OPENDAP: ACCESSING DATA IN A DISTRIBUTED, HETEROGENEOUS ENVIRONMENT

***P. Cornillon*[*], *J. Gallagher*[*], *T. Sgouros*[†]**

[*]*Graduate School of Oceanography, University of Rhode Island, Narragansett, RI, 02882 Email: pcornillon@gso.uri.edu*
[†]*Manual Writing, 15 Boston Neck Road, Wickford, RI, 02852, Email: tom@sgouros.com*

ABSTRACT

In the process of implementing a protocol for the transport of science data, the Open Source Project for a Network Data Access Protocol (OPeNDAP) group has learned a considerable amount about the internal anatomy of what are commonly considered monolithic concepts. In order to communicate among our group, we have adopted a collection of definitions and observations about data and the metadata that make them useful: differentiating between "semantic" and "syntactic" metadata, and defining categories such as "translational" and "use" metadata. We share the definitions and categorizations here in the hope that others will find them as useful as we do.

Keywords Metadata, data transport, syntax, semantics, data archive, data model, data attributes.

1 INTRODUCTION

In the distribution of science data, issues of format incompatibility have for years been among the most significant obstacles, hindering scientists from freely sharing their data, and impeding or discouraging correspondence with central data archives. To begin to address these issues for oceanographic data, scientists at URI and MIT undertook a project called the Distributed Oceanographic Data System (DODS) in 1993 (Sgouros 1996-2002). The objective was to make data sharing seamless, and allow scientists to import data into their data analysis software directly from remote sites, regardless of the format in which the data were stored.

The DODS effort consisted of two fundamental parts: a part that focussed on how data were to be moved over the network within the system, the data access protocol or DAP, and a part that focussed on the data sets served by the system, specialized applications for the use of these data, a directory of these data sets, and so on. In that the system was to be used for all classes of oceanographic data and because these data span such a large range of data types, formats and organizational structures, the DAP was by necessity designed to operate at a very low level. The result was that although developed for the oceanographic community, there was nothing in the DAP (based on C^{++} data types) that was oceanography-specific; i.e., the DAP could be used to move information about art objects as readily as information about the ocean. The two basic parts of DODS therefore consisted of a discipline-neutral portion, the DAP, and a discipline-specific portion, those parts that were oceanography-specific—data set population, specialized clients, and the like.

The neutrality with regard to scientific discipline of the DAP was quickly recognized by scientists in other fields who began to adopt the DAP in their data system development efforts. However, there was a concern among these same groups that with the primary focus of the DODS effort being on oceanographic data

system needs, evolution of the core software might diverge from their needs. As a result, specializations of the DAP began to appear that addressed problems faced by all but that took different approaches. The result was an undesirable duplication of effort. In order to address this, the Open Source Project for a Network Data Access Protocol (OPeNDAP) was established in 2000 for the development and promotion of software that facilitates access to data via the network. OPeNDAP is a non-profit corporation operating in Rhode Island. At the same time that OPeNDAP was created, the oceanography-specific part of DODS was named National Virtual Ocean Data System (NVODS). NVODS uses the OPeNDAP data access protocol. OPeNDAP has a broader mandate than DODS in that it focuses on network access to data in general. The present account concerns the OPeNDAP data access protocol.

Although the objective of seamless access to data over the network has to a large extent been met by OPeNDAP, some unexpected obstacles have turned up while other anticipated roadblocks failed to materialize. In the process of implementing real solutions to these problems we have learned a considerable amount about the internal anatomy of what are commonly considered monolithic concepts. In order to communicate among our group, we have adopted a collection of definitions and observations about data and the metadata that make them useful. We share these here in the hope that others will find them useful.

2 LAYERS OF INTEROPERABILITY

The essential problem addressed by OPeNDAP is to have a computer program operate on some data provided by some other institution, stored in some unknown format on some remote computer. One important outcome of the work on OPeNDAP is the observation that this is not a binary problem. That is, there are different *degrees* of success in getting intelligible data from a server to a client.

Part this realization came from a subtle restating of the problem. It is often the case that OPeNDAP users want to compare data from different sources. In that case, the user generally wants to see comparable data presented in identical fashion. This means that it is not enough simply to be able to receive data from remote files. It means that data from those remote files should appear uniform, to some degree. The more uniform data from different sources appear, the easier it is to compare them, and the more likely it is that some data processing can be automated. This is what is meant by rendering data "interoperable." We sometimes use "machine-to-machine interoperability" to make even more explicit the degree of uniformity required to be considered interoperable.

As there can be said to be differing degrees of uniformity, there are also differing degrees of being able to operate on remote data. As a matter of convenience in discussing these degrees, we have defined several "layers of interoperability," shown in Figure 1. These layers are conceptually similar to the layers of abstraction defined in internet design documents, and may be thought of as a somewhat more elaborate subdivision of the "presentation" and "application" layers defined in the OSI (ISO) model of the internet. Internet design documents using this model define six distinct layers of abstraction above the basic level of oscillating voltages connecting two computers. Roughly, they are: the physical layer (the kind of cable connecting neighboring machines), the data link layer (the kind of communication between one machine and its neighbor), the network layer (how the computer network is laid out), the transport layer (how do data get from one computer to another), the session layer (how does a program on one computer cooperate with a program running on another), the presentation layer (what kind of data are being received), and the application layer (how should the data be displayed to the user). See IEC (1994), Stevens (1999).

Despite wide acceptance of this model, we have found the standard division of the layers do not reflect the situations we have encountered. Our model presented here should not be viewed as the last word for all situations, and there is substantial overlap between the standard model and ours. For example, OPeNDAP does not address the levels below the session layer (though, not being tied to any particular transport or session architecture, it can take advantage of recent advances made on those levels). But we think of our model as a better description of a situation where the sources of data, the types of data, and the kinds of

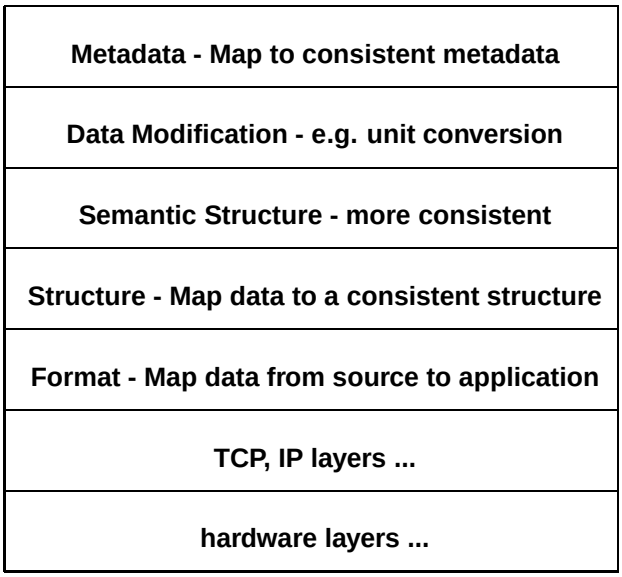

Figure 1. Layers of Interoperability

clients are somewhat more heterogeneous than is usually considered normal in other discussions of the internet.

An item of scientific data is accorded some meaning solely by its relationship to other items. A number in some time series is meaningful only to the extent it can be accurately placed within that series. Another number in some three-dimensional grid of measurements only has meaning in the relation it bears to its six neighbors, and its position in that grid. These are somewhat more intricate forms of context than is usually relevant in discussions of data transmission. For example, a word transmitted in a piece of text also bears meaning in relation to its context, but the relevant context—at least insofar as low-level issues of transmission format are concerned—is a fairly simple nearest-neighbor sort of context. That is, so long as the words and formatting directives in some HTML file are transmitted in the correct order, they have a fair chance of being understood at the other end.

Transmitting graphic images bears some relation to the problems of transmitting science data. For example, a pixel of some picture, especially one encoded in a progressive format like a progressive GIF, bears a similarly complex relation to its neighboring pixels, which may be encoded in distant parts of the file. But these formats avoid the problems of defining context by subscribing to a much more strictly defined Application Program Interface (API) than is possible for a file format meant to be useful for more than one purpose.

The range of data types as well as relationships between these data types and permissible operations associated with them defines the "data model" of a given API. For a graphic image file format, one might say that its data model was rigid, whereas for a multi-purpose API, it is important that the data model retain some flexibility.

The norm in the analysis of scientific data is that data are stored in a wide variety of different file formats, read and written each by their own specialized API. Some of these are widely used standards, while others are somewhat more *ad hoc*. Few are compatible with each other. For example, the netCDF and HDF formats are two standards popular in the world of earth sciences. Both work well for many of the same tasks. But they are not compatible with one another: files written by one are not readable by the other, even if the data in each file were similar to the other. See Rew, Davis & Emmerson (1993, April) or Rew & Davis (1990) for information about netCDF. See NCSA (2001) about HDF. NCSA (the National

Center for Supercomputing Applications), the developers of HDF, provide software that allows the HDF library (HDF4) to access netCDF files, but it does so by including the netCDF software.

The goal of OPeNDAP is to allow users to request data from remote sources, and to allow them to imagine the remote data to be stored in whatever format is most convenient for them. They should be able to download the data directly into the data analysis programs they are using. OPeNDAP provides both server software, to make data available to remote users, and client software, to access those data. Some of the client software can be used to convert programs written for a data access API like netCDF into network-ready "browsers" of remote OPeNDAP-accessible data.

The first step in achieving this goal was to address the "format" layer, the first layer above the one labeled "TCP" in the diagram in Figure 1. This refers to the ability of a program designed for some API simply to read data from some remote file, no matter what that remote file's format is. To do this, the OPeNDAP server software re-formats data into a well-defined transmission format (see Section 3), which is received by the client software and re-formatted again to be digestible by that client software.

This works, but the difficulty is that if an API is powerful enough to be useful to a wide variety of researchers, it is also powerful enough to be used in many different ways to solve the same problem. A trivial example here would be two different records of sea surface temperature, recorded on one-degree grid squares, with one using latitude and the other longitude as the X dimension. If a user of OPeNDAP client software was interested in comparing data from these two sources, the fact that they were both transmitted perfectly from their respective servers would be somewhat beside the point. That is, the arrays are inherently incompatible with one another, even though they share (or may be made to appear to share) the same file format.

For a more common example, consider a time series of two-dimensional satellite measurements. One scientist might reasonably choose to keep such a time series as a single netCDF file containing a three-dimensional grid of data, while another group might record the data as a collection of individual files, with one two-dimensional record in each file. Again, the fact that the two data sets are (or can be made to appear) stored in compatible formats is of little relevance to the problem of trying to compare data from the two sources.

With adequate metadata, both situations can be resolved mechanically, and in fact, the OPeNDAP project now supports an "Aggregation Server" to address situations like the second example. But before further discussion of the interaction of different kinds of metadata, it will be useful to present a cursory overview of the OPeNDAP data transmission protocol.

3 THE DATA ACCESS PROTOCOL

The OPeNDAP protocol defines how an OPeNDAP client and server communicate with one another. It consists of four components:

1. An intermediate data representation. This is used to transport data from the remote source to the client. The data types that make up this representation may be thought of as the OPeNDAP data model, though the array of available types is designed to accommodate many other data models.

2. A format for the ancillary data needed to translate a data set into the intermediate representation, and to translate the intermediate representation into the target data model. This format provides a description of the shape and size of the various data types stored in some given data set and is called the Data Descriptor Structure (DDS).

3. A procedure for retrieving data and ancillary data from remote platforms.

4. An API consisting of OPeNDAP classes and data access calls designed to implement the protocol,

The last two components are not important to the discussion in the present article, and will not be treated further here. The intermediate data representation itself consists of several components:

- A set of simple data types. These include bytes, integers, strings, floating point numbers (all to several different precisions), as well as a specialization of a "string" meant to hold an internet Universal Resource Locator (URL).
- A set of "constructor" data types, constructed from other data types (including other constructor data types). These include simple collections (structures), ordered collections (arrays), relational tables (called "Sequences" here), and "Grids", a structure containing a multidimensional array and a set of one-dimensional arrays to describe its dimensions.

The intermediate data representation also consists of a set of operators for selecting different segments of the data in some data set, and a definition of the external data format for transmitting simple data types (OPeNDAP uses Sun's XDR definitions for most of the simple data types, see Sun Microsystems (1987, June)), but these are also not relevant to the discussion here.

3.1 The Data Descriptor Structure (DDS) and the Data Attribute Structure (DAS)

The OPeNDAP system uses two different kinds of messages to describe the data it sends. One, the DDS, mentioned above, describes the data structures to be filled by the accompanying data. The other, the DAS contains whatever other descriptions are necessary for a client to make sense of the data. An example DDS and DAS are shown in Figure 2. The development of the DDS and DAS predate the development of XML. The next version of OPeNDAP (4.0) will replace the DDS and DAS formats shown here with an XML rendering of the same information. The general principles outlined in this article will not change, nor will the terminology we use to describe them. The DDS did not predate the development of ASN.1, and its associated ISO encoding rules. But until recently those standards' support for floating-point numbers was inconvenient (requiring the separate specification of mantissa, base, and exponent, for example), and the standard still does not support the concept of an indexed array. At the time, these shortcomings made ASN.1 impractical for science data. As the standard evolves (it now includes an XML form), it may eventually become appropriate for use with OPeNDAP (see International Telecommunications Union [ITU], 2002, July). Ultimately, the choice of notation is not important to the metadata taxonomy presented here.

The data described by the DDS in Figure 2 are stored as a catalog number and a Sequence (a relational table with no set length), with each record containing the name of the experimenter, a 32-bit integer containing the time, a Structure containing latitude and longitude values, and another Sequence containing measurements of depth, salinity, and temperature.

The DAS in Figure 2 shows additional information that is helpful to make sense of the data that arrives with the DDS. The DDS indicates that time is sent as an integer, but it is the DAS that explains how the time is encoded in that integer. The DDS explains that salinity values are encoded as 64-bit floating-point values enclosed in a Sequence that also includes depth and temperature values. The DAS defines the units of those salinity values.

Both the DDS and the DAS contain metadata: information about the data that, together with that data, make up the message. We refer to the metadata held in the DDS as "syntactic" metadata, information that describes the syntax of the data set, and to the metadata held in the DAS as "semantic" metadata, information about the semantics, or meaning, of the data values.

Over the course of the project it has become clear that a good way to categorize these kinds of metadata is to say that the information held in the DDS is "discipline-independent" while the information held in

```
Dataset {                    Attributes {
  Int32 catalog_number;        catalog_number {
  Sequence {                     String type "UN-format";}
    String experimenter;       station {
    Int32 time;                  String shipName "Lollipop";
    Structure {                  Int32 stationID 4533;
      Float64 longitude;         time {
      Float64 latitude;            String format "Seconds since 1/1/1970";}
    } location;                  location {
    Sequence {                     latitude
      Float64 depth;                 Float32 actual_range 0, 30.0;}
      Float64 salinity;            longitude {
      Float64 temperature;           Float32 actual_range -50.0, -48.0;}
    } cast;                      cast {
  } station;                       salinity {
} data;                              String units "ppt";}
                                   temperature {
                                     String units "deg Celsius";}}}}}
```

Figure 2. An example OPeNDAP DDS (left) and DAS (right).

the DAS is "discipline-specific." That is, the syntactic metadata, having to do with the structure of the data types making up the data, is assumed to be consistent with many different scientific disciplines, and so are put in the DDS, while the particular sets of metadata associated with oceanography, meteorology, solar physics or some other scientific discipline belong in the DAS. The fact that some data is packaged into arrays is a point of syntax. That it contains temperature measurements is a point of semantics.

There remain some points of ambiguity about these categories that merit further examination.

4 CLASSES OF METADATA: SEMANTIC AND SYNTACTIC

Whether a metadata object is syntactic metadata or semantic metadata depends on how it it to be used. For any computer program involved in the transmission of data, situated somewhere in the OSI layer model, "syntactic" information is what is used to transform the message from one representation into another as that message is passed up or down the levels. The message itself, or, in our terminology, the data and associated metadata, are simply whatever gets transformed and passed along to the next level in the model. For example, "syntax" means something different to a computer running an OPeNDAP client, than it does to one acting as a router, even if one is talking about the same message. The OPeNDAP client can use information about the size and data type of some array arriving from an OPeNDAP server. Had it needed to, the router through which this message passed could also extract these metadata values, but they would be irrelevant to its operation. The only syntax it needs to understand is enough to peel out the address to which the message is to be delivered, and to use rewriting rules to update the routing information in the packet headers. Not needing information about data types and structure, this computer considers OPeNDAP syntactic metadata to be "semantic" metadata and passes them along intact with the OPeNDAP semantic metadata and the data values themselves.

Syntactic metadata, absolutely necessary for properly receiving a message, is often at least partially implicit. A receiving program is built to understand a particular kind (or kinds) of data. The fact that this information is implicit in the operation of the receiver should not mask the fact that it is information nonetheless, and is necessary for the proper decoding of the message. Whether the decoding instructions come with the message or by some other method, the receiver of the message must have them if the

message is to be understood. The syntactic metadata might not travel with the message, but it still must be transmitted from data source to data user (or to both from a common third point), possibly by some software distribution system, an internet RFC document, or word of mouth.

The OPeNDAP approach is to make the dependence on syntactic metadata a little more explicit, enclosing syntactic metadata with the message, in something of the spirit of a multipart MIME document, which contains its own instructions for unpacking. It is probably impossible to make the dependence completely explicit, because the layers of dependence eventually rest on common cultural assumptions which are unlikely to lend themselves to convenient encoding. For example, at root the OPeNDAP protocol depends on Sun's XDR encoding for transmission of basic types, which depends on the use of eight-bit words and the meaning of floating-point decimal numbers, which itself depends on the decimal system, and so on. These things are implicit in the construction of both OPeNDAP servers and clients and are not encoded in the explicit syntactic metadata in the DDS.

However, even though it is probably impossible to encode everything, the OPeNDAP project has opted to encode more information into the message than is often the case. The OPeNDAP data model is extensible, including both basic types like integer and floating point numbers, and constructor types, like structures, lists, arrays, and relational tables (sequences). This allows the transmission of a vast range of possible data types, and allows the OPeNDAP software to be used with software encompassing a wide variety of data models.

A final note about the distinction between syntax and semantics: the relative nature of the terms notwithstanding, it is convenient to have some way to discuss the differing metadata types. Therefore, in what follows, it is to be understood that the terms "syntactic" and "semantic" are used relative to the operation of the OPeNDAP software, following our definition with respect to scientific disciplines.

5 CLASSES OF METADATA: SEARCH AND USE

One of the central motivations of discussions of metadata is making order of a complex world. There are a lot of data available, and it is important to establish ways to find these data. However, in the discussion of metadata, we have found it useful to distinguish between metadata required to *use* a data set, and those required to *find* a data set containing data of potential interest. We refer to the former as "use" metadata and the latter as "search" metadata. This distinction is quite important because, in our experience, most metadata discussions center around search metadata requirements and not use metadata requirements. For example, the Directory Interchange Format (DIF) of the Global Change Master Directory (GCMD)(GCMD, 2001) focuses almost exclusively on search metadata, such as data ranges. The directory is, of course, designed to be a search tool, making this focus entirely appropriate. Nonetheless, it should be clear that the OPeNDAP definition of metadata covers material not considered to be in that category by the GCMD. The Library of Congress Metadata Encoding and Transmission Standard (METS), another prominent nexus of metadata discussion puts what we call search metadata into the "External Descriptive Metadata" section (under the *mdRef* element), though some use metadata would go there, too. The DDS information belongs in the METS "Internal Descriptive Metadata" section (*mdWrap*), while our use metadata category would be spread among the remaining four sections of a METS entry. (METS, 2002) There is overlap between use metadata and search metadata, but one is not a subset of the other. To search for a data set, missing value flags are not required, while such information is crucial to using that data set. Similarly the range of the variables in a data set is not required to use the data but forms the basis for many data set searches. On the other hand, quality estimates for data could fit in both categories.

The Federal Geographic Data Committee (FGDC) metadata standard (Federal Geographic Data Committee [FGDC], 1998, June) provides some ways to encode syntactic use metadata. The FGDC standard can accommodate almost any kind of metadata, including use metadata. But FGDC only specifies a framework for identifying and storing metadata, and is not a standard itself. Metadata requirements are spelled

out in discipline-specific profiles. Some of these profiles do spell out syntactic use metadata requirements (FGDC, 2002, October) while others are not so explicit (FGDC, 1999, October).

The Open Geographic Information System (OpenGIS) system of the Open GIS Consortium (OGC) publishes extensive standards for syntatic use metadata (see OGC (2003, March) for several examples). Since the OGC concentrates on creating the conditions for strict machine-to-machine interoperability with its metadata standards, this is to be expected. But the OGC metadata standards are largely aimed at geospatial data, whereas the OPeNDAP standards are meant to be more general.

Some of the work done on the semantic web (see McGuinness & van Harmelen (2003, August) for example) is or will become relevant to OPeNDAP. This work, including DAML+OIL and the OWL languages, may eventually provide ways to explicitly model the relationships between metadata described here. The next release of the OPeNDAP protocol will feature metadata information in XML form. Future OPeNDAP releases may be able to make good use of the semantic web work.

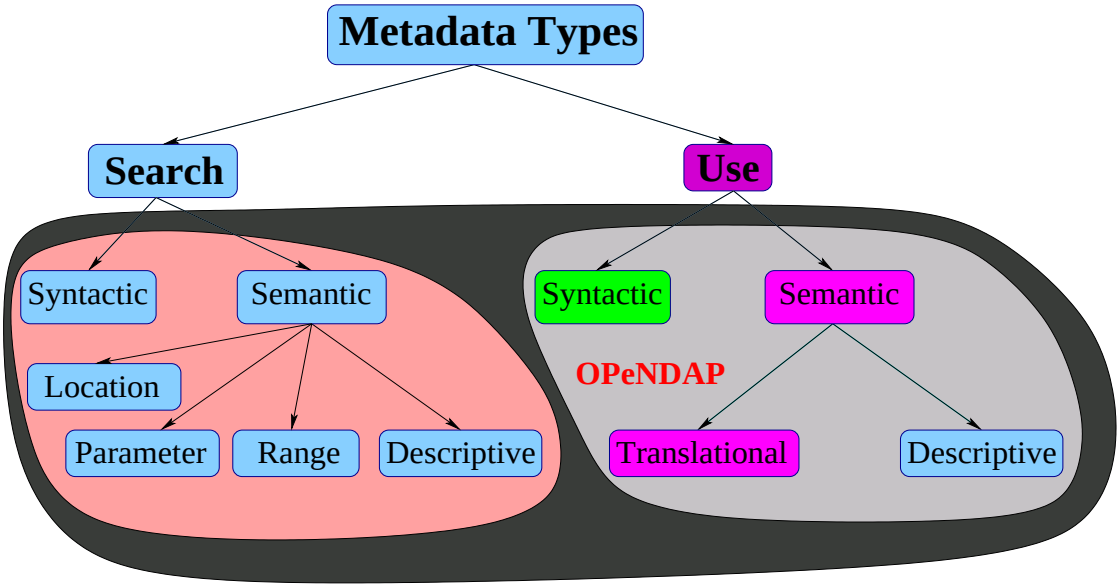

Figure 3. A schematic diagram of different metadata types.

5.1 Use metadata

Use metadata may be further subdivided into *translational* and *descriptive* use metadata. Translational use metadata refers to those metadata that define transformations between the data values received in an OPeNDAP client and values with semantic meaning to the user. One way to see this is that translational use metadata is the information that is required to properly label a plot of the data. Translational use metadata generally describes the operations that are performed on the data values or the names associated with them. A variable named "T" with a value of 223 might have two translations assigned to it. One might translate "T" into "temperature", or some other standard name, and the other might translate the number 223 into temperature with some known units. Descriptive use metadata might include the name of the sensor used to collect the data or the method of calibration.

Search metadata may also be further subdivided, in this case into parameter, range, location, and descriptive search metadata. Parameter search metadata contains the list of parameters or variables in the data set. This could be further subdivided into dependent and independent variables. Range search metadata

contains the ranges of variables within the data set. In most existing directory systems, only the ranges for time and space are included, but in future systems the range of all data values may be included as search metadata. Location search metadata is effectively a pointer to the data, a URL in the OPeNDAP case. Descriptive search metadata contains other information associated with the data set such as a generic description of the sensor used.

6 THE LAYER MODEL, AGAIN

Referring again to Figure 1, the metadata taxonomy introduced above makes explication of the layers simpler. The first structural layer refers to matching syntactic use metadata. Consider the two sea surface temperature records mentioned earlier: one is stored in a three-dimensional grid, with the dimensions of latitude, longitude and time, and the other is stored in many individual files, each containing a two-dimensional grid. These two data sets may be compatible on the format level, but not at the structural level – one consists of one 3-d object and the other of a number of 2-d objects. At the syntactic structural level, these data are all delivered as three dimensional objects. But even the fact that the two data sets have the same general structure does not mean that the structures are necessarily organized identically. For example, one data set could be organized as a longitude, latitude, time matrix while the other might be a time, longitude, latitude matrix. These data objects are syntactically similar, but not semantically similar. We therefore refer to the first structural level as the syntactic structural level. The OPeNDAP project supports a server capable of making the two data sources mentioned above appear to have a similar structure. (This is the Aggregation Server. See Caron & Sgouros, 2002.)

The semantic structural level addresses the semantic incompatibility of structures. In the example cited above, the order of dimensions within the arrays would be altered to one that is consistent for all similar data sets within the system.

It is important to keep in mind that in general each level in the system involves more potential modification of the data. At the format level, only the format is modified (as long as the data type required by the client is consistent with that delivered by the server). The syntactic structural level involves reorganization of the data objects, but not changes within an originating data object. The semantic structural level involves the reorganization of data within a data object.

The "Data modification" level of Figure 1 would involve modification to the actual data values (as opposed to the reorganization that characterizes the lower levels). All of the modifications desribed thus far leave, to the degree possible, the actual data values intact. In this level (or group of levels) the actual data values might be modified. For example, the temperature in two data sets might use different units, $^{\circ}$C in one case and $^{\circ}$F in another. One could imagine an operation that converted all data to the same units or all arrays into the same geographic projection.

The only operation that is required in addition to the ones listed above for complete interoperability is a transformation of the actual metadata. For example, temperature in two data sets might be identified with different names: "Temp" or "temperature". Modification to these metadata items are the ones that make up the top layer in Figure 1.

As should be apparent the requirements for interoperability in the system described above are on use metadata, not search metadata. In a top-to-bottom interoperable system there will be operations similar to some of those described above for search metadata.

7 CONCLUSION

Demands for semantic metadata are among the more contentious issues that one faces in the design of a distributed data system, where the data provided by the system is under the management of multiple

people or institutions. In our experience, users generally want more semantic metadata than providers are interested in generating. For example, the implementors of the GCMD (the users in this example) found that when too much search metadata was requested, many data set generators would simply not provide data set descriptions to the GCMD. The GCMD developed the "skinny DIF" to address this problem. The "skinny DIF" is a data set description that contains the minimum number of fields thought to be necessary to satisfy a rudimentary search. (GCMD, 2001)

Clearly, when defining the metadata requirements of any data system, a careful balance must be sought between satisfying the metadata needs of the user and minimizing the burden of producing metadata on the provider. Productive metadata discussions will therefore be a crucial part of the planning process and we believe that by focusing them on the framework described above we will make rapid progress to this end.

8 ACKNOWLEDGEMENTS

The work resulting in this manuscript was performed with support from the National Oceanographic Partnership Program (Grant #N000140010889) and the National Aeronautics and Space Administration (Grant #NCC5307). Salary support for P. Cornillon was provided by the State of Rhode Island and Providence Plantations.

REFERENCES

Caron, J., & Sgouros, T. (2002) OPeNDAP Aggregation Server Guide. Retrieved 25 March, 2003 from the Unidata website: http://unidata.ucar.edu/packages/dods.

Federal Geographic Data Committee (1998, June). FGDC-STD-001-1998 Content standard for digital geospatial metadata. Retrieved 25 March, 2003 from the Federal Geographic Data Committee website: http://www.fgdc.gov/metadata/contstan.html.

Federal Geographic Data Committee (1999, October). Biological data profile of the content standard for digital geospatial metadata, FGDC-STD-001.1-1999. Retrieved 25 March, 2003 from the Federal Geographic Data Committee website: http://www.fgdc.gov/standards/status/sub5 2.html.

Federal Geographic Data Committee (2002, October). FGDC-STD-012-2002 Content Standard for Digital Geospatial Metadata: Extensions for Remote Sensing Metadata. Retrieved 25 March, 2003 from the Federal Geographic Data Committee website: http://www.fgdc.gov/standards/status/csdgm.rs.ex.html.

Global Change Master Directory (GCMD) (2001) Directory Interchange Format (DIF) Writers Guide, Version 8.2001. Retrieved 15 December, 2002 from the National Aeronautics and Space Administration website: http://gcmd.nasa.gov/User/difguide.

International Electrotechnical Commission (1994) Information technology Open Systems Interconnection Basic Reference Model: The BasicModel ISO/IEC 7498 [Tech. report]. Retrieved 15 December, 2002 from the International Association for Computing Machinery website: http://www.acm.org/sigcomm/standards/iso stds/OSI MODEL/.

International Telecommunication Union (2002, July). Information technology Abstract SyntaxNotation One (ASN.1): Specification of basic notation. ITU-T Recommendation X.680 (Series X: Data Networks and Open System Communications). Retrieved 15 August, 2003 from the International Telecommunication Union website: http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf.

McGuinness, D. L., & van Harmelen, F. (2003, August). Web Ontology Language (OWL): Overview. W3C Candidate Recommendation. Retrieved 15 August, 2003 from the World Wide Web Consortium website: http://www.w3.org/TR/2003/CR-owl-features-20030818/.

Metadata Encoding and Transmission Standard (2002) Homepage of the Metadata Encoding and Transmission Standard. Retrieved 15 December, 2002 from the Library of Congress website: http://www.loc.gov/standards/mets.

National Centre for Supercomputing Applications (2001) HDF5 - A New Generation of HDF. Retrieved 15 August, 2003 from the National Centre for Supercomputing Applications website: http://hdf.ncsa.uiuc.edu/HDF5/.

Open GIS Consortium, Inc (2003, March). OpenGIS Implementation Specifications. Retrieved 25 March, 2003 from the Open GIS Consortium, Inc. website: http://www.opengis.org/techno/implementation.htm.

Rew, R. K., & Davis, D. P. (1990) NetCDF: An Interface for Scientific Data Access. IEEE Computer Graphics and Applications, 10(4), 7682.

Rew, R., Davis, G., & Emmerson, S. (1993, April). NetCDF Users Guide, Version 2.3. Boulder, Colorado: Unidata Program Center.

Sgouros, T. (1996-2002) DODS User Guide. Retrieved 15 August, 2003 from the World Wide Web: http://unidata.ucar.edu/packages/dods.

Stevens, W. R. (1999) UNIX Network Programming. (2nd ed.). Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Sun Microsystems, Inc. (1987, June). XDR: External Data Representation Standard. DARPA RFC 1014. San Jose, California.