DOCUMENT RESUME

ED 482 097 IR 058 785

AUTHOR Dingley, Andy; Shabajee, Paul

TITLE Today's Authoring Tools for Tomorrow's Semantic Web.

PUB DATE 2002-04-00

NOTE 12p.; In: Museums and the Web 2002: Selected Papers from an International Conference (6th, Boston, MA, April 17-20, 2002); see IR 058 778.

AVAILABLE FROM For full text: http://www.archimuse.com/mw2002/papers/dingley/dingley.html/.

PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)

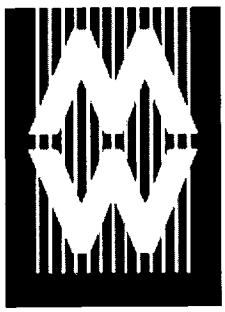
EDRS PRICE EDRS Price MF01/PC01 Plus Postage.

DESCRIPTORS *Authoring Aids (Programming); Biodiversity; *Computer Software Development; Conservation (Environment); Information Technology; *Metadata; *Museums; Standards; World Wide Web

IDENTIFIERS Information Infrastructure; Ontology

ABSTRACT

This paper reports on the development of a prototype authoring tool developed as part of on-going research around the needs of the ARKive project. The project holds text, rich-media and descriptions of factual statements about bio-diversity and conservation information. A key user community is that of school age children, requiring the mark-up of educational metadata in open standards, such as IEEE LOM (Learning Objects Metadata). This publishing architecture is intended to serve a range of audiences (ages, language and level of language skills). By storage of the content as discrete units, with extensive metadata describing each one, units may be retrieved and served to the audience as appropriate. Future developments may extend this to support ad hoc queries, not just rigidly pre-defined standard pages. Authoring development has shown that a simple and pragmatic tool based on Microsoft Word may still address advanced technologies, such as RDF, DAML and the future of the Semantic Web. Careful design has separated the process of describing a museum's exhibits, and the problem domain of the museum's area of interest. This improves the flexibility of solving the initial problem, allows the same code to be re-used on other projects, and assists publishing into other metadata formats. (Contains 10 references.) (Author/MES)



A&MI

Archives & Museum Informatics
2008 Murray Ave.
Suite D
Pittsburgh, PA
15217 USA
info@archimuse.com
www.archimuse.com

PAPERS
Museums and the Web 2002

Today's Authoring Tools For Tomorrow's Semantic Web

Andy Dingley, Codesmiths, Bristol and Paul Shabajee, Graduate School of Education & Institute for Learning and Research Technology, University of Bristol

Abstract

This paper reports on the development of a prototype authoring tool developed as part of on-going research around the needs of the ARKive project. The project holds text, rich-media and descriptions of factual statements about bio-diversity and conservation information. A key user community is that of school age children, requiring the mark-up of educational metadata in open standards such as IEEE LOM. A previous paper by the authors reported on the publishing architecture for this project. This publishing architecture is intended to serve a range of audiences (ages, language and level of language skills). By storage of the content as discrete units, with extensive metadata describing each one, units may be retrieved and served to the audience as appropriate. Future developments may extend this to support ad hoc queries, not just rigidly pre-defined standard pages.

Authoring development has shown that a simple and pragmatic tool based on Microsoft Word may still address advanced technologies such as RDF, DAML and the future of the Semantic Web. Careful design has separated the **process of describing** a museum's exhibits, and the **problem domain** of the museum's area of interest. This gives two advantages. First, most of the effort now supports a generic on-line museum that may be re-targeted from bio-diversity to any other topic. Secondly, solving the problem domain by ontological descriptions, not rigid program code, gives the ability to easily reference pre-existing or external vocabularies. This improves the flexibility of solving the initial problem, allows the same code to be re-used on other projects, and assists publishing into other metadata formats

The ARKive Project

ARKive (www.arkive.org) is a Wildscreen Trust (www.wildscreen.org) initiative to build a Web-based multimedia digital archive of the world's endangered animals and plant species. The project brings together thousands of films, videos, sounds and photographs of threatened and recently extinct species. Hewlett Packard Laboratories (http://www.hpl.hp.com/arkive/) are supporting ARKive by funding a research team to develop the technical infrastructure, including the content management software.

The effective and efficient educational use of these materials is fundamental to the aims of the project. A previous paper (Dingley and Shabajee, 2001) described a publishing architecture to offer content tailored to the dynamic needs of multiple educational user groups. This

paper describes the on-going development of a prototype-authoring tool to provide content to support this.

This paper reports on the development of a prototype-authoring tool developed as part of on-going research around the needs of the ARKive type projects by the authors. The work reported here is taking place independently of, and in parallel with, the development of the ARKive project by the Wildscreen Trust.

Project Teams

ARKive and similar projects incorporate many groups, each with its own agenda for the content authoring problems. The film-makers have little interest in authoring, but need shot-logging of the existing film library items (the process of describing scenes and shots on a piece of film and the objects they depict, often to extreme detail). Educators are interested in editing multiple copies of the same content, re-written for different audiences (language, content target age, language skills). Scientists require statements of complex facts in a way that is machine-accessible to allow comparison between species, and that may be published to other sites. The Semantic Web (Berners-Lee, 1998) researchers were interested in the techniques, more than the content.

The Publishing Architecture

The core of the prototype publishing architecture is that of dynamically assembling the publication from a large set of content units, each identified by detailed metadata. Some content units contain knowledge of different topics; others are the same material re-formatted for a different audience.

Content units

A content unit may contain different media types: plain text, machine-processable expressions of facts, images, video, as examples, although this list is deliberately open-ended for any new formats. All content units appear identical to the data store; they are stores of opaque content, with an associated transparent set of metadata. This metadata describes the application content ("this is a lion", "feeding behavior is shown"), the technical format ("this is a QuickTime video"), and a great deal of additional description ("the narration is in English", "the scene of the wildebeest being eaten is not suitable for 7-year -olds", "the copyright owner is..."). For many text units, the metadata is several times the size of the content itself – this is by deliberate design, and in any case, all space concerns are dwarfed by the video.

The scope of a content unit may be variable. In the simplest case, for ARKive, each unit contains a "species page"; a complete description of one species for one audience. This is equivalent to the current Web site; a traditional database-backed collection of static pages. The next stage is to provide content units for each section of the page (appearance, habitat, distribution etc.) and to duplicate these units for each audience and language combination. Producing a page is now a filtering operation; the relevant set of units is retrieved for that species and then filtered to choose the most appropriate unit of each set for the target audience. The set of units may itself be filtered: a general interest audience might not receive some detailed scientific content, a limited browser device might

have images but not video, and young children might be spared some carnivorous images. Overall ratings e.g. PICS (http://www.w3.org/PICS/) or IEEE LOM (IEEE LTSC, 2001) may be aggregated from those stored on each content unit.

Authoring and Storage

Content units are authored as part of a species-specific document, as this is the way the researchers and authors work. The species documents produced by the authoring tool reflect this scope. Each contains a simple header of workflow information, and then a list of content units. There is no implied structure of these units; they do not need to follow the paragraphs or section headings in the anticipated published species page.

"Research notes" may be stored at the level of the species document, or at each content unit. These are very simple free-text notes, intended for the author's guidance, and are never published. Our authoring workflow must cope with multi-author authoring, typically where a species expert may write a terse scientific description which a content author then re-writes or extends for a target audience. Writing appropriately for younger audiences, or translation, are expected use cases of this.

Content units are to be stored in a large RDF data store. The development of such stores is an area of current research, and so for pragmatic reasons we have deferred its implementation. In the future, each new species loaded will become part (although still identifiable) of this greater whole. At present we keep each species document as a separate XML file.

The future of ARKive may move away from this dependency on the species pages. Content might also be published as a habitat description (authored for one species, and then made available to all that share that habitat), a description of rare or endangered geology, or a cross-species description of British woodland carnivores.

Factoids

Early work in Web knowledge bases expressed descriptions of knowledge in a human readable format. The next step was to publish this knowledge in a way that made it machine accessible. This may be termed the "bottom up" approach (Motta, Buckingham Shum & Domingue, 2000). The contrary "top down" approach first produces a sufficiently expressive data model to meet the consumer's needs, distributes this by manual or automatic means, and then describes the content so as to meet the data model. As interested parties can now either use these shared standards, or at least transform their own representations in and out of it, this can form the basis of a Semantic Web. Dublin Core (http://purl.org/dc) is an example of this approach. Although sometimes dismissed as a "mere" lowest common denominator, widespread Dublin Core would still represent a considerable advance over most current practice.

Real progress in more intelligent ARKive type sites also requires what we have termed "factoids", an internal expression of an external fact. Although common standards for exchanging data can improve interoperability, a site that wishes to use this information to drive searches must also have some level of understanding about its meaning

– an ontology (see below).

In the case of ARKive, the data model represents its knowledge as a large set of content units, which may each contain factoids. Factoids may be seen as workflow units, allowing their source, completion and validation status to be tracked. They may also refer to a property whose meaning is defined in an ontology (see below). This meaning may only need to be opaque, but distinctly identifiable, i.e. it is still useful to recognize properties that are meaningful to compare, even if there is no machine understanding of their meaning. Our experience is that a factoid-based solution has these advantages over a text-based solution:

- Searching and general machine processing. There's a limit to what is possible with simple free-text searching, and any knowledge base as interesting as ARKive is far beyond this.
- Reification (making statements about statements), which offers the opportunity for validation and maintenance.
- Keeping track of where facts are referred to from other texts or media; e.g. species information or narration for a piece of video.

Content units contain factoids. A factoid is ARKive's expression of a fact, which is an externally pre-existing component of knowledge. A factoid will make reference to an ontology, and to be useful to the world outside our project, this ontology must be expressed in a communicable format, such as DAML+OIL (http://www.daml.org/).

Ontologies

An ontology is a formalized description of classes (things) and their related properties (statements which may be made about these things). The simplest level of processing merely identifies these properties, so that related properties may be automatically recognized as comparable. Apples and oranges both have a comparable property for "country of origin", but "number of segments" only applies to one. More sophisticated reasoning may allow inferencing that oranges and lemons are both citrus fruit, and so lemons may also have this property.

Use of Ontologies

The prototype uses DAML+OIL to represent its ontologies. They are used in two places. One represents the content unit and factoid structure. The other (of which there may be multiple instances) represents the external and publishable definitions of facts used by the factoids.

Content Store

The structural ontology for the content units is primarily a data-modeling exercise and would be familiar to any software developer with a background in object orientation or relational databases.

Descriptive Vocabularies

The descriptive vocabularies are simple in structure and mainly contain vocabulary lists. Vocabulary items may themselves have structure; an identifier unique within that vocabulary, a title (which may be repeated in

other languages) and an optional description of their appropriate meaning.

Some of these vocabularies (target audience, target language, rich media format) are likely to be created and controlled by ARKive. They may also contain representations of each item in other well-known vocabularies; i.e. some of the ARKive audiences also contain their equivalents in IEEE LOM. By presenting the authors with a choice from a single short and familiar list of target audiences, kept within the control of the project, they avoid having to make decisions about target ages based on a foreign country's school system.

The authoring tool loads vocabularies dynamically. A file directory is searched at start-up, and all those found therein are loaded. Authors may then select any of these to describe each factoid.

The Flying Bat Problem

There is a third use for ontologies within ARKive type systems: inferencing.

Searching ARKive for all the flying vertebrates should return both birds and bats. In simple implementations, this is likely to fail. Bats will be returned, because their flying behavior is unusual for a mammal and so is explicitly stated. For birds, flying is too trivial to describe and so it will probably never have been expressed in a machine searchable form. Even if birds' flight were to be stated for all birds, this would represent an enormous expansion of data volume and authoring work.

A common solution to this is to list a long string of unstructured keywords, hoping that they include all such concepts. The problem then is that although this improves the situation slightly, by offering a low-cost means of stating the trivial facts, it is so vague, unmanageable and restricted as to be near-useless. An obvious example is that of penguins. Such a simple list can express set membership, but not set exclusion. Stating "doesn't fly" for penguins, then querying with a simple text search will now return the flying vertebrates as being bats and penguins, but will still exclude eagles.

The power of an ontology is that it allows automatic inferencing to solve this. It is possible to state simultaneously that "birds fly", "penguins don't fly" and "mammals don't fly", and for a suitable reasoner to then determine correctly that "bats and birds other than penguins can fly", based on this ontology and an authored statement describing bats, the exception. In a sufficiently large knowledge domain, this ability to generalize and infer is essential, if the authoring requirement is not to be impossible.

Inferencing may also require an audience-related qualification. A search for "large fish" should exclude whales and dolphins, being marine mammals, but should this still be true for a 6-year-old Captain Ahab who is unaware of the distinction? This topic of ultimate accuracy over understanding is one of ongoing debate with our educational experts, although the technical team is still keen to represent it, as a technology demonstrator.

Ontology Tools

Ontology authoring is complex, and assisted editing tools exist for it. We evaluated both Protégé (http://protege.standford.edu) and OilEd (http://www.ontoknowledg.org/oil). Sadly, the standards for our chosen ontology description language (DAML+OIL) are changing in advance of the tool support, and so we found that reverting to a simple text or XML editor was necessary to use all of its features.

Protégé and OilEd have been developed by groups from different communities. Protégé's roots are in object and database modeling. It is easy to use for ontologies (like our content store) that approximate this class of problem. OilEd arises from the knowledge representation and reasoning field. We found it harder to use for the simpler ontologies, but more appropriate for the inferencing problems.

Nominals

An early prototype of the content and fact store was built. To avoid digression when species experts quibbled over our scientific data, rather than our techniques for storing it, we avoided biological data for real species and chose to describe Pokémon instead. The complete dataset is also conveniently available. This turned out to be quite a different problem from describing our real-world species, much simpler, yet illuminating.

A "hard" problem in knowledge representation is that of nominals. These are sets of values that are somewhere between the ontology and the instance data. An ontology might have a formalized representation of "fish" and "fowls" as distinct classes, and of color as a property, but it does not describe a set of "blue birds" or "silver fish". In many instances though, it is necessary to reason about these sets just as if they were described as classes in the ontology.

For Pokémon, there were no nominals. All of the descriptive properties that were required, and would *ever* be required, were clear to us from the outset. The Pokémon ontology could thus describe all of the necessary classes itself. For our real-world data, we continually encountered emergent properties: ways of structuring the data that were not apparent until authoring its content itself. Some antelope have a style of movement called "pronking" (trotting with all four feet moving together). Describing one antelope that pronks is simple (text will suffice), but a second pronking antelope should use a categorization factoid for its movement that is identical (not just a matching word in free-text), even though this term does not appear in the vocabulary.

Interoperability

Interoperability with other sites turns out to be relatively simple, compared with the complexity of our own internal authoring process. Our embedded metadata publishing used Dublin Core and extensive use of its qualifier mechanism. Where suitable established standards existed (e.g. IEEE LOM) we stored their equivalent terms within our own vocabulary lists, then published the well-known term in preference to our internal identifier.

For the publication of metadata from a Web site, a small amount of knowledge and effort can soon bring a site to a level in advance of likely practice for the near future. Existing good practices such as publishing

stable URL's, avoiding meaningless identifiers in URL's, etc., are still as valuable as they are for static sites.

Implementation

What we didn't do

Flat Text

Although ARKive intends to serve content in many final forms, most of them are heavily textual. The existing database-backed site simply holds a copy of the HTML code for each page, but in a SQL database. This is not using the power of the database as anything other than a content management system.

Issues of data accuracy were not a major drawback to flat text. Although simple typing errors are an obvious problem, the real problems are caused by subtle semantic errors, not simple syntax (e.g. issues related to interindexer consistancy as described in Markey 1984). This is particularly the case for judgment calls during video shot-logging: one person's notion of a "long shot" may be another's "close-up"

A major factor against the use of plain text was the development effort it would still require. It was always accepted that the major effort would need to be in supporting the authoring effort of selecting appropriate terms from large controlled vocabularies, particularly those for systematics and species taxonomy (Biosis 2002). These would always require authoring tools, no matter what the final format, and so the savings owing to a simplified format became proportionately less.

As may be seen later, the path we did choose fits in well with the integration of large amounts of pre-existing flat text. The existing ARKive site already contains data and text on >100 species, and a legacy integration path had always been required. The main issue with importing legacy data turned out to be that of data quality, particularly in mapping informal taxonomic labeling onto a more rigidly structured vocabulary.

The DTD approach

In this scenario, a purely XML approach would have been taken. An initial DTD or XML Schema would have been produced, describing the data model used to represent the content.

Within this scenario, there are two possibilities: one based on a DTD and one on a Schema. Even though XML Schema is now a long-established standard, it is still ignored by the majority of DTD-based authoring tools. XML Schema was an attractive basis to build the future of the project around, remembering that this is still early days for a hopefully long-term archive, but obsolescent DTD's were certainly not.

Existing XML authoring tools were considered, but rejected for their dependencies on DTD's. This approach would probably have been the quickest from the viewpoint of development time and just-sufficient integration.

Custom Code

Writing our own authoring tools from scratch, probably in Java, would have been the most flexible. This could have used the many pre-existing XML tools, including RDF parsers such as ARP (Carroll, 2001). Java would, however, have required more effort to build an editor, slightly more complexity for deployment ,and, most importantly, it would have been an unfamiliar editor for the authors. There was also a scheduling problem, in that it would not have been available for authoring, however simple, until almost all of it was completed.

Platform

The chosen platform was that of Microsoft Word and its Visual Basic for Applications programming language, with the Microsoft Windows XML component. This offered a reasonably competent coding language, no need to code a text editor, and (most importantly) immediate familiarity for the content authors. None of Word's built-in HTML or XML features was used, as Word's view of appropriate XML use is not quite the same as that of other workers in the field. There are no pre-existing RDF tools accessible from this platform.

The code, and its integration with the editor, is very simple. Normally the author sees only a standard version of Word, with a template containing custom styles. To insert specific items (new blocks of content, links to rich media, machine-processable facts) a toolbar button is pressed, a dialog allows appropriate values to be browsed, and a block of Word-formatted text is then inserted. Word's styles are used to identify properties within the data model. To export the completed document, a macro then walks through the Word document, translating each paragraph and appending it to the XML DOM component, then finally saving the resultant XML.

An advantage of this approach is for project scheduling, always a problem for software development. It is very simple to generate simple documents, thus allowing early testing by users. Most of the development complexity and effort goes into adding the more sophisticated features, which may grow incrementally as they are coded.

The exported document is in RDF, serialized as simple XML. Experience with the video shot-logging sub-project showed that RDF is transformable by XSLT, although this is a painful process! An RDF data model representing the same content may be serialized to a number of valid XML representations, yet XML tools (Schema or XSL) have no conception that these are equivalent. Although workable XSL stylesheets may be coded, their reliability depends on their author having manually foreseen and coded for every valid variation in the representation of RDF. This complexity echoes the experience of other groups (Cawsey, 2000). In an early and optimistic phase of the project, it was thought that automatic tools could be developed to produce these tools in turn. This proved not to be the case.

Why XML is a given

One issue that did not arise as a point for discussion was that of using XML. It now seems that XML is the *sine qua non* of such applications, with no obvious competition. This was interesting, as the team also included at least one person with a background in SGML.

The main reason for choosing XML, particularly over SGML, is related to the availability of tools, rather than the qualities of the format itself. A consistent DOM is available for XML, from several environments. Although true code portability is still something of a pipe dream, this allows the Java developer community to at least hold a conversation with the Word developers.

We chose to represent the authored content with RDF, primarily because this is the technical team's core research interest, but it is also a useful format. RDF tools are still immature and offer little immediate advantage, but by careful design (primarily control of the serialization into XML) it is possible to treat RDF as if it were simply XML. If using RDF had any major costs associated with it, i.e. the simple XML / XSLT approach would have suffered, then we would have chosen not to use it. Creation of RDF is a simple matter, even with a purely XML toolset, but importing valid RDF from other sources is not a practical proposition without either a genuine RDF parser, or by imposing additional constraints on how the RDF is represented in XML.

Software Availability

The authoring software developed as part of this project has been open-sourced and made available (http://www.xcml.org/docbadger/).

Observations on External Standards

Vocabularies

Few of the vocabularies and thesauri we needed were available in any formal notation. Those that were are mainly the educational vocabularies, e.g. LOM.

Several of the subject-specific vocabularies, particularly for animal taxonomy, conservation status (http://www.wcmc.org) etc. were available in text form, with good provenance and stability. Various Web-scraping scripts were able to transform these into RDF or DAML documents. Other projects, such as Tim Berners-Lee's Semantic Web Road Map (1998), http://www.w3.org/DesignIssues/Semantic.html, have taken a similar approach.

Property Sets

There is still no common adoption of property sets, for many common tasks. Dublin Core is an obvious solution to many of these, but there is scope for much improvement here. A surprising omission was bibliographic references, surely both a commonplace and easily formalized task. The de facto standard BibTex (http://www.isi.edu/webscripter/bibtex.o.daml) is still focused on solving the typographical problem of producing one's own papers, not the interoperability problem of sharing with others. While some groups wrestle with fundamentally difficult problems, a great advance towards a Semantic Web could be made just by wider adoption of the simplest steps.

Conclusions

- It is possible to develop useful authoring tools today, for full exploitation by tomorrow's Semantic Web. This is so, even if the extra information and structure captured today will not be exploited for some time to come.
- Despite its complexity, this solution remained independent of the ARKive problem domain. The generalized editing tool and description structure could be applied to the context of any museum or knowledge collection.
- Ontology tools are already useful for three aspects: structure of the content being created, for a thesaurus of descriptive terms, and for reasoning to infer knowledge about each item from a generalized description of the whole domain. These techniques are still far from mature, and the two areas aren't joining up yet.
- Pokémon aren't Gannets. Building a solution for gannets, which are not fully described before beginning to develop the solution, involves a complex issue, that of nominals.
- Issues of identifying and describing items consistently are significant. This will require either huge effort on maintaining consistency during authoring, or search and access tools that can resolve these gray matches and support validation and quality assurance tasks as part of an overall authoring workflow.

References

Berners-Lee, 1998. T. Berners-Lee (1998) Semantic Web Road map, http://www.w3.org/DesignIssues/Semantic.html

Biosis, 2002. Zoological Record Systematic and Subject Thesaurus http://www.biosis.org/zrdocs/zr_thes/systthes/

Carroll, 2001. Carroll J., (2001) Another RDF Parser http://www.hpl.hp.co.uk/people/jjc/arp/

Cawsey, 2000. Cawsey A. (2000) Presenting tailored resource descriptions: Will XSLT do the job?, WWW9, Amsterdam http://www.cee.hw.ac.uk/~alison/www9/paper.html

Dingley & Shabajee, 2001. Dingley A. & Shabajee P., Use of RDF for Content Re-purposing on the ARKive Project (2001) *International Conference on Advanced Learning Technologies*, Madison

Hunter & Lagoze, 2001. Hunter J. & Lagoze C. (2001) Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles, WWW10, Hong Kong, http://archive.dstc.edu.au/RDU/staff/jane-hunter/www10/paper.html

IEEE LTSC, 2001. IEEE LTSC Learning Objects Metadata (LOM), http://ltsc.ieee.org/doc/wg12/LOM_WD6-1_1.doc/

Markey, 1984. Markey K. (1984) Interindexer Consistency Tests: A Literature Review and Report of a Test of Consistncy in Indexing Visual Materials, *Library and Information Science Research*. 6: 155-177

Motta, Buckingham Shum & Domingue, 2000. Ontology-Driven Document Enrichment: Principles, Tools and Applications, *International Journal of Human-Computer Studies* , 2000, 52, (6), 1071-1109.

Wielinga, 2001. Wielinga B.J., Schreiber, A.Th.,Wielemaker, J. ,and Sandberg ,J.A.C. (2001) From thesaurus to ontology. *International Conference on Knowledge Capture*. Victoria, Canada

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

X	This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.
	This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").