Building an Open Language Archives Community on the DC Foundation

Steven Bird
University of Melbourne
and University of Pennsylvania

Gary Simons
SIL International

DRAFT: September 29, 2003

Abstract

The Open Language Archives Community is an international partnership of institutions and individuals that is creating a worldwide virtual library of language resources. We report on the development of OLAC metadata as a specialization of Dublin Core metadata and then describe the interoperability framework in which the metadata is validated, disseminated and aggregated. We also discuss the community-centered process by which OLAC standards and practices are created and maintained. In each of these three areas, metadata, interoperability, and process, we show how OLAC began with a model that was too cumbersome to implement then found a new formulation which worked in practice. By reporting on this experience of metadata in practice, we hope to show how a specialist community can address its resource discovery needs by building on the Dublin Core foundation.

A revised version of this paper will appear in: Hillmann and Westbrooks (editors) Metadata in Practice: A Work in Progress, ALA Editions, 2004.

1 Introduction

The rapid growth of ubiquitous computing and the multilingual, multimedia internet is stimulating the development of a new generation of language technologies. Speech processing, translation, information extraction and document summarization, among others, unlock the information content of large unstructured collections of text and speech, and open the door to more natural human-computer interfaces. At the same time, inexpensive hardware for digital capture and mass storage is stimulating widespread efforts to digitize and preserve the world's endangered linguistic heritage. All of these efforts depend on language technologies and language data. Yet, as these language resources proliferate it is becoming increasingly difficult to locate and reuse them.

In December 2000, a new initiative based on Dublin Core (DC) and the Open Archives Initiative (OAI, Lagoze and Van de Sompel 2001) was founded, with the following statement of purpose:

> OLAC, the Open Language Archives Community, is an international partnership of institutions and individuals who are creating a worldwide virtual library of language resources by: (i) developing consensus on best current practice for the digital archiving of language resources, and (ii) developing a network of interoperating repositories and services for housing and accessing such resources.

At the time of writing there are 25 participating repositories and several proposed standards and recommendations under review by the OLAC community [`www.language-archives.org`].

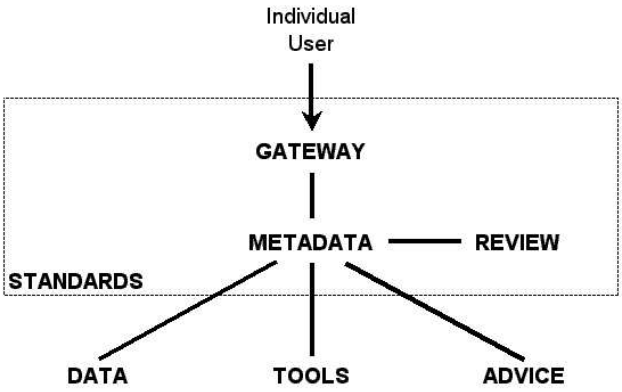

Figure 1: Vision of Community Infrastructure

The framework for OLAC's operation is provided by three standards, OLAC Metadata (Simons and Bird, 2003b), OLAC Repositories (Simons and Bird, 2003e), and OLAC Process (Simons and Bird, 2003c). This paper charts the community-based development of these standards and shows how, in each case, initial models that proved too cumbersome to implement were simplified over time through the adoption of ideas and practices originating in the Dublin Core community.[1]

2 Launching a metadata community

The Open Language Archives Community grew out of a collaboration between three international linguistic service organizations: the Linguistic Data Consortium (LDC), SIL International, and LINGUIST List. The LDC supports language-related education, research and technology development by creating and sharing linguistic resources; to date it has published over 200 linguistic databases and distributed more than 15,000 copies to research institutions worldwide.[2] SIL International serves the peoples of the world through research, translation, and literacy; at present it is facilitating language-based development in over 1000 languages worldwide.[3] The LINGUIST List is the home of linguistics on the web, currently hosting 2,000 pages of content, 100 mailing lists, and serving 20,000 subscribers worldwide.[4]

These three organizations could hardly be more different in their goals and constituencies, yet all three found themselves managing digital language documentation and developing software infrastructure and educational materials. Moreover, all three organizations were independently developing systematic methods to archive, catalog and disseminate these resources, while helping their associated communities to do likewise. Joining forces, the trio of organizations became a microcosm of the language resources community. With timely sponsorship from the US National Science Foundation, the group explored an OAI- and DC-based solution to their needs, making rapid progress on implementation owing to the simplicity and generality of these standards.

[1] For discussion of the details of the OLAC vocabularies, and on how OLAC builds on the OAI foundation, we refer the reader to (Bird and Simons, 2003a) and (Simons and Bird, 2003a).

[2] `www.ldc.upenn.edu`

[3] `www.sil.org`

[4] `www.linguistlist.org`

What users want	What users don't want	How OLAC meets the requirement
There is a single site on the Web where any user can go to discover what language information resources are available, regardless of where they may be archived.	The only way to discover language resources on the Web is to visit all the individual archives or to hope that the resources one is interested in have been indexed in an intuitive way by one's favorite general-purpose search engine.	Linguist List (`www.linguistlist.org`) will host a combined catalog of all participating archives.
All language resources (regardless of where they may be archived) are catalogued with a consistent set of metadata descriptions, so that the user can ascertain all the basic facts about a resource without having to download it.	The only way to get a good idea about what a resource contains, who is responsible for it, or what are its terms of availability is to retrieve it.	Every holding in the combined catalog is described using the OLAC metadata set. Since that metadata set includes all the elements of the Dublin Core, it offers enough breadth to handle all the basic facts about a resource.

Figure 2: Examples of User Requirements

This exploration quickly matured into a high-level vision and detailed low-level requirements (Simons and Bird, 2000a,b). The high-level vision described "seven pillars of open language archiving" and presented the simple model of resource discovery shown in Figure 1. Individual users would be able to access the DATA, TOOLS and ADVICE they needed by visiting a single GATEWAY to access aggregated METADATA. The operation of the system would be governed by a small set of STANDARDS and high-quality content would be encouraged by peer REVIEW within the community.

Corresponding to this high-level vision were 36 low-level requirements covering the needs of five special interest groups: (i) users, the people who want to access language materials which have been stored away in archives; (ii) creators, the people who create the language materials that get archived; (iii) archivists, the people who manage the process of acquiring, maintaining, and accessing the information resources stored in archives; (iv) developers, the people who create data models, tools and formats for storing and manipulating digital language documentation; and (v) sponsors, the organizations that fund the creation of information resources and their maintenance in archives. Each requirement consisted of three statements: the desired state, the situation we want to avoid, and how OLAC would meet the requirement once it was functioning. The first two user requirements are shown in Figure 2.

These two documents – the high-level vision and the low-level requirements – were augmented with a survey of language archives (Bird and Simons, 2000a), a white paper showing how OLAC could be built on DC metadata and the OAI Protocol for Metadata Harvesting (Bird and Simons, 2000b), and a mock-up demonstration of an OLAC service provider. All of these components were presented at the Workshop on Web-Based Language Documentation and Description, held in Philadelphia in December 2000. Present at this meeting was a very broad cross-section of the language resources community representing work in Africa, Asia, Europe, Australia, and North and South America. Over the course of the three-day workshop, including working group sessions and consultations with geographical and domain representatives, a strong consensus was built. In the closing session of the workshop OLAC was formally established. In the following weeks, those with archived resources set about developing OAI-compliant metadata repositories.[5]

The first twelve months was a period of active development of the repositories and supporting infrastructure. The metadata format alone went through a succession of four versions during this time. In the second year, 2002, we froze the format and recruited new archives to join, more than doubling the level of participation. In December 2002, on the second anniversary of OLAC's formation, we revised the format and infrastructure based on the previous year's experience, and in 2003 we are now in a period of refinement and adoption of the core standards upon which OLAC is built.

[5]The technical infrastructure we developed to support this work is reported in (Simons and Bird, 2003a).

Taking OAI and DC off-the-shelf as proven standards having widespread acceptance in the digital libraries community was decisive, permitting OLAC to unite disparate subcommunities and reach consensus. In particular, DC was simple, applicable to all kinds of resources, and widely used outside our community. Significantly, DC represented neutral territory as it was not developed by any special interest within the language resources community. Had we come to our first workshop with the proposal that the community needed to invent a metadata standard, all resolve would have dissipated in factionalism. Thus, not only was DC both simple and mature, it was also a political expedient.

During this period, three standards were developed and adopted. One for the metadata format which extended DC (see §3), one for the functioning of the repositories which extended the OAI-PMH (see §4) and one for governing the organization and operation of the OLAC community (see §5).

3 OLAC Metadata: A standard for resource description

As we saw in Figure 1, language resources can be divided into three broad categories, namely data, tools, and advice. By *data* we mean any information that documents or describes a language, such as a published monograph, a computer data file, or even a shoebox full of hand-written index cards. The information could range in content from unanalyzed sound recordings to fully transcribed and annotated texts to a complete descriptive grammar. By *tools* we mean computational resources that facilitate creating, viewing, querying, or otherwise using language data. Tools include not just software programs, but also the digital resources that the programs depend on, such as fonts, stylesheets, and document type definitions. By *advice* we mean any information about what data sources are reliable, what tools are appropriate in a given situation, what practices to follow when creating new data, and so forth (Bird and Simons, 2003b). In the context of OLAC, the term *language resource* is broadly construed to include all three of these types: data, tools and advice. The purpose of OLAC metadata is to facilitate the discovery of language resources.

Over the past three years, work on OLAC metadata has centered on two key issues: extensions to the Dublin Core Metadata Element Set (DCMI, 1999) to support the description of language resources, and a suitable XML representation of this metadata. In this section we summarize the requirements on OLAC metadata (§3.1), then review our first solution and its problems (§3.2), and then present our OLAC application profile based on the recent DC XML schemas (§3.3).

3.1 Principal requirements for OLAC metadata

While the community has complex resource discovery needs, we adopted DC's minimalist philosophy and identified a small set of widely-used categories and descriptors that could be used to extend DC for application to language resources. The most important of these are subject language, language codes, and linguistic types, discussed below.

Subject Language. This is a language which the content of the resource describes or discusses, as distinct from the language the resource is in. For example, a grammatical description of French written in English would have English as its language, and French as its subject language. The same description would apply to a French text with English annotations. OLAC metadata needs to distinguish language from subject language.

Language Codes. These are a standard set of codes for language identification. The existing ISO 639 vocabulary (ISO, 1998) covers less than 10% of the world's languages, and does not adequately document what languages the codes refer to. The use of conventional language names in resource description is fraught with problems (Constable and Simons, 2000), leading to low precision and recall. However, SIL's Ethnologue (Grimes, 2000) provides identifiers for some 7,000 living and recently extinct languages, while LINGUIST List provides identifiers for some

400 ancient and constructed languages.[6] In order to meet the need for precise identification of language or subject language on any language resource, OLAC employs the unambiguous ISO codes, augmented with Ethnologue and LINGUIST codes.

Linguistic Types. These are a standard set of codes for classifying the content of a resource according to recognized structural types of linguistic information (e.g. dictionary, grammar, text). This permits members of the community to identify resource types according to the most fundamental linguistic categories.

In addition to these requirements on OLAC metadata, there are three basic requirements on OLAC metadata management – migration, evolution, and extensibility:

Migration. OLAC metadata originates in existing institutional and individual repositories, and there are existing guidelines and examples for exporting this metadata to Dublin Core. To facilitate migration to OLAC metadata, we must specify all OLAC refinements and encoding schemes as optional. Thus, a DC record is a valid OLAC record, and a repository can enrich its exported records by progressively replacing free text with coded values (e.g. *Spanish* $\rightarrow$ es) and selecting suitable refinements (e.g. Subject $\rightarrow$ subject.language). A related requirement concerns the ability to dumb-down to DC for interoperability with the wider digital libraries community.

Evolution. Once an encoding scheme has been adopted, subsequent changes must be carefully controlled. Redefining a coded value to mean something different would cause problems for users and repositories that employ the existing coded value. In particular, when the interpretation of a coded value is narrowed in scope, the existing code must be expired and a new code adopted.

Extensibility. Subcommunities with specialized resource discovery needs should be able to extend OLAC metadata with their own refinements and encoding schemes, and build services based on the enriched metadata.

3.2 OLAC 0.4 Metadata: Proliferating vocabularies

In its first six months of development, OLAC metadata went through versions 0.1-0.4, the last of which was in active use for about 18 months. Version 0.4 consisted of the 15 DC elements plus 8 community-specific elements (Simons and Bird, 2001). Each of the latter was a refinement to an existing DC element and supplied a vocabulary for encoding its values. Two additional community-specific vocabularies were defined, one for specifying role as a refinement of Creator and Contributor, and one for encoding values of Rights. The ten community-specific vocabularies (many of which were never developed) are listed in Table 1.

The draft OLAC metadata standard (Simons and Bird, 2001) provided additional comments on these elements to describe usage, e.g. for Format.markup:

> "For a resource that is a text file including markup, Format.markup identifies the markup system it uses, such as the SGML DTD, the XML Schema, the set of Standard Format markers, and the like. For a resource that is a stylesheet or a software application, Format.markup names a markup scheme that it can read as input or write as output. Service providers will use this information to match data files with the software tools that can be applied to them. Recommended best practice is to identify the markup scheme by a URI giving an OAI identifier for the markup scheme as a resource in an OLAC archive. Thus, if the DTD, Schema, or markup documentation is not already archived in an OLAC repository, the depositor of a marked-up resource should also deposit the documentation for the markup scheme. A resource identified in Format.markup should not also be listed with the *requires* refinement of Relation."

Several refinements and encoding schemes (cpu, os, sourcecode) were primarily for describing linguistic software. Some were easy to define but none were heavily used, there being only two small software repositories holding

[6] http://linguistlist.org/ancientlgs.html, http://linguistlist.org/constructedlgs.html

Element	Qualifier	Definition
Creator/Contributor	Role	The role played by the creator or contributor in the creation of the resource (author, editor, translator, ...)
Format.cpu	CPU Requirement	The CPU required to use a software resource (x86, mips, alpha, ppc, sparc, 680x0)
Format.encoding	Character Encoding	An encoded character set used by a digital resource (vocabulary never defined)
Format.markup	Markup Scheme	A markup scheme used by a digital resource (vocabulary never defined)
Format.os	OS Requirement	An operating system required to use a software resource (Unix/Linux, Unix/Solaris, OS2, MacOS/OSX, MSWindows/win95, ...)
Format.sourcecode	Source Code Language	A programming language of software distributed in source form (C, Java, Python, Tcl, ...)
Rights	Rights Management	Information about rights held in and over the resource (vocabulary never defined)
Subject.language	Subject Language	A language which the content of the resource describes or discusses (aa, ab, ae, af, am, ar, ..., x-sil-AAA, x-sil-AAB, ...)
Type.functionality	Software Functionality	The functionality of a software resource (vocabulary never defined)
Type.linguistic	Linguistic Data Type	The nature or genre of the content of the resource from a linguistic standpoint (transcription/phonetic, lexicon/thesaurus, text/dialogue, ...)

Table 1: OLAC Qualifiers (Version 0.4)

less than 1% of the total number of harvested OLAC records. Others (encoding, markup, functionality, rights) were never employed as qualifiers, or else used in a completely unconstrained manner, providing inadequate data on which to base an encoding scheme. Once a good extension mechanism had been found (in version 1.0) these were all dropped, leaving only our basic qualifiers (subject language, language codes, and linguistic type) and a small number of others (e.g. role).

Language codes were derived from external authorities and required no particular attention in the context of OLAC. Linguistic type, on the other hand, proved extremely difficult to manage. It mushroomed to a 70-item vocabulary with two-levels of detail separated by a slash (e.g. transcription/phonetic). However, the two-level organization was unstable: terms that were introduced to improve coverage led to subtle problems of demarcation; the details of the two-level structure could not be finalized; and the terms were found to be of two cross-cutting domains (some describing structure, others describing content). The final resolution, reached in version 1.0, consisted of just three terms (Johnson and Dry, 2002).

OLAC 0.4 metadata was formalized both as a proposed standard (the human readable document), and as an XML schema (the machine readable document).[7] The XML representation expressed refinements using dotted element names (e.g. subject.language), and expressed coded values using a code attribute. XML Schema was used to validate the content of the code attribute, according to the name of the host element. Element content was left unconstrained, either for holding free-text descriptions (in the case when no coded value is provided), or for holding free-text elaborations (in the case when the coded value is insufficient on its own). For example <subject.language>Spanish</subject.language> expresses a refinement without an encoding scheme, and is an intermediate step on the way to an encoded value: <subject.language code="es"/>. A language code is

[7]The OLAC 0.4 metadata standard is available as (Simons and Bird, 2001), while the corresponding schema is available at `http://www.language-archives.org/OLAC/0.4/olac.xsd`.

inadequate for identifying dialects, so free-text content can be used to provide the necessary elaboration, e.g.: <subject.language code="es">Andalusian</subject.language>. A second attribute permitted third-parties to represent encoding schemes involving element content: the scheme attribute held the name of the scheme, and the content was assumed to be constrained accordingly. A third attribute, refine, was used for elements having vocabulary of refinements (Contributor, Creator, Date, Relation).

OLAC 0.4 metadata satisfied the requirements listed in §3.1 for representing the subject language refinement, language codes, and linguistic types (and a selection of other qualifiers), and it satisfied the migration requirement. However, the evolution and extensibility requirements were not well supported. Vocabulary evolution had unacceptable bureaucratic overheads, as each vocabulary revision entailed a new release of the XML schema and the metadata standard. Extensibility was not well supported, since the XML Schema language is unable to constrain element content based on the value of the scheme attribute.

In sum, the OLAC 0.4 metadata proved too difficult to manage over the long term. Administratively, it encouraged us to seek premature closure on issues of content description that can never be closed. Technically, it forced us to release new versions of the metadata format with each vocabulary revision, and forced us to create software infrastructure to support an unwieldy conglomeration of four syntactic extensions of simple Dublin Core:

```
<element.EXT1 refine="EXT2" code="EXT3" scheme="EXT4">
```

After a year of using this model we realized it was untenable, and discovered new DCMI work on the XML representation of DC and DC qualifiers (Powell and Johnston, 2003; Johnston et al., 2003). This provided the missing support for vocabulary evolution and extensibility that OLAC 0.4 was lacking. At this point OLAC metadata was a *semantic* extension of DC metadata, but syntactically unrelated to DC. With the arrival of a standard XML representation for DC it was now possible to reconceive OLAC metadata as also being a *syntactic* extension of DC.

3.3 OLAC 1.0 Metadata: An application profile for language resource description

An "application profile" is a hybrid metadata record that combines elements and attributes from multiple authorities (Heery and Patel, 2000). We can view OLAC metadata as an application profile for the language resources community, combining DC elements with a small selection of community-specific qualifiers. With the new XML representation of Dublin Core, it is straightforward to implement this model in XML Schema.[8]

Consider the XML representation for the following DC subject element: <subject>Spanish</subject>. We can specify the language refinement and encoding scheme using the special xsi:type attribute, thus: <subject xsi:type="olac:language" code="es"/>. The xsi:type attribute is defined in the XML Schema standard; it directs the schema validator to override the definition of the XML element using the specified type (here, olac:language). OLAC-defined types add an optional code attribute and restrict its range of values to a specified vocabulary. Element content is reserved for unrestricted commentary. Thus, the following are all acceptable OLAC 1.0 elements:

```
<subject>Spanish</subject>
<subject xsi:type="olac:language">Spanish</subject>
<subject xsi:type="olac:language" code="es"/>
<subject xsi:type="olac:language" code="es">Andalusian</subject>
```

The above example also illustrates the migration path from simple DC to OLAC metadata. Dumb-down to DC is straightforward (though OLAC provides this service centrally to ensure that best practices are followed, see §4).

[8] Please see `http://www.language-archives.org/OLAC/1.0/olac.xsd` for details.

```
<?xml version="1.0" encoding="UTF-8"?>
<olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
           xmlns="http://purl.org/dc/elements/1.1/"
           xmlns:dcterms="http://purl.org/dc/terms/"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.language-archives.org/OLAC/1.0/
                http://www.language-archives.org/OLAC/1.0/olac.xsd">

  <title xml:lang="fr">Petit Dictionnaire Yémba-Français</title>
  <dcterms:alternative>Yemba-French Dictionary</dcterms:alternative>
  <date xsi:type="dcterms:W3CDTF">1997</date>
  <identifier xsi:type="dcterms:URI">http://www.ldc.upenn.edu/sb/home/publications.html#dic

<!-- Elements with OLAC extensions -->

  <subject xsi:type="olac:linguistic-field" olac:code="morphology"/>
  <creator xsi:type="olac:role" olac:code="editor">Bird, Steven</creator>
  <creator xsi:type="olac:role" olac:code="editor">Tadadjeu, Maurice</creator>
  <language xsi:type="olac:language" olac:code="x-sil-BAN">Dschang</language>
  <type xsi:type="olac:linguistic-type" olac:code="lexicon"/>
  <type xsi:type="olac:linguistic-type" olac:code="language_description"/>

</olac:olac>
```

Figure 3: Example of an OLAC 1.0 Metadata Record

In addition to language identification and linguistic type, OLAC 1.0 metadata currently provides three other extensions: discourse type, linguistic field, and participant roles. Each extension is accompanied with human-readable documentation that provides the semantics for the vocabulary (Aristar Dry and Appleby, 2003; Aristar Dry and Johnson, 2002; Johnson, 2003; Johnson and Aristar Dry, 2002). Additionally, extensions provide summary documentation using the OLAC Extension schema.[9] This summary documentation provides six pieces of information: (i) the short name by which the extension is accessed (i.e. the name of the complexType that defines the extension); (ii) the full name of the extension for use as a title in documentation; (iii) the date of the latest version of the extension; (iv) a summary description of what the extension is used for; (v) the Dublin Core elements with which the extension may be used; and (vi) the URI for a complete document that defines and exemplifies the extension. This information is extracted and displayed in human-readable form on the OLAC website (Simons and Bird, 2003d). A complete OLAC record is shown in Figure 3. It conforms with the OLAC schema,[10] which imports DCMI schemas.

Evolution. In OLAC 1.0 metadata, the format of the metadata container is identified as a standard (Simons and Bird, 2003b), while the metadata extensions have the status of recommendations (Simons and Bird, 2003d). The evolution of individual extensions, and of the metadata format itself, are now decoupled. This was an important step in enabling OLAC metadata to reach version 1.0. It liberated vocabulary editors to continue developing the vocabularies that define OLAC as a community, without forcing a premature closure timed with the 1.0 release of the new container format.

Extensibility. An OLAC metadata record may use extensions from other namespaces. This makes it possible for subcommunities within OLAC to develop and share metadata extensions that are specific to a common special

[9] http://www.language-archives.org/OLAC/1.0/olac-extension.xsd

[10] http://www.language-archives.org/OLAC/1.0/olac.xsd

```
<complexType name="role">
  <complexContent mixed="true">
    <extension base="dc:SimpleLiteral">
      <attribute name="code" use="required">
        <simpleType>
          <restriction base="string">
            <enumeration value="calligrapher"/>
            <enumeration value="censor"/>
            <enumeration value="commentator"/>
            <enumeration value="corrector"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

Figure 4: A Example Type Declaration for a Third-Party Extension

interest. By using xsi:type, it is possible to extend the OLAC application profile without modifying the OLAC schema. For instance, suppose that a given domain required greater precision in identifying the roles of contributors than is possible with the OLAC Role vocabulary (Johnson, 2003), and defined additional role terms including commentator. and included a term commentator. If the extension were named example:role, this new term would be used as follows:

```
<contributor xsi:type="example:role" code="commentator">Sampson, Geoffrey</contributor>
```

In order to do this, an organization representing that domain (say, `example.org`) could define a new XML schema providing the complexType declaration shown in Figure 4. The extension schema is associated with a target namespace (e.g. `http://www.example.org/role/`) and stored on the organization's web site.[11]

Remaining shortcomings of this approach. OLAC 1.0 metadata is implemented in XML Schema following DCMI guidelines (Powell and Johnston, 2003; Johnston et al., 2003). This has two unfortunate consequences. First, thanks to the use of the xsi:type attribute, the XML representation of OLAC metadata is now tied to XML Schema, which is just one of many available methods for validating XML. Given the volatility of XML technologies it seems undesirable to complicate the metadata representation by embedding special directives to be interpreted by the validation technology. At such time as the validation technology is changed, every repository that uses the format will have to modify its XML representation of the metadata. The second shortcoming is that the xsi:type declarations are not constrained as to which DC element they modify. For instance, it would not be a validation error for a metadata record to use the role extension on the title element, even though this violates the intended semantics. Despite these problems, we believe that the significant benefits of conforming to DCMI guidelines outweigh the disadvantages.

[11] Please see (Simons and Bird, 2003e) for an example of a complete OLAC extension definition.

4 OLAC Repositories: A framework for interoperation

OLAC Repositories (Simons and Bird, 2003e) is the second of the three standards that govern the operation of OLAC. In order to put metadata into practice, it is not enough to simply define a standard for expressing metadata descriptions. There must also be an infrastructure that supports the interoperation of metadata across the community in order to meet its resource discovery needs. OLAC has built that infrastructure on the Open Archives Initiative protocol for metadata harvesting (Lagoze et al., 2002). The OAI protocol and its application are well documented in a volume edited by Cole (2003); we have described its application to OLAC in that volume (Simons and Bird, 2003a) and elsewhere (Simons and Bird, 2003f). Rather than repeating a description of the harvesting protocol here, we focus on practical steps that were taken in developing the OLAC infrastructure to make it easier for would-be participants to interoperate within the community.

In the OAI approach to interoperation, the institutions that want to make their resources known participate as data providers by publishing metadata about their holdings. Other institutions that want to provide value-added services for the community participate as service providers by harvesting the metadata and adding it to the information pool that their service is based on.

The OLAC Repositories standard specifies how a would-be data provider must construct a repository of metadata descriptions so as to make it harvestable by service providers. Two approaches are described; a data provider may construct either a dynamic repository or a static repository. With a dynamic repository, the data provider implements a CGI interface that dynamically queries a database. This is the standard OAI approach and was the only method available when OLAC was launched in December 2000. This is a straightforward task for a programmer who knows how to build dynamic web sites; thus we had half a dozen repositories interoperating within a matter of weeks. However, the number of participants did not continue to grow and we soon concluded that programming a dynamic repository went beyond the technical capacity of most potential providers of language resources.

In response we developed Vida, the Virtual Data Provider.[12] It was a service hosted by OLAC that allowed would-be data providers to submit the metadata description of their archive and its resources as a static XML document. Vida then provided the CGI interface that allowed service providers to harvest the metadata from these XML documents. When this service became available, we experienced a dramatic increase in the number of data providers within a matter of months. This is because most potential OLAC data providers found it much easier to map their existing metadata catalogs into a static XML document than to implement a dynamic interface to a database. After seeing the success of Vida for OLAC, the OAI generalized the idea by publishing specifications for a "static repository," and implemented a "static repository gateway" that provides the harvesting interface for a set of static repositories (Van de Sompel and Lagoze, 2002). The OLAC Repositories standard now includes the option of submitting OLAC metadata records as a static repository that is registered with OLAC's static repository gateway.

OLAC has taken one more step to make it even easier for metadata to be put into practice by the language resources community. Learning the XML technologies needed to create a static repository still poses an obstacle for many small projects and individuals who would like to publish metadata describing their work. To meet the needs of this portion of the community, LDC developed the OLAC Repository Editor (ORE), and it is now hosted on the LINGUIST List site.[13] ORE is a forms-based metadata editor that any potential contributor may run from a web browser; it creates and registers a static repository.

It is not enough just to publish metadata. To complete the circle of interoperability, it must also be easy for would-be service providers to harvest the information and offer value-added services. While using the OAI protocol to harvest metadata from all OLAC data providers is straightforward for an experienced programmer, it is beyond the reach of many potential OLAC service providers. Thus, OLAC has implemented two services to make this easier. The

[12] `http://www.language-archives.org/vida/`
[13] `http://www.linguistlist.org/ore/`

first of these is the OLAC Aggregator,[14] a derivative of the OAI Aggregator.[15] OLACA is a service that harvests all the metadata records from all registered data providers and republishes them through the OAI protocol. Thus a would-be service provider can harvest all OLAC metadata records from a single source. Even more significant in simplifying the task of being a service provider is Viser, the Virtual Service Provider.[16] It takes advantage of a query facility that is built into OLACA to make it possible for any web site to dynamically display a page showing relevant OLAC metadata records by simply creating a link to the Viser URL that contains the appropriate query within its parameters.

5 OLAC Process: A method for developing consensus

OLAC Process, the third in the trio of OLAC standards, describes the purpose and vision of OLAC and the four core values which guide OLAC's operation: openness, consensus, empowering the players, and peer review (Simons and Bird, 2003c). It is through documents that OLAC defines itself and the practices that it promotes; thus a key aspect of the OLAC process is how documents are developed and promulgated. The process document sets out an organizational structure consisting of six categories of participants: the Coordinators, an Advisory Board, the Council, Archives and Services, Working Groups, and Participating Individuals. The process document specifies three types of documents (Standard, Recommendation and Note) along with a detailed document process involving six levels: Draft, Proposed, Candidate, Adopted, Retired, and Withdrawn. The process document also defines a process for the working groups that are responsible for creating documents and taking them through their lifecycle. Finally, there is a registration process concerning OLAC Archives and Services.[17]

Versions of the process document prior to December 2002 incorporated community-wide voting as part of advancing standards and recommendations through the document process. After two years of experimentation with the process, it was clear that this aspect was too cumbersome. At the OLAC workshop in December 2002, Diane Hillmann presented the model of the DCMI Usage Board and described its operation (Hillmann and Sutton, 2003). Workshop participants discussed this new model, and agreed that a new OLAC "Council" would be created to replace the voting process, and that this would streamline the document process. At the time of writing, the initial council members have been nominated by the coordinators and approved by the advisory board, and we are moving forward quickly with the promotion of several OLAC documents from proposed to candidate status, and from candidate to adopted status.

6 Conclusion: OLAC metadata in practice

The future of language technology and empirical linguistics depend on the ability to create and re-use a rich array of language resources, including data, tools, and advice. Until recently there has been no systematic way for members of the language resources community to describe the resources they have created or to discover the resources they need. Over the past three years, the Open Language Archives Community has built consensus around a community-specific metadata set. Building on the DC foundation, OLAC has adopted the elements and qualifiers of Dublin Core and identified a small set of language-related extensions, including subject language, language identification, and linguistic type. These apply across the whole field, and have now gained widespread adoption.

[14] `http://www.language-archives.org/cgi-bin/olaca.pl`

[15] `http://oai-perl.sourceforge.net/`

[16] `http://www.language-archives.org/viser/`

[17] In developing the OLAC Process document we have incorporated many ideas from the DCMI Process documents (Iannella and Heery, 1999; DCMI, 1999; Hillmann and Sutton, 2003).

Over this period OLAC has learned from its own experience in three significant ways. The difficulty in finalizing vocabularies and in supporting extensions led to the adoption of a more flexible and open-ended model based on the recently established DC XML format. The technical challenges of becoming a data provider or a service provider led to the development of services that have made it significantly easier for potential participants to interoperate within the community. The unworkability of the community voting process led to the establishment of the OLAC Council, modelled on the DC Usage Board. In each of these three areas, OLAC began with a model that was too cumbersome in practice, then found a new formulation which worked in practice.

The decision to build on the DC foundation has been critical to the acceptance of OLAC metadata. DC was simple, well established, and widely accepted. Different factions of the community were not pitted against one another to argue for their own approach. Instead, we united around the external standard, and got the basic infrastructure up and running within a matter of weeks. Moreover, DC demonstrated the value of a minimalist approach. Linguists are known for their preoccupation with faithful data modelling, and would never have invented a metadata format with a flat structure in which all elements were optional and repeatable. However, this minimalist approach has proven to be a key factor in achieving stability, scalability, and acceptability. We hope that other specialist communities contemplating development of digital archives infrastructure will benefit from the OLAC experience reported here.

References

Aristar Dry, H. and Appleby, M. (2003). OLAC linguistic subject vocabulary. `http://www.language-archives.org/REC/field.html`.

Aristar Dry, H. and Johnson, H. (2002). OLAC linguistic data type vocabulary. `http://www.language-archives.org/REC/type.html`.

Bird, S. and Simons, G. (2000a). A survey of the state of the art in digital language documentation and description. `http://www.language-archives.org/docs/survey.html`.

Bird, S. and Simons, G. (2000b). White paper on establishing an infrastructure for open language archiving. `http://www.language-archives.org/docs/white-paper.html`.

Bird, S. and Simons, G. (2003a). Extending Dublin Core metadata to support the description and discovery of language resources. *Computers and the Humanities*, 37. to appear.

Bird, S. and Simons, G. (2003b). Seven dimensions of portability for language documentation and description. *Language*, 79:557–82.

Cole, T. W., editor (2003). *Library Hi Tech: Special issue on the Open Archives Initiative*, volume 21(2). Bradford UK: Emerald.

Constable, P. and Simons, G. (2000). Language identification and it: Addressing problems of linguistic diversity on a global scale. SIL Electronic Working Papers 2000-001, Dallas: SIL International. Revised version of a paper presented at the 17th International Unicode Conference, San Jose, CA. `http://www.sil.org/silewp/2000/001/`.

DCMI (1999). Dublin Core Metadata Element Set, version 1.1: Reference description. `http://dublincore.org/documents/1999/07/02/dces/`.

DCMI (1999). Guidelines for dublin core working groups - working draft 1.4. `http://dublincore.org/documents/1999/06/02/wgguidelines/`.

Grimes, B. F., editor (2000). *Ethnologue: Languages of the World*. Dallas: Summer Institute of Linguistics, 14th edition. http//www.ethnologue.com/.

Heery, R. and Patel, M. (2000). Application profiles: mixing and matching metadata schemas. In *Ariadne*, volume 25. UK Office for Library and Information networking (UKOLN), University of Bath. http://www.ariadne.ac.uk/issue25/app-profiles/.

Hillmann, D. I. and Sutton, S. A. (2003). DCMI Usage Board (UB) administrative processes. http://www.dublincore.org/usage/documents/process/.

Iannella, R. and Heery, R. (1999). Dublin Core Metadata Initiative – structure and operation. http://dublincore.org/documents/dcmi-structure/.

ISO (1998). ISO 639: Codes for the representation of names of languages-part 2: Alpha-3 code. http://lcweb.loc.gov/standards/iso639-2/langhome.html.

Johnson, H. (2003). OLAC role vocabulary. http://www.language-archives.org/REC/role.html.

Johnson, H. and Aristar Dry, H. (2002). OLAC discourse type vocabulary. http://www.language-archives.org/REC/discourse.html.

Johnson, H. and Dry, H. A. (2002). Olac linguistic data type vocabulary. http://www.language-archives.org/REC/type-20021212.html.

Johnston, P., Cole, T., Habing, T., Hunter, J., Lagoze, C., and Powell, A. (2003). Notes on the W3C XML schemas for Qualified Dublin Core. http://dublincore.org/schemas/xmls/qdc/2003/04/02/notes/.

Lagoze, C. and Van de Sompel, H. (2001). The Open Archives Initiative: Building a low-barrier interoperability framework. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 54–62. http://www.cs.cornell.edu/lagoze/papers/oai-jcdl.pdf.

Lagoze, C., Van de Sompel, H., Nelson, M., and Warner, S. (2002). The Open Archives Initiative Protocol for Metadata Harvesting, Version 2.0. http://www.openarchives.org/OAI/openarchivesprotocol.html.

Powell, A. and Johnston, P. (2003). Guidelines for implementing Dublin Core in XML. http://dublincore.org/documents/dc-xml-guidelines/.

Simons, G. and Bird, S. (2000a). Requirements on the infrastructure for open language archiving. http://www.language-archives.org/docs/requirements.html.

Simons, G. and Bird, S. (2000b). The seven pillars of open language archiving: A vision statement. http://www.language-archives.org/docs/vision.html.

Simons, G. and Bird, S. (2001). OLAC metadata set. http://www.language-archives.org/OLAC/olacms.html.

Simons, G. and Bird, S. (2003a). Building an Open Language Archives Community on the OAI foundation. *Library Hi Tech*, 21:210–218. http://www.arxiv.org/abs/cs.CL/0302021.

Simons, G. and Bird, S. (2003b). OLAC metadata. http://www.language-archives.org/OLAC/metadata.html.

Simons, G. and Bird, S. (2003c). OLAC process. http://www.language-archives.org/OLAC/process.html.

Simons, G. and Bird, S. (2003d). OLAC recommended metadata extensions. http://www.language-archives.org/REC/olac-extensions.html.

Simons, G. and Bird, S. (2003e). OLAC repositories. http://www.language-archives.org/OLAC/repositories.html.

Simons, G. and Bird, S. (2003f). The open language archives community: An infrastructure for distributed archiving of language resources. *Literary and Linguistic Computing*, 18:117–128.

Van de Sompel, H. and Lagoze, C. (2002). Specification for an OAI static repository and an OAI static repository gateway. http://www.openarchives.org/OAI/2.0/guidelines-static-repository.htm.