Metadata Editing by Schema

Hussein Suleman

Department of Computer Science, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa
hussein@cs.uct.ac.za

Abstract. Metadata creation and editing is a reasonably well-understood task which involves creating forms, checking the input data and generating appropriate storage formats. XML has largely become the standard storage representation for metadata records and various automatic mechanisms are becoming popular for validation of these records, including XML Schema and Schematron. However, there is no standard methodology for creating data manipulation mechanisms. This work presents a set of guidelines and extensions to use the XML Schema standard for this purpose. The experiences and issues involved in building such a generalised structured data editor are discussed, to support the notion that metadata editing, and not just validation, should be description-driven.

1 Introduction

Editing of structured metadata over Web interfaces introduces complexities because of the fixed structure of standard HTML interfaces. A typical problem occurs when a metadata field is repeatable, as there is no simple way to duplicate a single field in a static HTML form. As an example of this, users may have multiple first names (e.g., Goolam Muhammad in Arabic), multiple last names (e.g., Guzman Aranda in Español) or both. In order for any metadata format to correctly capture information about individuals, there has to be flexibility in the data format as well as the input mechanism. Thus, the number of first names and last names should ideally be variable, without requiring data and providing a facility to add more fields of that type as needed.

This problem is exacerbated when there are nested metadata elements, e.g., a name element containing separate sub-elements for first and last names. If there are multiple names in addition to multiple first and last names, then the input mechanism must cater for repeatability at different levels within the metadata.

Many existing Web-based metadata tools, such as Meta builder [1] and DC-dot [2], use fixed formats for their input forms, thereby placing restrictions on the metadata format due solely to the input mechanism used. A general solution to this problem requires the creation or use of a general-purpose metadata specification language and a tool to interact with users to perform the required editing based on the specifications for metadata formats. Initial work was done in devising such a format in the Web Characterization Repository project [3]. In that project, metadata formats for different types of resources (papers, tools, etc.) were specified in terms of tabular descriptions supporting only flat formats without nested elements. A similar technique was employed in the Mantis project [4] from OCLC, which used a Java applet to support modification of forms without additional server interaction. The Reggie and MetaEdit products from the MetaWeb project [5] also used Java applets and a home-grown format and XML DTDs respectively for field specification. More recently, the EPrints software [6] uses a home-grown field specification language to generate standard HTML forms when editing metadata.

With the emergence and growing popularity of the Open Archives Initiative's Protocol for Metadata Harvesting [7], an increasing number of digital library systems are using XML Schema [8] to define their metadata formats precisely. XML Schema is a declarative language to specify the format of XML files. For metadata formats encoded in XML, only a subset of the full XML Schema specification is required since metadata representations usually have a rigid structure.

It was hypothesised in this work that XML Schema can be used as the basis for a metadata description language that can drive a generalised editing process. The current trend towards writing XML Schema for existing and new metadata formats provides a ready backdrop against which to develop tools for data input to complement the schema-based validation tools used by the W3C (e.g., XSV [9]) and OAI (e.g., Repository Explorer [10]). Thus, any XML record created with such an editing tool can be validated using schema-based validation tools before being stored or processed further.

2 Interaction Model

In order to test this premise, the MDEdit Perl module was built to drive an editing process based on a subset of XML-Schema, augmented by elements of user interfaces. MDEdit uses plain vanilla HTML for its user interface. As a result of this, every change in the input form structure requires a client-server interaction. While this is not the most efficient operation, this approach was taken to illustrate that generalised metadata editing is still possible in the worst-case scenario where a browser has no advanced interactivity functionality. This is critical if such methods are to be employed on small form-factor devices such as PDAs and cellphones.

Fig 1 illustrates the interaction model employed by the MDEdit module. For new metadata records, MDEdit reads in the schema file and generates an HTML form from it with placeholders for minimal numbers of elements as specified by the schema. Alternatively, when a metadata record already exists, MDEdit will fill out the values already known when creating the form. Thereafter, while editing the values stored in the form, the user may request additional input elements for a single field or set of fields (as allowed by the schema). The server will then regenerate the form with additional input mechanisms inserted into the appropriate position, while still retaining all the data already entered. Finally, the user submits the form, and the server then checks that the number and type of each field and subfield conform to the schema. If there are errors, the form

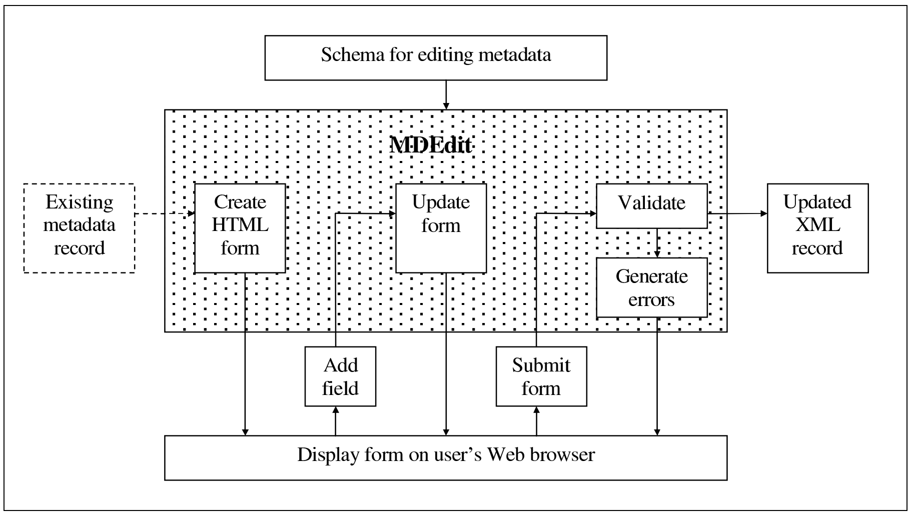


Fig. 1. Interaction between MDEdit, user, schema, and metadata record(s)

is regenerated with errors highlighted. If there are no errors, the data from the form is converted into an equivalent XML record, completing the process.

3 Data Types

MDEdit supports only a few data types of those available in the XML Schema standard, as these were deemed sufficient to provide input mechanisms for most popular metadata formats. The types MDEdit can operate on are:

– *string*
– *complexType*, containing a *sequence*
– *simpleType*, containing a *restriction* with *enumerations*

The following schema excerpt shows a possible general definition for names of people, with repeatability of elements at multiple levels of nesting, as discussed previously.

```
<element name="name" minOccurs="1" maxOccurs="unbounded">
   <complexType>
      <sequence>
         <element name="first"
             type="string" minOccurs="1" maxOccurs="unbounded"/>
         <element name="last"
             type="string" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
</element>
```

4 Extensions

While XML Schema is sufficient for defining the structure of metadata, specifying the visual elements of a user interface (e.g., number of rows in a list box) requires additional information. According to the XML Schema standard, every element may have an an *appinfo* annotation containing tags specific to the application. This was exploited to define application-specific extensions, some of which are listed in Table 1.

Table 1. Sample of additional MDEdit schema tags to define appearance of HTML forms

Tag	Description
<caption>	an alternate caption to use instead of the field name
<description>	an optional description of what the field is
<rows>	number of rows to use for the field display
<columns>	number of columns to use for the field display
<inputtype>	*password* - password entry box that displays * instead of characters *file* - file upload box *radio* - use radio buttons for list instead of <select>

An example of an annotation to a schema using these extensions is as follows:

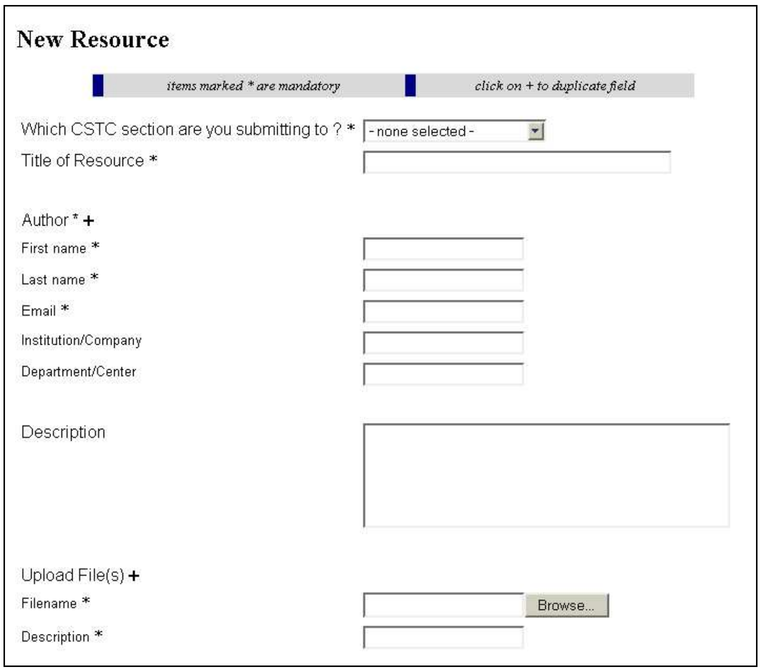
```
<element name="test" type="string">
   <annotation>
      <appinfo>
         <caption>Test Input Field</caption>
         <rows>40</rows>
      </appinfo>
   </annotation>
</element>
```

This example results is an input field in the HTML form with the text label *Test Input Field* and a text box 40 characters wide for data entry/editing.

5 Rendering

At each stage in the process, the internal representation of the data and type information is used to generate an equivalent HTML representation. The combination of schema type information, structural information and extensions is used to determine an appropriate and intuitive visual aspect and/or input device for each metadata field.

Fig 2 shows a typical HTML rendering of a simple annotated schema. The schema used to generate this interface can be found in [11].


New Resource

*items marked * are mandatory* | *click on + to duplicate field*

Which CSTC section are you submitting to ? *	- none selected -
Title of Resource *	
Author * +	
First name *	
Last name *	
Email *	
Institution/Company	
Department/Center	
Description	
Upload File(s) +	
Filename *	Browse...
Description *	

Fig. 2. Typical user interface generated by MDEdit

6 Analysis and Future Work

The MDEdit module has been used extensively while developing demonstrations of the Open Digital Library (ODL) [12] componentised model for building systems. Any loss of interactivity because of client-server communication is made up for by the generalisations possible because of the model and schema-driven nature of the tool. It has also been adopted for use on the website of the Networked Digital Library of Theses and Dissertations [13] to handle registrations of new members and addresses the problem that such registrations typically include varying numbers of individuals.

While the MDEdit tool is practically useful, it is more important as a demonstration of the principle that schema can be used to drive the process of metadata entry and editing. Various avenues remain to be explored in terms of such schema-driven metadata entry. These include:-

- extending existing tools to understand all aspects of the XML Schema specification, instead of just the subset used by MDEdit,
- investigating the feasability of serial entry drivers to create XML configuration files at a terminal, and
- using the User Interface Markup Language [14] as an intermediate representation so that the resulting interfaces can easily be retargeted to multiple devices.

While this project has implications for building digital libraries in a declarative fashion, it also vindicates the design of XML Schema as a multi-purpose descriptive language, with sufficient expressive power and extensibility to support functions beyond simple type-checking.

7 Acknowledgements

This work was funded in part by the US NSF (grant: 0002935) and was conducted while at and in conjunction with members of the Digital Library Research Laboratory at Virginia Tech.

References

1. Vancouver Webpages (2003), Meta builder. Website http://vancouver-webpages.com/META/mk-metas.html
2. UKOLN (2003), DC-dot metadata editor. Website http://www.ukoln.ac.uk/metadata/dcdot/
3. Suleman, H., E. A. Fox and M. Abrams (2000), "Building Quality into a Digital Library", Proceedings of the Fifth ACM Conference on Digital Libraries, San Antonio, Texas, USA, June 2000, pp. 228-229.
4. Shafer, Keith E. (1998), "Mantis Project Provides a Toolkit for Cataloging", OCLC Newsletter, No. 236, pp.21-23. Available http://www.oclc.org/oclc/new/n236/research_mantis_project.htm
5. Distributed Systems Technology Centre (2003), The MetaWeb Project. Website http://www.dstc.edu.au/Research/Projects/metaweb/
6. Open Citation Project (2003), GNU EPrints 2. Website http://software.eprints.org/
7. Lagoze, Carl, Herbert Van de Sompel, Michael Nelson, and Simeon Warner (2002), The Open Archives Initiative Protocol for Metadata Harvesting Version 2.0, Open Archives Initiative, June 2002. Available http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm
8. Fallside, David C. (editor) (2001), XML Schema Part 1: Structures and Part 2: Datatypes, W3C, 2 May 2001. Available http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/
9. Thompson, Henry S., and Richard Tobin (2003), XML Schema Validator. Website http://www.ltg.ed.ac.uk/ ht/xsv-status.html
10. Suleman, Hussein (2001), "Enforcing Interoperability with the Open Archives Initiative Repository Explorer", in Proceedings of the ACM-IEEE Joint Conference on Digital Libraries, Roanoke, VA, USA, 24-28 June 2001, pp. 63-64.
11. Suleman, H. (2002), Open Digital Libraries, Ph.D. dissertation, Virginia Tech. Available http://scholar.lib.vt.edu/theses/available/etd-11222002-155624/
12. Suleman, Hussein, and Edward A. Fox (2001), "A Framework for Building Open Digital Libraries", in D-Lib Magazine, Vol. 7, No. 12, December 2001. Available http://www.dlib.org/dlib/december01/suleman/12suleman.html
13. Fox, Edward A. (2003), Networked Digital Library of Theses and Dissertations. Website http://www.ndltd.org
14. Phanouriou, Constantinos (2000), UIML: A Device-Independent User Interface Markup Language, Ph.D. dissertation, Virginia Polytechnic Institute and State University.