NEW METHODS FOR ENHANCING THE EFFECTIVENESS OF THE DUBLIN CORE METADATA STANDARD USING COMPLEX ENCODING SCHEMES

István Szakadát, László Lois, and Gábor Knapp*

1. PROBLEM OUTLINE

After several pilot projects performed in the previous years (e.g.[1, 2]), the Hungarian Government decided in 2002 to start a new initiative for enhancing the digitization process of Hungarian cultural contents within the framework called the National Digital Archive (NDA). The most important question of the NDA concept was how the digitized cultural content can be integrated at as high a level as possible. In order to integrate the contents of the different archives within the NDA framework, two basic standards were offered to the participants: the architecture described by the Open Archives Initiative (OAI) helped the content integration, and the Dublin Core Metadata Standard was proposed for the data exchange format. During the implementation process we were faced with some problems when we wanted to apply qualified DC schemas.

Since the first workshop of the Dublin Core Metadata Initiative (DCMI), several authorities (including ISO,[3] NISO[4] and CEN[5]) standardized the basic 15 elements of the schema.[6] The Dublin Core Metadata Element Set was originally agreed at a workshop convened by the Online Computer Library Centre (OCLC) and the National Centre for Supercomputing Applications (NCSA) at Dublin, Ohio in March 1995. This invitational workshop convened selected librarians, archivists, humanities scholars, geographers and standards makers, with the specific aim of achieving consensus on a list of metadata elements which could produce basic descriptions of data in a wide range of subject areas. Due to its simplicity, the schema is widely accepted and used as the metadata exchange format among different digital object domains. The Dublin Core metadata are usually expressed

* Budapest University of Technology and Economics, Budapest, Hungary, syi@axelero.hu, lois@hit.bme.hu, knapp@mokk.bme.hu,

Information Systems Development: Advances in Theory, Practice and Education
Edited by O. Vasilecas *et al.*, Springer, 2005

using XML syntax. The basic elements are represented as tags, the values are as simple character strings, no attributes were defined. The resulting record can be easily parsed and validated by standard XML tools, however it remains human readable.

The Dublin Core as a whole is not so stable as some established metadata standards (like MARC) because it is a 'loose implementation model' which is evolving as a variety of people continue to refine it. The most important layer, the Elements of the Dublin Core standard are fixed. Interoperability was made possible by the stable and well defined list of 15 elements.

> "The simplicity of Dublin Core can be both a strength and a weakness. Simplicity lowers the cost of creating metadata and promotes interoperability. On the other hand, simplicity does not accommodate the semantic and functional richness supported by complex metadata schemes."[5]

For more sophisticated document description and search support the schema had to be extended. As a first step, DCMI qualifier attributes were introduced to refine the meaning of the basic terms and type attributes to specify the source or syntax of the value.[7] Qualified DC schemas appeared for a more precise description of different document types in several institutes (e.g. in the European Broadcasting Union).[8] Although the recommended set of qualifiers was published quite early, different needs resulted in different element and qualifier sets, and the semantic unity could only be sustained with difficulty, and the syntax of records became more and more confused.

The publication of DCMI Metadata Terms in 2003[9] can be considered as an attempt to make the Dublin Core based metadata schemas manageable, and support interoperability and semantic consistency, while retaining the simplicity and flexibility. The terms for various purposes were organized into a linear list. Application Profiles for the description of resource types using the terms defined by DCMI became consistent and they now support interoperability and novel terms can be easily defined. However, the association of elements and different extended term types (for refinement, encoding scheme or vocabulary term) is not complete and not always consistent.

Identifying the role and possibilities of current encoding schemes recommended for particular elements and extending their usage to all elements makes Dublin Core based description more expressive.

2. INCONSISTENCY PROBLEMS OF CURRENT SOLUTIONS

Generally, in order to characterize a resource, a set of statements can be used, each consists of a property name and a value. The Dublin Core Metadata Initiative defined a simple system that could be accepted by most repository owners for metadata exchange. In Dublin Core based schemas property names are represented by a set of Elements and Element Refinements (formerly called qualifiers), property values can be simple character strings or can be controlled by Encoding Schemes.

Elements and Element Refinements always used to express semantics, the properties of the resource. The Encoding Schemes have a more complex role. They are essentially used for syntax description, but sometimes they may have semantic role either. Although

all terms are defined in a single namespace, it is worth to discuss the two basic types separately.

2.1. The First Interpretation of the Qualified DC: Qualifiers as Attributes

If we would like a more sophisticated description we need a structure, a hierarchy of properties. Adding new elements to the basic 15 items can be easily done in a technical sense, but if the new element is not widely accepted, interoperability remains restricted. Better solution is to retain basic elements and use terms that narrows the meaning of the related element. How it is possible technically?

The first attempt was to qualify DC elements with attributes. Syntactically it can be represented by an attribute called 'qualifier' within the XML tag.

```
<dc:creator qualifier="director">John Smith</dc:creator>
```

The advantage of this method is, that the record remains human readable, and the relation between the element and the qualifier appears in the metadata record itself. Using the qualifier attributes the application of the "dumb-down" principle (i.e. ignoring the qualifiers not known by the client) is easy. However, in this case the qualifiers are built in the schema and can not be changed separately. It is especially confusing in the world of audiovisual documents, since the number of contributor or creator roles is more then 300 (according to the European Broadcasting Union's list), which is changing rapidly and permanently. So there are qualifiers that change independently of the remaining part of the schema. The changes are usually initiated by different organisations than the board responsible for the schema itself. Another disadvantage of such a method is, that a repository owner with special needs either has to build his own schema (define, publish elements and harmonize them with standards etc.), or he has to use an already accepted, but not customized schema. That's why, the qualifier as an attribute is not recommended by the DC community any more.

2.2. The New Interpretation of the Qualified DC: DCMI Metadata Terms

The new recommendation is the set of the DCMI Metadata Terms, which consists of a well defined, standardized set of the following types of terms:

- original DC Elements (like Title, Creator, Relation etc.),
- other Elements (like Audience).
- Element Refinements (like Abstract, dateAccepted etc.),
- Encoding Schemes (like ISO3166, TGN etc.),
- Vocabulary Terms (like Image, MovingImage, Text, etc.).

Using this set anyone can assemble an Application Profile by selecting proper items. The elements, the commonly used element refinements, and the other types of terms are collected in a common namespace, so selecting items from this controlled list ensures, that all terms are defined and understandable for all partners in a unified way. Since both elements and element refinements form a single namespace, the relation between them can

be defined at the namespace level, and need not appear in the metadata exchange records. The element refinement term inherits all properties from the element that it refines.

> "However, since Element Refinements are properties of a resource (like Elements), Element Refinements can alternatively be used in metadata records independently of the properties they refine. In DCMI practice, an Element Refinements refines just one parent DCMI property."[11]

For example, the element refinement 'DirectorOfPhotography' inherits all the properties of the 'Creator' element, but the 'DirectorOfPhotography' term has a narrower meaning then the 'Creator' term, so the former can be considered as a generic subordinate of the latter. The usage of the two terms is identical, the difference between them reveals only in the relation of the terms described in the namespace. It is an important shift comparing to the original qualified DC interpretation, when the qualifier of an element had different role in the DC based description. The difference between the two interpretations is shown by the changed naming convention:

> "A shift from the former view to the latter is reflected in the names assigned by the Usage Board to Element Refinements, with a move away from adjective-like names such as 'created' (approved in July 2000) towards noun-phrase-like names such as 'dateCopyrighted' (approved in July 2002)."[11]

The consequence of this approach is that an element refinement can have just one parent: the 'DirectorOfPhotography' element refinement can refine only the 'Creator' element, but it can not be used to refine the 'Contributor' element, or conversely. It means that the implicit organizing principle among the DCMI terms is the so called mono-hierarchy (this term is borrowed from librarian heritage).

However this simple solution can cause some confusion. If we have an element refinement term (for example the 'DirectorOfPhotography'), this term can be subordinated to the 'Creator' element if we would like to describe a photo as a resource, but the same element refinement term should belong to the 'Contributor' element in the case of a bibliographic metadata schema. The same term (with the same meaning) can have two superordinated terms. If one would use one general schema for these two cases, the element refinements can have two (or more) parents i.e. the refinement relation is poly-hierarchical. However, this option is unacceptable if we would like to maintain the validity of the dump-down principle, because in this case we could not know which parent elements ('Creator' or 'Contributor') should be used if the element refinement term ('DirectorOfPhotography') can not be parsed.

It has to be notified, that the basically two-level hierarchy in DCMI terms can be easily expanded to more levels. The DCMI element refinement term has a 'Refines' property that defines the generic relation to the parent element. There is no any practical example among the DCMI Terms, but theoretically it is conceivable, that an element refinement term refines another element refinement term. In this case we can have three or maybe more levels of a hierarchical structure.

Using DCMI terms applications made for the handling of pure Dublin Core metadata can be adapted easily, because the exchange format is unchanged (e.g. eprints[14]).

Comparing the two approach:

Property	DC Qualifiers	DCMI Terms
Adding new refinement element	The whole schema has to be changed (versioned)	Only the new element refinement has to be defined and added
Metadata record readability	The two-level hierarchy appears in the record itself	The hierarchy appears only in the namespace
Element refinement restrictions	Element refinement with the same name can refine multiple elements	Element refinements has to be unique for all the schema and restricts only one element
Special requirements	New schema has to be created for different document types	The application profile for different resource types can be assembled using the DCMI terms
Expanding hierarchy	Requires well defined use of attributes	Easy, but full understanding of metadata exchange records assumes the knowledge of the application profile
Using "dumb-down" principle	Easy (only the exchange record is required)	Requires information from the DCMI term namespace
Application support	Special application has to be developed for qualified search	Applications developed for pure DC can be used

2.3. The Two Basic Types of Encoding Schemes

The next important DCMI metadata term types are the encoding schemes which are used to control the values of the properties of the resources. An encoding scheme can define the parsing rules, the interpretation of the value, or even the semantics. Encoding schemes usually appear as the value of the Type attribute in the original qualified DC schemas. The DCMI defines only two types of encoding schemes "officially" (the first two items of the following list); practically, however, there are three:

- Vocabulary encoding schemes,
- Syntax encoding schemes,
- Semantically structured values with optional syntax restrictions.

Looking at the list above, it can be seen that the last version promises the most flexible description (and the most complicated as well).

'Vocabulary encoding schemes' are used to indicate the simple fact, that the property value is derived from a controlled vocabulary. Using terms from the same origin is essential to ensure integration and interoperability (e.g. the Library of Congress or Dewey Decimal Classification). Referencing to another scheme (beyond DCMI terms), however, makes the development of the collection management and search application more complicated. Other vocabulary encoding schemes satisfying special requirements can be easily added.

(It is worth mentioning, that one of the DCMI vocabulary encoding schemes is principally different from the others. The Getty Thesaurus of Geographic Names is a system of proper names, while all others are composed of common nouns.)

The basic 'Syntax encoding schemes' can help metadata harvesting clients to interpret and validate values. For example, the W3C-DTF scheme defines date formats, supported ISO and RFC specification the country codes, or URI syntax.

2.4. Vocabulary Terms

Among the DCMI terms there are some vocabulary terms, which are all elements of the DCMIType encoding scheme. These terms can be the values of the 'Type' DC element, and this simple fact clearly shows that these terms have different role in the DC description: a vocabulary term can be the value of a property (an element or an element refinement). The class of the elements of the DCMIType encoding scheme is the only one example among the DCMI Terms for the 'Vocabulary Terms', but the functionalities of the DCMIType encoding scheme and the other vocabulary encoding schemes (like LCSH, TGN) are the same: providing uniform semantics for all terms what we would like to describe the values of the DC properties with. From this point of view we can say, that semantically vocabulary terms have the similar role as 'Element refinements' (to provide more exactly, more refined description), however their grammatical role differs.

Although 'Vocabulary Terms' appear as identical parts of the term system, their functionality significantly differs from the 'Element' and the 'Element Refinement' terms. The narrowing of semantics in this case is obtained by restricting the values of a property for a controlled list. All vocabulary terms currently registered among DCMI terms are related exclusively to the 'Type' element, however all controlled lists could be treated this way. The DCMI give preference to the 'Type' element over other elements (e.g. 'Subject'). The Vocabulary terms fit into a semantic hierarchy (using the 'narrower' and 'broader' properties of the terms), and can be expanded easily if required. For example we can say, that the 'type' ('Element') of a resource is an 'Image' ('Vocabulary term'), or a 'Moving Image' ('Vocabulary term' with narrower semantics as 'Image'), or a 'Silent Moving Image' ('Vocabulary term' with narrower semantics as 'Moving Image'). The latter term is not included into the DCMIType encoding scheme, but theoretically we can easily add it to this class.

2.5. The Third Type of Encoding Scheme

In a previous section we already mentioned, that in spite of the belief of the DC community we can differentiate three types of encoding schemes. The interesting exceptions among encoding schemes are the Box and Period terms which are intended to specify the syntax of the Coverage and the Date Elements, and the Spatial and Temporal Element Refinements. Conventional syntax encoding schemes are restricted to formal syntax, thus no more information is supplied about the document. These exceptions however provide more semantics as well.

Let us see the Period term![10] Using the structured element value, it is possible to determine a period without definite Start or End point, or a period with uncertain bounds characterized by a name (e.g. 'Middle ages').

```
<dcterms:date xsi:type="dcterms: Period"> Start=1991; End=2000;
Scheme="W3CDTF"</dcterms:date>
```

This – DCMI supported – encoding scheme is called DCSV (Dublin Core Structured Value) scheme[8] where punctuation characters are used in the element value to describe a structured value as follows:

- equals-signs (=) separate plain-text labels of structured value-components from the values themselves,
- semi-colons (;) separate (optionally labelled) value-components within a list,
- dots (.) indicate hierarchical structure in labels, if required.

If we use the Period encoding scheme to describe the value of the Date property of a resource (as in the above example), we can know more about the resource. We could express the same knowledge if we would define two new Element Refinements (dateStart and dateEnd) within a NDA namespace recording the following two DC-sentences:

```
<NDA:dateStart> 1991 </NDA:dateStart>
<NDA:dateEnd> 2000 </NDA:dateEnd>
```

These two DC-records and the previous record expressed with the help of the Period Encoding Scheme are semantically identical. The syntax of the Period term allows us to refine the Date property of the resource within the structure of the value of this property. It means that the Period term has not only a syntactic controlling power, but it has a semantic refinement ability as well.

The DCMI-DCSV recommendation intends to define a compact human-readable data-structuring method avoiding certain punctuation characters which can cause difficulties in some encoding environments. The DCMI-DCSV notation should normally be used only when no other suitable scheme is available. Note that some other more simple methods exist i.e. the comma-separated value (CSV) and tab separated value (TSV) schemes but these schemes are not human interpretable since the order of the values is fixed but not indicated textually. The XML provides a general solution using tags contained within angle brackets (<, >) to indicate the structure hence this approach covers the DCMI-DCSV but the human-readability is worse.

The speciality the DCMI-DCSV scheme is that it has a semantic message besides syntax definition as well. Note, that a syntax encoding form is embedded in the data structure. This encoding formalism is not familiar with the original DC concepts. However it is accepted, because closely linked semantic terms can not be expressed using unstructured values.

3. COMPLEX ENCODING SCHEMES FOR CONSISTENT DESCRIPTION

The XML attributes are commonly used to provide information that is not relevant to the value i.e. the used encoding scheme, the unique identifier within a global system etc. We show that the XML attribute is only slightly capable of element refinements. To demonstrate this we show the possibilities of the refinement encoding on a simple example.

Let us imagine that we have to describe a Contributor property of a resource (its value let it be ‘Rex Gleam’) and we would like to refine the Contributor element itself, i.e. we would like to express Rex Gleam’s role subordinated to Contributor (let it be ‘lighting

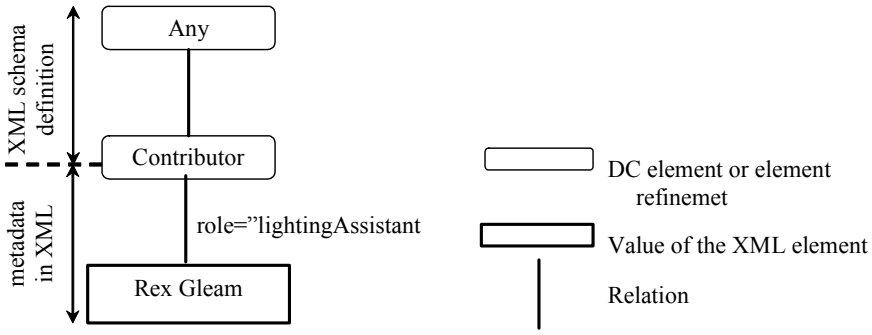

Figure 1. Refined description of a property by using XML attributes

assistant’). Assume that one has defined a ‘qualifier’ attribute for all DC elements within the NDA namespace. By using this attribute to indicate the ‘role’ refinement we have to introduce one more attribute to describe the content of the role since an attribute can not contain structure or multiple values (see Figure 1.):

```
<NDA:contributor NDA:qualifier=”role”
NDA:role=”lightingAssistant”>Rex Gleam</NDA:contributor>
```

Alternatively, we get the same result by avoiding the qualifier attribute:

```
<NDA:contributor NDA:role=”lightingAssistant”>
Rex Gleam</NDA:contributor>
```

On the other hand, the original schema definition files of the dublincore.org (dc.xsd and dcterms.xsd) do not enable us to extend the schema of the DC elements with attributes not specified by them hence only the common XML attributes are useable i.e. xsi:type or xml:lang attributes. Hence the example above assumes that the original DC XML namespace is overridden with a new XML schema. The name ‘lightingAssistant’ could be a name in a namespace, but the attribute value validation of this is difficult in XML.

When we avoid using the attributes, we have two other solutions to encode the person and his/her role:

- *new term approach*: a new term for each role is introduced and the value of the term describes the person,
- *structured value (or mixed encoding scheme) approach*: a structure within the XML element value is used where both the person and the role is described (see Figure 2.)

An example for the first approach could be any *dcterm* element, but to solve our current problem the result of the first approach is the following:

```
<NDA:lightingAssistantContributor>Rex Gleam
</NDA:lightingAssistantContributor>
```

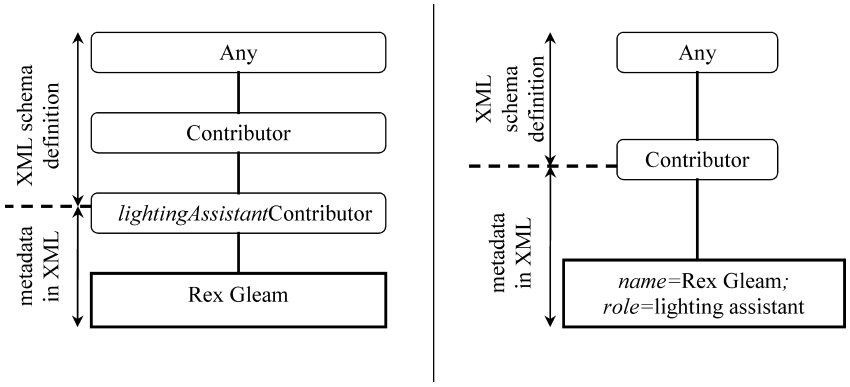

Figure 2. Comparing refinement using a new DC Term (left) and using structured value (right).

The second approach is based on a new XML element data type similarly to the Period encoding scheme. The DCMI-DCSV version is the following:

```
<NDA:Contributor xsi:type="NDA:personDCSVType">
name=Rex Gleam; role=lightingAssistant </NDA:Contributor>
```

while an XML friendly version of the above DCMI-DCSV scheme can also be used:

```
<NDA:Contributor xsi:type="NDA:personComplexType">
<name>Rex Gleam</name>
<role>lightingAssistant</role>
</NDA:Contributor>
```

Since the original *dc* and *dcterms* XML namespaces do not allow to extend the types within the namespace we have to derive a new Contributor element within a new namespace called NDA. Of course, we can "refine" the Creator element in the same way, and we can handle the problem mentioned earlier, that on the one hand a role, i.e. a photographer, can belong to the Creator element (in the case of a photo schema), while on the other it can refine the Contributor element (within a bibliographic schema). If we would like to resolve this problem with refinement elements, we would be faced with the problem of poly-hierarchy, which is not allowed within the DCMI, but if we use the offered DCSV-like solution, we can semantically refine both the Creator and the Contributor elements (even the Publisher one, also), and we do not harm the dump-down principle as well.

Using the structured value approach the role could be taken from a simple controlled list or from a thesaurus (i.e. from a new Vocabulary Encoding Scheme), and the – semantic – poly-hierarchy is also allowed, similarly to the avoided XML attribute-based approach. Furthermore, the *structured value approach* enables the schema designer to extend the structure with more sub-elements and within the XML based version the 'role' sub-element could be imported from a namespace:

```
<NDA:Contributor xsi:type="NDA:personComplexType">
<name>Rex Gleam</name>
```

```
<agent>Person</agent>
<agentID xsi:type="NDA:agentNameSpace">02x3h54f5f2</agentID>
<role xsi:type="NDA:roleType">lightingAssistant</role>
</NDA:Contributor>
```

The DCMI-DCSV version of the last example is a bit more complicated:

```
<NDA:Contributor xsi:type="NDA:personDCSVType">
name=Rex Gleam;
agent=Person;agentID=02x3h54f5f2; agentNS="NDA:agentNameSpace";
role=lightingAssistant; roleNS="NDA:roleType";
</NDA:Contributor>
```

The XML-based version could be more easily validated with an XML based system and the validation of the DCMI-DCSV version could be validated with an application outside the XML domain. Both versions are human-readable, but in this aspect the DCMI-DCSV scheme is the favourable.

Extending the DC refinement formalism to the other elements, the problem of the several possible subtypes (refinements) of the Contributor (Creator and Publisher) element can be handled. Defining a semantic-syntactic structured Role scheme, combined with an embedded RoleType Vocabulary Encoding Scheme containing the list of possible roles, and namespace containing the enumerated type of the proper names of agents, makes possible to accurately express the role-agent relation without leaving the domain of DCMI concepts.

4. CONCLUSIONS

The mission of the Hungarian National Digital Archive (NDA) is the enforcement of the digitization of the documents of cultural heritage. Digitization does not only mean the production of the digital representation of physical objects. The main purpose is to make accessible cultural values, and to achieve as high a level of integration among the different archives as possible. Based on some offered standards and solutions, the metadata of the different archives can be freely harvested which makes simultaneous searches possible.

However, at this point an important problem arises from the differences between the requirements of the collection management of the archives and the searching expectations or intentions of the users, or of the searching service providers. The generalization of the structured DCMI encoding scheme, like the term Period, will make it possible to satisfy both the professional's requirement of sophisticated description and the support of user search as well.

The choice of the best solution depends on the application. First, the solution must conform the DCMI recommendations as closely as possible hence we do not recommend using the XML attributes for DC element refinements. When one does not have to introduce many element refinements, and the mono-hierarchy is held, we recommend introducing them as a direct descendant of the DC elements like the so called *dcterms*. When the new refinements require a poly-hierarchical relation to the DC elements, like the set of the role refinements, we recommend using a child element structure within the appropriate descendant of the DC or *dcterms* element. The structure could be based on either the

DCMI-DCSV approach which can be validated against a syntax rule, or on the XML-based approach where not only the syntax but also the content of a sub-element can be validated. The XML-based version also enables us to use external namespace referencing, controlled lists etc., but it is important to underline that the overall metadata structure should not be too large or complicated since one must create and maintain the metadata and the human-readability must be held as much as possible.

REFERENCES

1. G. Knapp, G. Magyar, G. Németh, Building an Open Document Management System with Components for Trust, (Twelfth International Conference on Information Systems Development, Melbourne, 2003).
2. G. Magyar, G. Knapp, I. Szakadát, Information Systems Development in the e-Age, conference paper (Tenth International Conference on Information Systems Development, London, 2001).
3. Dublin Core Metadata Element Set - Reference Description - Version 1.1, (CEN, 2000).
4. The Dublin Core Metadata Element Set, ANSI/NISO Z39.85-2001, (ANSI, 2001).
5. Information and documentation – The Dublin Core metadata element set, (ISO, 2003); http://www.niso.org/international/SC4/n515.pdf.
6. Dublin Core Metadata Element Set, Version 1.1: Reference Description, (2003); http://dublincore.org/documents/2003/06/02/dces/.
7. Dublin Core Qualifiers, (2000); http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/.
8. EBU Core Metadata Set for Radio Archives, Tech 3293, (European Broadcasting Union, 2001).
9. DCMI Metadata Terms, DCMI Usage Board, (2001); http://dublincore.org/documents/2003/11/19/dcmi-terms
10. DCMI DCSV: A syntax for writing a list of labelled values in a text string; http://dublincore.org/documents/dcmi-dcsv/.
11. DCMI Grammatical Principles, (2003); http://dublincore.org/usage/documents/principles.
12. S. Cox, DCMI Period Encoding Scheme, specification of the limits of a time interval, and methods for encoding this in a text string, (2000); http://dublincore.org/usage/documents/2000/7/11/dcm-period/index.shtml.
13. Powell, M. Nilsson, A. Naeve, P. Johnston, DCMI Abstract Model, (2004); http://www.ukoln.ac.uk/metadata/dcmi/abstract-model.
14. http://www.eprints.org/.