DRTC – ICT Conference on
Digital Learning Environment
11th –13th January 2006
DRTC, Bangalore

Paper: N

Implementing LOM Schema in DSpace

A.R.D. Prasad
Documentation Research and Training Centre
Indian Statistical Institute Bangalore
ard@drtc.isibang.ac.in

Abstract

DSpace facilitates Qualified Dublin Core. There have been attempts to extend DSpace ability to host other metadata schemes e.g. electronic theses and dissertations like Tapir. However, the user community has often expressed the requirement for other metadata formats like VRA core, IMS LOM etc. Support for many metadata formats will greatly enhance the use of DSpace and the type of resources that could be preserved using DSpace. Since version (1.2.2), DSpace comes with much needed input forms with which one can define ones own submission forms. This feature provides additional ability of adding any metadata format to DSpace, in addition to existing Qualified Dublin Core. This papers attempts to provide guidelines for modifying DSpace files to accommodate additional metadata formats like IMS-LOM.

1. Learning Object Metadata (LOM)

The LOM standard is also referred as IEEE P1484. The full name of the standard is "Standard for Information Technology – Education and Training System – Learning Object and Metadata. There are 4 sibling standards in IEEE series.

1. 1484.1.2.1: IEEE standard for Learning Object Metadata
2. 1484.1.2.2: Standard for ISO/IEF 11404 Binding for Learning Object Metadata Model
3. 1484.1.2.3: Standard for Learning Technology – XML Schema Definition Language Binding for LOM
4. 1484.1.2.4: Standard for RDF Binding for LOM Data Model

Some of the basic unqualified data elements of LOM are given below, as full list of data-elements is quite large.

- *General:* General information about the learning object as a whole
- *Identifier:* A globally unique identifier
- *Catalog:* Identification or catalogue entry for the entry. Ex: ISBN, URI
- *Entry***:** The value of the id within the identifier
- *Title:* Title of the learning object
- *Language:* Language or languages used in the learning object
- *Description:* Text describing the document
- *Keyword:* Keywords representative of the learning object content
- *Coverage:* The time, culture, geography to which learning object belongs
- *Structure***:** Organizational structure of the learning object
- *Aggregation Level:* granularity of the learning object
- *Life Cycle:* History of the current status of the learning object
- *Version:* The edition statement of Learning Object
- *Status:* Completion status like draft, final, revised etc
- *Contribute:* People and organizations contributed to the learning object
- *Role:* Kind of contribution like author, publisher, editor etc

- *Entity:* Identification of people and organizations that have contributed
- *Date:* Date of contribution

2. DSpace and Metadata

Any digital library software like DSpace should be ideally metadata-schema-independent, so that the DSpace administrators may adopt whichever metadata scheme is most ideal to his repository or a collection in his repository. Presently DSpace follows Qualified Dublin Core for input and exposes Unqualified Dublin core through OAI-PMH protocol, Dublin is the only W3C approved metadata schema and generally accepted as lowed common denominator.

With the introduction of '`inputforms.xml`' in DSpace 1.2.2 version, it has become easier to create user defined submission workflow. The earlier versions of DSpace, always offered mechanism to add non-DC elements, though for submission workflow screens for metadata input, one has to modify the java programs. This was typically the case with tapir (2), which incorporated a few metadata elements for describing electronic theses and dissertations (ETD). Of course, tapir has many java programs for additional functionalities (required for ETD) other than submission workflow.

To accommodate a metadata schema other than Dublin core like LOM, the following issues are to be addressed.

1. Adding New Elements to the database
2. Input Forms for submission process
3. Indexing
4. Display of search results
5. Import/Export
6. OAI-PMH and crosswalks

2.1 Adding New Elements

One should undertake a good study of the available metadata schemes to identify the best-suited scheme for a collection of digital resources. For example, if the collection contains images, one should perhaps adopt VRA

Core. It is necessary to identify which elements of the newly adopted scheme already map with existing Dublin Core elements, so that the elements not mapped with DC can be created. This is normally achieved by adding metadata elements either through DSpace administration or directly to the 'dctyperegistry' table in PostgreSQL. Though, ideally, DSpace administrator should be able to create different registries for the required schemes in addition to Dublin Core, creating different tables for each scheme may result in programming complexity. It is simpler to rename the 'Dublin Core Registry; to 'Metadata Registry' and allow the administrator to add elements from various metadata schemes under one umbrella i.e. 'Metadata Registry'. This is fairly a trivial issue and could be implemented in all the DSpace versions.

2.2 Input Forms for submission process

Since DSpace 1.2.2, one can define the submission workflow in 'input-forms.xml' however, if a repository adopts many metadata schemes for various collections, the inputforms.xml will tend to be large and unwieldy to make modifications. Perhaps, a kind modular approach may be adopted in the future versions of DSpace, where each schema will have separate file and the inputforms.xml file only contains the information on collection handle and the respective file to be used for the elements to appear in input form.

2.2.1 Tags in input-forms.xml

Input-forms.xml:
For a clear understanding of the notation, $DSPACE_HOME is the home directory of DSpace installation. It could be any directory like '/dspace' or '/home/dspace' or '/user/local/dspace' or '/opt/dspace'. For example, if you come across `$DSPACE_HOME/config`, it means the 'config' directory under DSpace home directory where DSpace is installed.

If one wants to create a separate input form for LOM, one has to modify the `$DSPACE_HOME/config/inputforms.xml` file. The file 'input-forms.xml' is fairly well commented. However, more notes are given in the following paragraphs.

Just as a CDS/ISIS database can have any number of input worksheets, in DSpace provision is given to create any number of input forms for a given digital library. One can define an input form for each collection in DSpace. Though it is not mandatory to have collections created by the type of document, the general practice is to create collections by type of digital document. For example, the collection can be books, theses and dissertations, learning objects etc. For such an organization of collection, each collection required a different set of metadata elements. These sets of metadata ideally should be in conformity with existing standards. For example, for theses one may follow ETDMS; for learning objects one may follow LOM and for general collections one may follow Dublin Core. One may create any number of input forms, however, all them should be defined in the $DSPACE_HOME/config/input-forms.xml file.

<form-map>

In this tag you should add the new forms with their 'form-name' along with associated collection-handle. e.g.

```
<form-map>
  <name-map collection-handle="default" form-
  name="traditional" />
  <name-map collection-handle="1849/4" form-name="lom" />
  <name-map collection-handle="1849/5" form-name="etd" />
  <name-map collection-handle="1849/6" form-name="lom" />
</form-map>
```

The third and fifth lines in above code indicates that form 'lom' will be used for the collections 1849/4 and 1849/6, whereas form 'etd' will be used for the collection 1849/4. The 'default' form will be used for the rest of the collections. If you remove the second line, i.e. the default-handle, you should ensure every collection in your repository is associated with a 'name-map' entry (This is not advised). It goes without saying that '<form-map>' should have an end tag i.e. </form-map>

<form-name>

As all the different forms are accommodated in the same file, they should be distinguished from each other with a name, which is called 'form-name'. Each input 'form-name' should be associated with one or more collections.

There is a form-name called ‘default’ which is used for collections where no form is defined. If you do not add your own input form, DSpace uses default form for input.

<collection-handle>
In DSpace, its handle addresses each collection. This is to ensure uniqueness, even if you have defined collection under different communities or sub-communities with the same name. For example, you have defined a collection by name ‘books’ under the community ‘Physics’ and also another collection with same name ‘books’ under the community ‘Chemistry’. The DSpace will have no way of resolving such situations. That is why collection handle is used, as handle is unique to each collection. You can see the handle of the collection at the bottom of the browser when you place your mouse pointer on the collection or when you click the collection, you may see the handle as a part of the URL in the browser. For example, if you see *http://drtc.isibang.ac.in/1849/4.* Here the collection handle is 1849/4.

For digital repositories not registered with CNRI, the handle may look like 123456789/4

Note: DSpace assigns handles not only to each digital object in the repository, but also to each community, sub-community and collection.

<**form-definition>**
This tag includes the definitions of all the forms and ends with an ending tag </form-definitions>. You should ensure that new forms are defined in between the starting and ending tag.

<form-name>
Each of the newly defined forms should start with <form name ... > and end with </form>. For example, if you have defined two forms, one for etd collection and the other for lom collection, you should have the following.

```
<form name=”etd”>
      ...
      ...
</form>
```

```
<form name="lom">
      ...
      ...
</form>
```

Here 'form' is the tag and 'name' is the attribute and 'etd' and 'lom' are attribute values.

<page>
DSpace allows you to have any number of pages in designing an input form. You may divide all the elements of particular metadata standard into any number of pages. What should go into first page or second page is immaterial and you can intuitively decide what elements should go in the first page, second page and so on. The syntax of this tag is

```
<page number="1">
      ...
      ...
</page>
```

<field>
Each <field> tag contains one element of metadata, where one can define various attributes of a field like its name, what is the data type, whether it is repeatable, what should be its heading in the input form, whether any help message should be displayed, whether it is mandatory or optional etc.

<dc-element>
This is the tag where we define an element in a metadata schema. As DSpace supports Dublin core, the tag name appears as 'dc-element'. e.g.

```
<dc-element>title</dc-element>
```

Though we can define a new element, which is not in Dublin core, still we should use this tag as 'dc-element'. Presently, there is no mechanism to use 'lom-element' or 'etd-element', as 'input-forms.xml' uses fixed pre-defined tags. However, this really does not prevent any one to define ones own elements from other metadata schemes. If the element does not have a qualifier, it requires only one line definition.

<dc-qualifier>
As the very name suggests, this meant to be a qualifier to an element. It cannot occur alone and should be preceded by <dc-element>. e.g.

```
<dc-element>contributor</dc-element>
<dc-qualifier>advisor</dc-qualifier>
```

<repeatable>
It can have two values i.e. either true or false. If the field is repeatable like author names, it should be true. If the field is not repeatable like date. issued, it is false. e.g.

```
<repeatable>true</repeatable>
<repeatable>false</repeatable>
```

<label>
This tag is used as the heading of the column. For example, `<label>`Title`</label>`, can be used as caption to the title field.

<input-type>
One should be clear about what type of input is required for a particular field. DSpace allows the following types of input. In DSpace, input type can be any one of the following:
name, textarea, onebox, twobox, date, dropdown, qualdrop_value

name: typically used for names of persons, where it will have first name and last name, but will be stored as one element in the database. Note that Dublin Core does not have two separate elements for first name and last name. It is assumed that they are treated as rendered. However, under the input-type 'name' DSpace allows you to enter last name and first name separately. While displaying the field, it shows in rendered format. That is the reason, in case of South Indian names, I suggest to enter first name of the author in last name column and last name (which is normally initials) in the first name column, so that in the display it will appear correctly. For example, to make it appear as Ranganathan, S.R. you should enter Ranganathan as the last name, though it is first name and 'S.R.' in last name!!!

textarea: used in case of abstracts or any other field where a large text requires to be entered.

onebox: creates only one input-column (box) to enter the data. e.g. Title

twobox: used in case of keywords, where we required the input area to appear in two columns, more a display sugar. One can do away with and can use one box.

date: meant for entering dates

dropdown: provides a drop down menu. For example, type of document, where the drop down menu may contain values like book, article, theses etc. The 'dropdown' should be used with value-pairs, which is explained below. e.g.

```
<input-type value-pairs-name="common
types">dropdown</input-type>
<input-type value-pairs-
name="common_iso_languages">dropdown</input-type>
```

qualdrop_value: It is mandatory to use it with repeatable fields. It is similar to two boxes, except the first box is used for drop down menu where you choose one of the values in the drop-down menu. The second box is used for entering data. For example, in case of Identifier, the first box, which is drop-down menu, will have values like ISSN, URI, etc and in the second box, you have to enter the actual ISSN or URI. The 'qualdrop_value' should be used with value-pair. e.g.

```
<input-type value-pairs-
name="common_identifiers">qualdrop_value</input-type>
```

<hint>

This is meant to provide help message to the end user. e.g.

<hint>In case of South Indian names enter first name in the last name column and the last name or initials in the first name column</hint>.

This help message appears below the input box.

<required>
Using this tag you can make some fields mandatory. If this tag is empty, DSpace assumes it is optional to fill up the values in a field. For example, DSpace makes title field mandatory and the rest of the fields optional. However, if you feel a few more fields should be made mandatory, you can make them. The optional fields are written as follows:

```
<required></required>
```

However, mandatory fields are written as, e.g. the title field can be defined as

```
<required>Enter the title of the item</required>
```

When a user does not fill up title and moves to the next page, DSpace shows the message embedded between the starting and ending statement of the tag and does not allow you to move forward until you enter a value in this field.

<form-value-pairs>
The following tags are defined outside the 'form-definitions'. If you do not have pull down menus in your input sheet, you can totally ignore these tags. These are mainly used for pull-down and qualdrop_value data types. Again these tags can be commonly used by any number of forms. For example, you have defined '`<form-definitions>`' for LOM, ETD, VRA Core etc, the value-pairs need not be defined for every form-definition, they can be shared by all the different meta-data schemes.

<value pairs ... >
Here, we define the name of each value pair and to which element of the metadata schema it is associated. e.g.

```
<value-pairs value-pairs-name="common_idetifiers" dc-
term="identifier">
```

The above line states that the name of the value pair is "common-identifiers" and it is associated with the 'identifier' element in the metadata scheme. When we define an element in the metadata scheme the definition (refer the earlier definitions under 'field') should look like the following:

```
<field>
         <dc-element>identifier</dc-element>
         <dc-qualifier></dc-qualifier>
      ...
          <input-type value-pairs-
name="common_identifiers">qualdrop_value</input-type>
      ...
            <required></required>
         </field>
```

<pair>
The 'pair' tag contains actual pairs of displayed values (in the drop down menu) and the values stored in the database. e.g.

```
      <pair>
            <displayed>ISSN</displayed>
            <stored-value>ISSN</stored-value>
      </pair>
```

2.3 Indexing

If one feels the necessity of indexing some of the newly added elements, one can modify dspace.cfg file, so that the required elements can be indexed. When you open the dspace.cfg, look for the lines given below:

```
##### Fields to Index for Search #####

# DC metadata elements.qualifiers to be indexed for
search
# format: - search.index.[number] = [search
field]:element.qualifier
#         - * used as wildcard

###   changing these will change your search results,
###
###   but will NOT automatically change your search
displays  ###
```

```
search.index.1 = author:contributor.*
search.index.2 = author:creator.*
search.index.3 = title:title.*
search.index.4 = keyword:subject.*
search.index.5 = abstract:description.abstract
```

The first column search.index[number] simply indicates how many fields you are indexing. In the second column you have to state which field should be indexed. There is a difference between the field names used by metadata schema like Dublin Core and the way Lucene search engine's naming pattern for the field name. DSpace uses qualified Dublin core where the notation is represented as 'fieldname.qualifier', the Lucene search engine uses only field names. Once the difference is clear, it should not be difficult to index the required fields in a metadata schema. First we mention under what field name Lucene should index, followed by ':' (colon), followed by metadata element, followed by '.' (dot), followed by qualifier or wild card e.g.

```
Search.index.11 = abstract:description.abstract
Search.index.8 = abstract:description.tableofcontents
```

The above two lines index the metadata elements description.abstract and description.table of contents under abstract. That is while searching, if you enter

```
abstract:database
```

If the term 'database' is found either in abstract or table of contents, it will be retrieved.

The use of wild card '*' for qualifiers indicates all the qualifiers under the element to be indexed. e.g.

```
search.index.4 = keyword:subject.*
```

indexes the words occurring in any of the qualifiers of 'subject' like ddc, lcc, lcsh, mesh, etc. under the field name 'keyword'. That means one can give a search expression like

```
keyword:lung
```

If the word lung occurs in any of the qualifiers like mesh, lcc, lcsh etc. it will be retrieved.

Note: Presently, use only the DSpace stated Lucene field name only. For example, if you use

```
search.index.14 = guide:contibutor.advisor,
```

it does not work. It requires a lot more modification in DSIndexer.java file.

2.4 Search Result Display

DSpace allows you to display all the elements with data in 'show full item record' though for short description of the record (default display), one has to modify the Java program, ItemTag.java. However, one should have a bit of programming knowledge to modify the display of search results. In ItemTag.java, there are two functions called renderFull() and renderDefault() or render(). We need not modify the renderFull() function, as it will display all the elements including the newly added ones. However, renderDefault() or render() should be modified, if you wish to display some of newly added elements. Here, the code is not given because DSpace verions 1.2.2 used the function renderDefault() and version 1.3.2 used the function render() for default rendering. In any case if you modify any Java programs, you have to run

```
ant update
rm -rf $TOMCAT_HOME/webapps/dspace*
cp $DSPACE_SOURCE/build/*.war
$TOMCAT_HOME/webapps/
$TOMCAT_HOME/bin/startup.sh
```

2.5 Import/Export

Normally, when a collection is exported, DSpace exports the bitstream, handle, license, contents and dublin_core.txt file for each item. Though the dublin_core.txt file contains dc-like format, it is not the format that DSpace exposes metadata through OAIcat. In fact, `dublin_core.xml` uses qualifiers too. In any case, as the purpose is to export metadata, it really does not matter, whether the element tags are in conformity with newly adopted metadata scheme. However, this will be an issue when one wishes to use the exported records in a digital repository which uses other than DSpace software, in which case one has to convert the entire set of records to suit the import operation of non-DSpace digital repository software. In

essence, Export/Import issue is a non-issue as long as these operations are performed within DSpace user community or digital repositories using DSpace.

2.6 OAI-PMH

This is the most critical aspect in dealing with the metadata schemes other than Dublin core. DSpace filters the Qualified Dublin Core elements and exposes only the Dublin Core elements in response OAI requests. It is fairly simple to add new metadata schemes to DSpace by modifying `oaicat.properties` files and writing a java program similar to that of OAIDC Crosswalk.java. Modification to `oaicat.properties` should result in responding correctly to the OAI verb `ListmetadataFormats`.

The Crosswalk.java for the new schema, should

1) extend Crosswalk class
2) have the name space entry for the new scheme
3) create crosswalk for the dcregistry entries for the new metadata format
4) have the tags of the new metadata format
5) suppress the dcregistry elements which are not relevant

One can borrow the idea from the recent versions of DSpace, as they expose contributor.author as creator. If one uses the new metadata scheme in its OAI verb, the output will convert all the records in the digital repository into the new scheme, which in fact may not be quite meaning full. For example, crosswalk from a collection described in VRA Core to ETDMS may result in not quite full data. This becomes more apparent when Harvesters collect the cross walked metadata. If the repository has more than one scheme, one has to resort selective harvesting on a collection by giving the metadata scheme used for that collection.

3. Conclusion

Many metadata formats are emerging and with so many schemes, it will become too difficult to ensure inter-operability. In addition, unlike Dublin Core, many metadata schemes have very large set of elements. Hence, these elements are categorized administrative metadata, descriptive metadata etc. Presently, DSpace can only address descriptive metadata, which is similar to traditional cataloguing. DSpace in the context of e-learning could only be used to offer courseware without the elaborate features of dedicated e-learning software like Moodle, Manhattan etc. where specific roles for teachers, students are assigned. DSpace model is simple, that it allows storage and retrieval of learning objects.

4. References

1. Dublin Core Metadata Initiative. *http://dublincore.org/*
2. Introduction to the Tapir for DSpace. *http://www.thesesalive.ac.uk/dsp_home.shtml*
3. DSpace at University of Manitoba. *https://mspace.lib.umanitoba.ca/ETDMS.zip*