </xsl:element>
   </xsl:element>
</xsl:element>
```

results in the following RDF:

```
<dc:identifier>
   <uiLib:DOI>
      <rdf:value>10.1234/1.23456789</rdf:value>
   </uiLib:DOI>
</dc:identifier>
```

In this way, the Digital Object Identifier (DOI) for an article can be obtained from the CrossRef system via an OpenURL [4]. Similar techniques are used to retrieve value-added metadata about citations, such as links and DOIs.

XSLT is used to create the RDF metadata file. Once the RDF is created, XSLT is also used to:

https://doi.org/10.23106/dcmi.952106655

- Generate RDF triples that are inserted into a simple relational database for indexing and searching.
- 'Dumb-down' to unqualified Dublin Core for use with protocols such as Open Archives (OAI) Protocol for Metadata Harvesting.
- Transform RDF metadata into HTML for display in a Web browser.

4 Summary of issues

Until early 2001 there was minimal guidance on encoding DCQ using RDF. Only trivial or very simple examples were available. It was the availability in early 2001 of a redraft of *Expressing Qualified Dublin Core in RDF* [5], in combination with existing RDF documentation, notably the *RDF Model and Syntax Specification* [6], that finally allowed us to utilize DCQ for our project.

Early on, however, we discovered that the DCMI-approved DCQ refinements and encodings were not sufficient for all of our needs. While work on additional refinements and encodings is underway, the existing guidance from some of the DCMI Working Groups, such as the Citation Working Group and the Agents Working Group, is sparse and not in final form. Few examples are available. Pending further progress by these Working Groups, we needed extensions in three areas. To do this we created our own namespace (uiLib in the following examples) and an RDF schema for that namespace.

4.1 Citation-related extensions

Local extensions were required to represent bibliographic citation metadata, such as journal title, ISSN, CODEN, volume, issue, page, etc. These metadata seemed to fall under the purview of the Citation Working Group. The existing guidance provided by the Working Group was useful, but didn't contain enough detail. In order to allow the desired types of citation linking, we had strict requirements for fine-grained control of and access to these metadata elements. Based on the various working drafts and also on proposals coming from the mailing lists, we used three overlapping methods to express this metadata, both for the articles themselves and also for the citations.

The first method was to create a local sub-property or refinement of Identifier, "uiLib:citation." This refinement is designed to contain a human-readable citation. For example:

```
<uiLib:citation>A. Author. "A Title" Some Jrnl. 25.3
(Sep. 1999): 279-305.</uiLib:citation>
```

This content is used primarily for display purposes.

The second method was to create a local encoding for Identifier, following the OpenURL scheme [7]. For example:

```
<dc:identifier>
  <uiLib:OpenURL-OBJECT-METADATA-ZONE>
    <rdf:value>
      genre=article&amp;aulast=Author&amp;
      issn=0098-3500&amp;volume=25&amp;
      issue=3&amp;spage=279&amp;date=1999-09
    </rdf:value>
  </uiLib:OpenURL-OBJECT-METADATA-ZONE>
</dc:identifier>
```

The third method utilized the Relation refinement "dcq:isPartOf." In this approach an rdf:Description which contains journal metadata is embedded within the dcq:isPartOf node. For example:

```
<dcq:isPartOf>
  <rdf:Description rdf:ID="JournalIssue">
    <dc:identifier>
      <uiLib:ISSN>
        <rdf:value>1234-5678</rdf:value>
      </uiLib:ISSN>
    </dc:identifier>
    <dc:title>Some Journal</dc:title>
    <dcq:alternative>Some Jrnl.</dcq:alternative>
  </rdf:Description>
</dcq:isPartOf>
```

Utilizing one or more of the above methods, and using a minimal amount of string parsing, every element required for citation linking can be extracted from the metadata.

4.2 Agent related extensions

Local extensions also were required to represent additional Creator and Contributor properties, such as affiliation and email address. These are issues within the purview of the Agents Working Group. The initial proposals from this group were more concrete than those from the Citation Working Group, but still predate last July's DCQ release and are far from final. Currently there are also competing proposals, e.g., ones based on vCard and LCNAF. Our implementation makes use of the semantics originally proposed by the Agents Working Group [8], pending an update from them.

In order to maintain the significance of the sequence of authors, an RDF Sequence container was used. This is illustrated below, along with the DC Agent Working Group semantics that we utilize:

```
<dc:creator>
 <rdf:Seq>
  <rdf:li>
   <dca:Person rdf:ID="AUTHOR-1">
    <dca:agentname>
     <dca:FNF>
      <rdf:value>Author, A. N.</rdf:value>
     </dca:FNF>
    </dca:agentname>
    <dca:agentaffiliation>
     Big University
    </dca:agentaffiliation>
    <dca:agentidentifier
     rdf:resource="mailto:ana@big.edu"/>
```

https://doi.org/10.23106/dcmi.952106655

```
      </dca:Person>
    </rdf:li>
    <rdf:li>
      <dca:Person rdf:ID="AUTHOR-2">
        …
      </dca:Person>
    </rdf:li>
  </rdf:Seq>
</dc:creator>
```

The namespace prefix dca is used for DC Agent tags. At the time of this writing, there was no official namespace URI or RDF Schema for DC Agent, so the URL to the standards document was used as the namespace URI.

4.3 Type and encoding extensions

Finally, local extensions were needed to allow for additional types and encodings beyond those defined by the DCMI Type Vocabulary and by DCQ itself. For example, types were needed to differentiate among different kinds of citations, such as conference proceedings, journal articles, and patents. Also, new encodings were needed for Subject and Identifier, such as PACS for the Physics and Astronomy Classifications Scheme, ACMCCS for the ACM Computing Classification System, and ISSN, CODEN, or publisher-specific journal codes for identifying journals. In these cases we just created our own encodings or types which we defined in our local RDF Schema.

5 Conclusions

Our initial overall assessment of DCQ and RDF for scientific and technical journal article metadata is very positive. However several concerns remain:

- There is a steep learning curve associated with RDF; this will dissuade some potential implementers. The value of RDF must be better described to provide sufficient incentive.
- DCQ documents imply that local extensions are allowed and even encouraged, but existing documentation is weak as to best practices and requirements for creating local extensions.
- The algorithms for transforming from DCQ in RDF to unqualified Dublin Core have become highly complex and dependent upon access to RDF Schemas; this alters the original connotation of "dumb-down."
- When using RDF there are opportunities to extensively modularize metadata structures (e.g., RDF Statements); but the extent to which this should be done, or is desirable, remains unclear.

6 References

[1] W. H. Mischo and T. W. Cole. "Processing and Access Issues for Full-Text Journals," in *Successes and Failures of Digital Libraries,* S. Harum and M. Twidale (eds.). Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, 2000, pp. 21-40.

[2] http://dli.grainger.uiuc.edu/

[3] http://www.crossref.org

[4] H. Van de Sompel, P. Hochstenbach, and O. Beit-Arie. *OpenURL Syntax Description*. Ex Libris, 2000. http://www.sfxit.com/openurl/openurl.html

[5] S. Kokkelink and R. Schwänzl. *Expressing Qualified Dublin Core in RDF, Working Draft*. Dublin Core Metadata Initiative, 2001. http://www.mathematik.uni-osnabrueck.de/projects/dcqual/qual21.3.1/

[6] O. Lassila and R. R. Swick (eds*.). Resource Description Framework (RDF) Model and Syntax Specification*. W3C, 1999. http://www.w3.org/TR/REC-rdf-syntax/

[7] A. Powell and A. Apps. Encoding OpenURLs in Dublin Core metadata. *Ariadne Issue 27*, March 2001. http://www.ariadne.ac.uk/issue27/metadata/

[8] R. Iannella (ed.). *DC Agent Qualifiers, Working Draft*. Dublin Core Metadata Initiative, 1999. http://www.mailbase.ac.uk/lists/dc-agents/files/wd-agent-qual.html

https://doi.org/10.23106/dcmi.952106655