

A Scalable Metadata Framework for the Virtual Observatory

Raymond L. Plante

DRAFT

Last updated: June 10, 2002

Abstract

The VO will be an evolving environment that will enable increasingly complex interactions between resources. Naturally, expanding capabilities will drive the need for increasingly detailed information to be exchanged in the form of metadata. To encourage a smooth evolution of metadata standards that won't eventually collapse under its own weight, our long-term vision should include the establishment of a structured metadata definition framework. In this paper, I propose a foundation for the development of such a framework that is extendable yet scalable. The goals of such a framework are to unify the use of metadata across the different contexts of a VO application, clarify how the metadata should be bound to specific coding schemes and software interfaces, provide a manageable framework for extending metadata schemas, and specify a structured representation of metadata definitions upon which schema-independent software can be built. I outline an approach to such a framework that attempts to leverage off of existing standards for schema definition, service description, and ontological relationships. I describe how such a framework would be used in the various VO application contexts. This document does not present a fully developed framework; rather, it is hoped that feed the development of a community-based development of a metadata framework. I expect that such a framework will be critically valuable for establishing a process the evaluation and adoption of metadata standards by the global VO community.

Contents

1	Motivation	1
2	The Role of Metadata in a VO Application	2
3	Goals of the Framework	2
4	An Approach to a Metadata Framework	3
4.1	Principles	4
4.1.1	Leveraging Existing Standards	4
4.1.2	Schema Partitioning	4
4.1.3	Mixing Multiple Schemas	4

4.1.4	Definition Styles	5
4.1.5	Use Across Different Contexts	5
4.2	Structure of a Metadata Dictionary	5
4.2.1	Components	5
4.2.2	Possible XML Markup	7
4.3	Example Uses	9
4.3.1	Publishing Metadata Schemas	9
4.3.2	Process of Adopting Standard Schemas	9
4.3.3	Resource Description	9
4.3.4	Cone Search Support	10
4.3.5	Client Configuration	10
4.3.6	Metadata Access within Data Access Layer	11

1 Motivation

The motivation for establishing a standard framework for metadata definition is part lessons from past schema definition efforts and part promise of emerging XML-based technologies. A natural consequence of defining metadata for a broad community can be a large and complex standard. Examples from the Z39.50 community include the BIB-1 (~100 terms) and GEO-1 (over 300 terms). A large standard has real implications for the cost of supporting it, both for those that define the standard and those that must implement it. [*managing and extending*] [*software implications*].

On the bright side, XML-based standards such as XML Schema, WSDL, and RDF provide an opportunity to automate the management and implementation of a metadata standard. They enable us to envision the publishing of dictionaries of metadata within which their meanings, range of values, and interrelationships are encoded in a structured way such that generic software can read the dictionary and understand how to use it. The existence of such generic software provides the hope for making a structured framework *scalable*; that is, as we define more metadata, covering more detailed concepts, the cost of supporting the metadata standards grows little or not at all.

The ideas presented here are based largely on an exploration of metadata frameworks called FlexQuery¹ [1, 2] which was carried out as part of the ISAIA project [3]. One goal of this study was to demonstrate how a metadata dictionary could be used to automate the manipulation of metadata within a distributed search application in a schema-independent way.

¹<http://monet.astro.uiuc.edu/~rplante/topics/FlexQuery/>

2 The Role of Metadata in a VO Application

Metadata come into play in many ways within a VO application. (For a detailed description, see McGlynn [4].) These include:

1. **Resource description.** This is expected to come in two forms. First is the use of metadata to provide general descriptions about what a resource is: its type (archive or service), its provenance, and its coverage of some concept space (e.g. sky, frequency, sensitivity). This information would typically be used for resource discovery. The second form might describe the interfaces it supports and perhaps how it supports it. While this second form would also be used for resource discovery, it could also be used to mediate between different services via analysis of the resources capabilities. The same concepts could conceivably appear in both forms. For example, one archive may contain images from the optical waveband, while another may allow searching for images by waveband.
2. **Metadata exchange with a service.** In general, a service takes an input and returns an output. The most obvious example for the VO is a search service (such as the cone search). The same concept may be used in both the query and the response but not necessarily in the same format or syntax.
3. **Data access through a data model.** Metadata is used to communicate what the data is, how to access it, and how to make logical sense of it.
4. **Semantic analysis.** This refers to a class of contexts in which a process makes use of the relationships between metadata to do some transformation. For example, one might want to convert a position from one reference frame to another. Alternatively, one might want to translate a search query from metadata expressed by the user—say, as a simplified example, a sky cone search—into metadata the a data resource supports—say, a sky rectangular search.
5. **Data dictionary.** Traditionally, a data dictionary is usefully primarily to real people: for developers, it provides a reference for building support for metadata into software; for users, it might help using an application. A structured format for the dictionary allows it to be used to configure software automatically (as described below in this document).

A major aim of a metadata framework is to unify the use of metadata across these different contexts.

3 Goals of the Framework

The goals are as follows:

1. *To unify the use of metadata across the different contexts.*

In particular, we want to ensure that a concept used in one context is the same as in another context, even if they are rendered in different ways. In particular, the metadata should behave in the same way. For example, restrictions on a metadatum's value—e.g. type, allowed values—should be the same in all contexts. If a query tests to see if a metadatum matches one of a set of controlled values, the rendering of the metadatum in the response must draw from the same values.

2. *To clarify how metadata can be bound to various encoding schemes and software interfaces.*

A structured definition of metadata will elucidate the most obvious way to render the metadata in a wire protocol or store it in memory. Where it is not obvious, it should be possible to describe the binding explicitly. (This is the strategy used by WSDL.) This will make it easier to have the different renderings look as similar as possible.

3. *To provide a manageable framework for extending schemas or creating new ones.*

“Manageable” can mean several things:

- It’s not necessary to define all the metadata for the entire application domain from the top down in order to make use of the metadata in an automated way.
- It is possible to use multiple (complementary) metadata schemas in a single application without difficulty. This allows, then, one to partition the concept space in logical and practical ways; applications can then pick and choose which parts to use.
- It’s not necessary to get the definitions correct the first time if the schema can be easily changed in the future.
- It’s possible to create new, specialized metadata integrated with “standard” VO metadata for specialized applications involving only a few resources. It is thereby possible to test out candidate schemas for VO “standard” status.

4. *To enable the development of schema-independent software for supporting metadata.*

For example, we can conceive a package that can be configured to convert a query using a standard VO syntax into a form natively supported by a data resource. If this package has access to the metadata definition, this software can be made schema-independent. While some uses of metadata within software will have to be metadata-specific, many uses of metadata—most notably, searching—can be done without special hard-wired knowledge of the metadata’s meaning. This is a powerful way to reduce the cost to curators of supporting new metadata as part of a standard search service. A major result of the FlexQuery project demonstrating the usefulness of this capability.

4 An Approach to a Metadata Framework

The key feature of the metadata definition framework proposed here is the ability to encode the definition of metadata as an XML document. Traditionally, one might consider such a document as a *schema definition* or a *metadata dictionary*; however, we need to encode more than just the prose definitions and the syntax associated with the metadata. We would also like to define the supported operators that can be used to do such things as compare metadata values with test values. Finally, we may wish to include semantic relationships between metadata and descriptions of how the metadata can be rendered in different formats or contexts. Thus we might think of the XML document as a *metadata manual*. In this document, use of the term *metadata dictionary* implies this greater scope of information.

It is important to note that the aim of the framework described here is very similar to the aims of the RDF Schema (RDFS) and DAML+OIL standards. The extent to which this framework is similar—or more importantly, different—to those frameworks is unclear at this time. Clearly, it makes sense not to reinvent a wheel where it already exists; thus, *if RDFS/DAML+OIL can be used to meet the goals outlined above, then we should adopt them to encode our metadata dictionaries*. Important caveats to keep in mind when considering an RDFS/DAML-based framework are the availability of supporting software and the ease of which the framework can be understood and applied properly by our community.

4.1 Principles

This document does not present a complete framework design but rather a foundation or initial approach to the design of such a framework. Thus, it is worth spelling out a few guiding principles upon which this approach is based.

4.1.1 Leveraging Existing Standards

The framework should leverage of existing standards where they are appropriate. The key ones are XML Schema, WSDL, and RDF. “Leveraging” usually means making it possible to use software that complies with these standards naturally with the framework.

For the case of XML Schema, clearly the most important of the three, it should be straightforward to render the metadata dictionary in XML Schema. This might be done in one of several ways:

- define the metadata syntax (and possibly prose definition) in an external XML Schema document which is reference or perhaps “included” in the metadata dictionary document.
- use XML Schema markup directly to define metadata definitions and types within the document. This is the strategy used by WSDL for defining metadata and types that are used in service messages.
- use markup that derive from XML Schema markup, extending it as necessary
- use custom markup that is modeled closely after XML Schema with the extra needed features added in.

For the last three alternatives, it should be possible to create an XSL stylesheet that converts a metadata dictionary into an XML Schema document which can in turn be used to verify XML documents. Such a transformation should be metadata-independent, relying only on the definition markup.

As for WSDL integration, it will be useful to combine metadata information in with a WSDL description. For example, for a cone search service, it would be useful to indicate the range of values of the searchable metadata that will return none non-empty results—that is, the coverage of the underlying database. The WSDL specification provides a mechanism for inserting such information via WSDL extensions.

As discussed above, RDF, RDF Schema, and DAML+OIL all represent existing technologies relevant to NVO metadata; in particular, what they provide over XML Schema is a richer encoding of semantic relationships.

4.1.2 Schema Partitioning

It should be possible to partition the concept space into separate schemas. For example, we can define the space-time metadata (see Rots [5]) within a single schema. Another schema may handle resource provenance; while another, specifically address optical filters.

4.1.3 Mixing Multiple Schemas

It should be possible to use multiple schemas in the same application when they have all been defined within the framework. This means there needs to be a way to uniquely identify schemas and the metadata they define. This is normally done through the use of namespaces.

4.1.4 Definition Styles

Organizations (e.g. the NVO) can adopt specific requirements on how metadata can be defined. For example, they may wish to exclude the use of attributes as part of the definition of a metadata element.

4.1.5 Use Across Different Contexts

A particular metadata dictionary document can be reused, with perhaps small changes, for a different purpose within a different context. This is a central feature of the framework that makes it easier for providers and clients to conform to schema standards.

Figure 1: The different roles of the metadata dictionary in a catalog search application. (Figure taken from [1].)

Consider a cross-catalog query application that specifies specific metadata that should be supported as part of a query. (This is the example is explored in the FlexQuery demonstration.) In this case, there are three key types of players that handle a metadata dictionary defining the metadata used in the application. The process starts with a schema definer creating and publishing a metadata dictionary document. A service provider, by downloading the document, can use the document in two ways. First, she might edit the document to indicate the range of values in the search terms that apply to her service; when integrated into a WSDL description of the search service, it becomes a service registration document that can be submitted to a global registry. Next, she keeps a second copy locally, adding in tags that describe how this metadata should be mapped to the native attributes of her database; standard software that understands the metadata dictionary would be employed to do that conversion automatically each time a search query is received. The application or portal developer can make use of the metadata dictionary as well for generating search GUIs on the fly.

4.2 Structure of a Metadata Dictionary

4.2.1 Components

In general, a metadata dictionary document would contain the following components:

Identification The most important part of this component is an identifier for the schema the document describes; this identifier serves to establish a namespace that allows the schema to be mixed with other schema in VO applications. Also included should be an identification of the organization (or individual) responsible for the definition of the schema. Furthermore, to facilitate the extension of the schema, it should be possible to indicate a revision identification. Finally, this component should include a piece of markup that indicates the current context of the document. This might be an attribute with the following possible values:

- definition** an original schema definition, independent of any application.
- resource** a resource description
- service** a service description
- client** a schema used to configure a client application.

It should be noted that with XML Schema, namespaces are identified by a URI for the XML Schema document (e.g. `xmlns="http://www.example.com/ExampleSchema"`). XML instance documents can associate a namespace prefix with this URI; that is, the XML Schema document does not enforce a particular prefix. A similar scheme could be used with a metadata dictionary as long as the following caveats are considered:

- Responsibility and revision information, as described above, should be handled sufficiently.

- Not all uses of the metadata may be in the form of XML (e.g. an SQL query); a natural way of handling namespaces will be necessary. (The specifics would most appropriately be handled in the Binding component of the document; see **Bindings**, p. 6.)

Syntactic Types Like XML Schema, it is important to associate metadata with value types. Like an WSDL document, this component allows one define new types or import existing types defined in other schemas.

Dictionary This matches a metadatum’s name with a type, a human readable definition, allowed values, and the definitions of any controlled values.

Operators that can be used with a metadatum could be defined here, as well. An operator can be thought of as a function whose principle input is the associated metadatum value. This is an extension over traditional metadata dictionaries that would be useful primarily to search applications. Most operators, therefore, would compare a metadatum’s value against a test value and return true or false. While primitive types might have the usual operators predefined (e.g. “less than” and “greater than” for numeric values; “contains” for strings), it would be possible to define operators that are specific to a particular metadatum. For example, if one defines a metadatum that represents a person’s name, one might also define an operator called “last name starts with.” Specialized operators would be most helpful for complex types such as sky position or date.

It may, depending on the context, include as part of the metadata definition other markup, such as user-oriented tips (used in the “client” context) or definitions of operators that can be applied to the metadata.

Semantic Relationships This describes how a particular metadatum is related to other metadata. The RDF resource—property—value model (also referred to as “subject—predicate—object”) is an appropriate model for doing this. In particular, we could use OIL+DAML markup to describe these relationships.

Bindings Similar to the Bindings section in a WSDL, this component describes how the schema can be bound to different encoding schemes and software APIs. Through proper leveraging off of XML Schema, a rendering of the metadata in XML may be obvious; in this case, a description of the binding to XML may not be necessary. An important use for this component is to describe the encoding of metadata in a particular query syntax. It could be used by a client application to create query messages from inputs entered into a GUI. A more important use, however, is by data providers to describe how to convert standard queries into native forms. Standard software made available to the provider which understands the markup would carry out the translation. Thus, if the data provider wishes to support a new set of metadata (and their operators), she need only describe the mapping in XML; no new software need be written.

4.2.2 Possible XML Markup

The example below is meant to give the flavor of how one might define a set of metadata to describe radio interferometers. In particular, it defines a new type, “antennaPair,” that could potentially be associated with multiple metadata. This new type is used to define a metadatum called “Baseline Name.”

```
<?xml version="1.0"?>
<!DOCTYPE VOSHEMA SYSTEM "http://www.us-vo.org/xml/voschema.xsd">
```

```

<!-- The root element.
The context attribute indicates that this is an original metadata
definition document.
The name attribute gives an optional identifier for this document.
The defns attribute provides a suggested namespace identifier to use
when rendering the metadata in some form (which may not be XML).
The remaining attributes identify the namespaces that will be referenced
in this document.
-->
<VOSCHEMA context="definition" name="interferometer" defns="vointf"
targetNamespace="http://www.us-vo.org/metadata/dictionaries/interferometer"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:votef="http://www.us-vo.org/metadata/dictionaries/telescopes"
xmlns="http://www.us-vo.org/xml/voschema/">

<PROVENANCE>
  <!-- identification of responsible organization/individual -->
  <!-- identification of version/revision -->
</PROVENANCE>

<TYPES>
  <!-- import standard XSD types (this may not be necessary) -->
  <import namespace="http://www.w3.org/2000/10/XMLSchema"
location="http://??"

  <!-- import types defined in a previously defined metadata dictionary -->
  <import namespace="http://www.us-vo.org/metadata/dictionaries/telescopes"
location="http://www.us-vo.org/metadata/dictionaries/telescopes.vomd" />

  <!-- create new types using XML Schema markup -->
  <TYPE name="antennaPair">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d+-\d+" />
      </xsd:restriction>
    </xsd:simpleType>
    <OPERATORS>
      <!-- define some default operators for comparing antenna pairs -->
    </OPERATORS>
  </TYPE>

</TYPES>

<DICTIONARY>
  <ELEMENT name="Baseline Name" type="antennaPair">
    <ANNOTATION>
      <DEFINITION xml:lang="en">
        An identifier for an interferometer baseline indicating the
        two antennae that make up the baseline.
      </DEFINITION>
      <TIPS>
        <!-- Tips correspond to XML Schema's appInfo -->
        <TIP role="implementer" xml:lang="en">...</TIP>
        <TIP role="user" xml:lang="en">...</TIP>
      </TIPS>
    </ANNOTATION>
  </ELEMENT>
</DICTIONARY>

```



```

    <TIPS>
  </ANNOTATION>

  <VALUES>
    <!-- This describes the domain of possible values for this
          metadatum. It should be straightforward to convert
          this into XML Schema markup that defines a simpleType
          that inherits from another simpleType with
          restrictions. If specific vocabulary are specified as
          allowed values, the markup should require that their
          definitions be included as well. -->
  </VALUES>

  <OPERATORS>
    <!-- This metadatum would "inherit" any previously defined operators
          associated with the antennaPair type. Additional operators
          specific to this metadatum would be defined here if necessary. -->
  <OPERATORS>
</ELEMENT>

  <!-- additional metadata elements would be defined in the same way as necessary -->
</DICTIONARY>

<SEMANTICS>
  <!-- semantic relationships between the metadata defined above,
        both amongst each other and with metadata defined in other schemas,
        would be encoded here -->
</SEMANTICS>

<BINDINGS>
  <!-- This section would contain markup describing how the metadata can
        be rendered in some encoding scheme. In particular, conversions
        to native metadata schemes would be described here. -->
</BINDINGS>

</VOSCHEMA>

```

4.3 Example Uses

4.3.1 Publishing Metadata Schemas

In the sample markup given in the previous section, the `context` attribute of the root element is set to “definition,” indicating that the document is meant to represent an original definition of a metadata schema. Typically, the schema would cover a limit scope of related concepts. The basic definitions, including the metadata elements, the human-readable definitions, and operators, would be required. The semantic relationships, which may refer to externally defined metadata, would be set down at this time as well. Bindings would be defined for standard renderings such as within the query syntax. On the other hand, conversions to native metadata would not be defined, as this definition should be independent of any particular service’s support of it.

To publish the metadata dictionary, it need only be made web accessible so that it can be referred to in other XML documents. For standard VO metadata dictionaries, it would be useful to make the document

searchable along with other standard metadata dictionaries; thus, it could also be loaded into a searchable registry.

4.3.2 Process of Adopting Standard Schemas

The structured format provides a convenient vehicle for proposing, testing, and adopting new metadata schemas. The advantage of the structured format is realized with existence of schema-independent software that can manipulate the metadata in useful ways.

Consider the following scenario. Imagine that three institutions wish establish an application that interoperates across their respective resources which requires new metadata not covered by a current VO standard. They can create a new metadata dictionary that defines the new metadata and share it amongst themselves, to be used in addition to existing standard schemas. Using standard VO metadata software, they can integrate the new metadata into their application with little or no additional programming needed (depending on the nature of the application). If their application is successful, they can decide to propose the new schema be accepted as a VO standard. Or, if they don't expect the new metadata to be applicable outside their narrow focus, then they can choose not to propose it as a standard.

The VO may impose certain requirements or restrictions on how the metadata are defined. The first obvious requirement would be that the schema be submitted in the metadata dictionary XML markup format; however, other requirements might be imposed as well, such as metadata types should not include XML attributes. A testing and evaluation period can be set; the existence of software that understands the dictionary would accelerate this process. If the proposal is accepted, then the provenance information is changed to those of the VO community, and the document is registered in the VO dictionary registry.

Proposals for correcting or extending existing schemas would proceed in a similar way. Here, the use of revision numbers will be critical to managing changes of this type. Again, the ability to use existing metadata software will be critical to identifying problems of backward compatibility.

4.3.3 Resource Description

The process of resource description would start with the creation of a VO-adopted metadata dictionary that defines the general metadata that describes a resource. A resource curator would download this document and edit it to describe their own resource.

In terms of the example given above, he would accomplish this by first changing the root element's context attribute to "resource"; this establishes that the document now is a resource description. Next, he would, for each defined metadatum, alter the VALUES description so that it now describes what part of the domain applies to the resource. For example, suppose the metadatum "waveband" accepts possible values of "gamma-ray", "x-ray", "ultraviolet", etc., the curator would eliminate all waveband names that do not apply to the resource. If the metadatum "bandpass" accepts a range of frequencies, he would enter the minimum and maximum that applies to his resource.

Of course, it's not required that the curator manually edit the document with a text editor. Because of its structured format, it is straightforward to dynamically build a web form allowing the curator to enter the value restrictions.

When the changes are complete, the edited form of the document would be submitted to a VO registry.

One might argue that under this model, the resource description contains information that is not needed or is redundant. One example is perhaps the metadatum definition. However, it is actually useful to be able

to render the meaning of the metadata along with the values as they apply to the resource. In particular, it makes it easier for the curator to complete the description if the meanings of the metadata are right there without having to look them up. Of course, an application can ignore any information in the document that it doesn't need.

4.3.4 Cone Search Support

In this example, we are describing a resource's support for a standard service, such as the cone search. Ideally, we would like to describe supported services using WSDL. However, in addition to describing the interface (via WSDL markup), we might also like to specify additional information that gives us some idea of how it might behave. In the case of the cone search, it would be useful to describe what the queryable columns are and what range of values can be returned in a query response. Including this in the service description would allow portals and clients to make intelligent decisions about whether to send a query.

In general, the technique for specifying the range of possible values that can be returned by a service could be done in the same basic way as described in the previous section on *Resource Descriptions* (albeit, in a way that integrates into a WSDL service description). Curators could also alter the metadata descriptions to indicate which operators it can support natively; this information could be used by portals to strategically alter the query to match the capabilities of the service. The context attribute appropriate in this case would be "service."

4.3.5 Client Configuration

Client interfaces can also take advantage of metadata dictionaries. Consider a general-purpose search tool which can be used with any search schemas. Such a client could download metadata dictionaries at run-time and create graphical interfaces to supported search operators on-the-fly. If a particular metadatum only allows values taken from a controlled list of strings, the interface might present an interface that allows the user to click on the desired values to search against.

The fact that annotations are included in the dictionary is also valuable: it allows the client to present this information as help to the user. Furthermore, instead of downloading the dictionary at run-time, the client could use its own version that has been modified to, say, provide additional tips to the user about searching on particular metadata. (In this case, the `context` attribute should be changed to "client.")

A general-purpose search tool of this type was demonstrated as part of the FlexQuery project (see Fig. 2).

Figure 2: The FlexQuery demonstration search client (taken from Plante et al. 2002 [2])

4.3.6 Metadata Access within Data Access Layer

In practice, the structured form of a metadata dictionary may have not have much run-time impact on how metadata is accessed via a standard Data Access Layer. Instead it may have an impact on the interface design and its evolution as more metadata are incorporated into the layer infrastructure. The dictionary will provide a blueprint for how access to metadata should map into the interface, either implicitly or via an explicit binding description. This will be helpful in particular for complex metadata (e.g. space-time metadata) that are naturally hierarchical in structure.

A common practice now is to use schema definitions to generate software classes automatically. Straight-forward conversion of a metadata dictionary into XML Schema would allow us to use off-the-shelf code

generators of this type.

References

- [1] Plante, R., Guillaume, D., & Mehringer, D. <http://monet.astro.uiuc.edu/~rplante/topics/FlexQuery/>
- [2] Plante, R. L., Guillaume, D., Mehringer, D., & Crutcher, R. 2002, in Astronomical Data Analysis Software and Systems XI, in press.
- [3] Hanisch, R. J. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP), 201
- [4] McGlynn, T. 2002, Metadata Working Group Discussion: “Metadata frameworks for the VO: An operational view”, <http://archives.us-vo.org/metadata/0136.html>
- [5] Rots, A. 2002, Metadata Working Group Discussion: “Space-time Metadata”, <http://archives.us-vo.org/metadata/0050.html>