

A Function-Based Data Model for Visualization

Lloyd A. Treinish
IBM Thomas J. Watson Research Center
Yorktown Heights, NY
lloyd@us.ibm.com

Abstract

The definition and implementation of coherent methods of representing data as a formal data model have been used to create general-purpose tools for visualization, computation and data management. Despite the relative success of these data models, there are gaps in their capabilities to support either specific classes of data or to map well to a user's problem domains. Therefore, a higher-level data model based upon a simple functional definition is proposed and tested, which leverages some of the facilities of extant implementations.

CR Descriptors: H.2.1 Data Models; H.2.8 Scientific Databases.

Additional Keywords: Visualization Systems.

1. INTRODUCTION

The instrumentation technology behind data generators in a myriad of disciplines is rapidly improving, typically much faster than the techniques available to manage and use the resultant data. Highly visible examples occur in projects such as NASA's Earth Observing System and the DOE Accelerated Strategic Computing Initiative (ASCI), which produce many TBs of data. For scientific visualization the role of data management can be expressed by the need for a class of data models that is matched to the structure of the data as well how such data may be used.

1.1 Conceptual Models in Visualization

To place data models in context consider the following taxonomy, which decomposes visualization into a set of conceptual models.

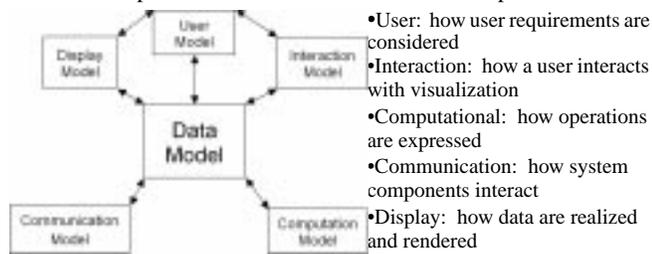


Figure 1. Conceptual Models in Visualization.

1.2 Data Models

A data model is a representation of data, that is how data are described (e.g., abstract data type) and how data are used (e.g., applications programming interface). To serve as a lexicon of data (i.e., a lingua franca for software and users), a data model must include formal definitions and algebra to express the organization and manipulation of data. In this context, visualization itself is not treated as special -- just another consumer and generator of data. Another way to view this idea is a layer that provides a logical link between the concepts that scientists use to think about their domain (e.g., particle trajectory, cerebral cortex shape, plasma temperature profile, or gene) and the underlying data from simulation, experiment or storage system. In particular, this layer provides tools that are common to all applications for data definition, metadata support, and query formulation and execution.

1.3 Related Work

There are many other efforts to develop data models related to visualization. While their results are widely used, they have limitations.

Common Data Format (CDF), developed at NASA/Goddard Space Flight Center initially in the mid-1980s, was one of the first imple-

mentations [8]. It is based upon the concept of providing low-level abstract support for scientific data that can be described by a multi-dimensional block structure. From that effort spawned the Unidata Program Center's netCDF, which is more focused on data transport [9]. Both CDF and netCDF support disk-based operations.

Another example is the Hierarchical Data Format (HDF) developed by the National Center for Supercomputing Applications [7]. HDF uses an extensible tagged file organization to provide access to basic data types like a raster image, multidimensional block, simple hierarchical collections, etc. Currently, all of HDF's data structures are memory resident. A lack of scalability in HDF is being addressed with a new implementation (HDF5) that offers improved performance at an array access level by enabling users to have more control over how data are stored and accessed.

VisAD (Visualization for Algorithm Development) was developed by the University of Wisconsin to provide interactive computation and visualization facilities derived from a set of abstract models. The VisAD data model assumes that data objects are approximations to mathematical objects [5]. It supports a wide range of data types associated with rich metadata, which are embodied as abstract data classes in Java. These facilities are coupled with a display model implemented in Java3D, which is defined by a set of mappings from primitive data types to primitive display types [6].

Data Explorer is an extended, client-server data-flow system for visualization that is built upon a data model, which supports generalized field representation with an API, high-level-language and visual program access [1]. The data model is derived from the mathematical notion of fiber bundles as an abstraction for scientific data management [3]. In Data Explorer, this idea is specialized and extended to incorporate localized, piecewise field descriptions, support compact representations that exploit regularity, and data-parallel execution. This permits consistent access to data independent of its underlying grid, type or hierarchical structure via an uniform abstraction to provide polymorphic functions. Data communication among subsequent operations is accomplished by passing pointers, and sharing of these structures among such operations is supported [4]. Currently, the physical disk-based format, dx, provides only simple sequential access.

From the aforementioned efforts, ASCI, as part of its overall data management effort, is developing a very comprehensive data model to ensure coverage of many different representations of simulation data. The main entities in the model are computational meshes and the simulation variables related to them. This data model also uses the notion of fiber bundle sections for the mapping between topological spaces. It then introduces the idea of cell complexes, which are structures that tile a physical space with geometric cells that share common faces as a metaphor for computational meshes. Under the ASCI program, there is on-going development for this model and as well as tools that utilize it. The model is based upon a three-layer approach of providing data structures for array and table access, then fiber bundles to provide basic mappings, and finally a mesh/field level to provide field as well as cell complex access. HDF 5 is being used to provide the underlying access to storage [2].

In addition, there are other classes of data models. These include, for example, geographic information systems (a set of static, two-dimensional spatial layers), mechanical computer-aided design (static, 3d hierarchies and non-mesh representations), and the relational data model (tables of discrete non-spatially-oriented values).

2. A FUNCTION-BASED UNIFIED DATA MODEL

Despite the capabilities of these scientific data models, there remain a few areas which are not adequately addressed. This ranges from the representation of other data types such as ordered structures (e.g., molecular models), tables and relations, highly irregular, inconsistent or non-spatial sampling (i.e., observations), and aggregation of disparate types. As is often the case, highly generic approaches are often difficult for many scientists to adapt to their own problem domains. Therefore, another aspect that needs addressing is more direct mapping at a user level. An approach complementary to the aforementioned efforts is taken to resolve these limitations.

To begin it is necessary to look at the fundamental organization and definition of data. Any data set may be considered as a single or multi-valued function of one or more independent variable(s) called dimensions, enumerated from 1 to j . Such dimensions may be space (length, width, height), time, energy, etc. A parameter may have more than one value, which is characterized by tensor rank, i , the number of values per dependent variable. The number of elements in a particular parameter is j^i , which can be generalized as a set of tuples. The function(s) composing a data set really are dependent variable(s). Thus, data or \mathbf{D} implies a parameter or field of one or more (dependent) values that is a function of one or more (independent) variables,

$$\mathbf{D} = [y_1, y_2, \dots, y_i] = [f_1(x_1, x_2, \dots, x_j) \quad (1)$$

$$f_2(x_1, x_2, \dots, x_j)$$

$$\vdots$$

$$f_i(x_1, x_2, \dots, x_j)]$$

These functions are continuous in nature, but sampled or discretized in a fashion often dictated by the specific computations to be performed. Operations imply a process of transformation between different functions of this class, whether it is solved as a set of partial differential equations that define flow of heat or generating pixels as a rendering of some geometry.

2.1 Functional Mapping

Consider $\mathbf{F}(\mathbf{D})$, where \mathbf{D} are data and \mathbf{F} is some computation, which may include an operation in a visualization system such as realization or transformation. \mathbf{D} can be extended beyond mesh sampling or aggregation by examining topological mapping, α ,

$$\alpha : \mathbf{T}_1 \rightarrow \mathbf{T}_2 \quad (2)$$

where α is a mapping between two topological spaces (e.g., a visualization operation). If both α and α^{-1} are continuous then α is considered *homeomorphic*. Commonly, there may be more than one mapping such that

$$\alpha_1 : \mathbf{X} \rightarrow \mathbf{Y} \text{ and } \alpha_2 : \mathbf{X} \rightarrow \mathbf{Y} \quad (3)$$

If α_1 can be deformed to α_2 then α_1 is considered *homotopic* to α_2 . Since such deformable mappings occur often in visualization, this is a useful classifier for $\mathbf{F}(\mathbf{D})$ (see below). Thus,

$$\mathbf{F} : \mathbf{X} \times [0, 1] \rightarrow \mathbf{Y} \quad (4)$$

If \mathbf{F} is continuous such that $\mathbf{F}(x, 0) = \alpha_1$, $\mathbf{F}(x, 1) = \alpha_2$ and as the real variable, t , in $\mathbf{F}(x, t)$ varies continuously from $[0, 1]$, α_1 is deformed continuously into α_2

As a result, homeomorphism generates equivalence classes whose members are topological spaces while homotopy generates equivalence classes whose members are continuous maps. Hence, homotopy equivalent classes are topological invariants of \mathbf{X} and \mathbf{Y} which enables one to vary \mathbf{X} or \mathbf{Y} through a family of spaces, $\mathbf{C}(\mathbf{X}, \mathbf{Y})$, as a collection of valid visualization mappings.

Reconsider $\mathbf{F}(\mathbf{D})$ such that $\mathbf{F} : \mathbf{X} \times [0, 1] \rightarrow \mathbf{Y}$, then:

- \mathbf{X} is the space where \mathbf{D} are known
- \mathbf{Y} is the space for the (new) mapping
- \mathbf{X} and \mathbf{Y} are homotopic
- \mathbf{X} and \mathbf{Y} imply points in space with some coordinates with respect to a reference system -- in theory
- $\mathbf{F}(\mathbf{D}, t)$ implies sampling of continuous domain -- in practice
- \mathbf{D} is a continuous function, but stored as a discrete set in $\mathbf{C}(\mathbf{X}, \mathbf{Y})$ on which $\mathbf{F}(\mathbf{D}, t)$ varies

But in practice, there are five cases for \mathbf{F}

1. If \mathbf{X} and \mathbf{Y} are the same, then provide results unchanged (i.e., available "samples")
2. If \mathbf{X} and \mathbf{Y} are equivalent classes discretely, then map between the spaces, $\mathbf{X} \rightarrow \mathbf{Y}$ (e.g., cartographic warping -- homotopy)
3. If \mathbf{X} and \mathbf{Y} are not equivalent classes discrete, then construct a mapping between the spaces (e.g., interpolate between spaces)
4. If \mathbf{X} is unknown (i.e., scattered data), then construct a mapping to \mathbf{Y} (e.g., impose an \mathbf{X} and interpolate) or provide discrete values
5. If \mathbf{X} and \mathbf{F} are unknown (i.e., tabular data), then provide discrete values

The relationship between these cases is shown in Table 1, which introduces a layered structure. Essentially these layers moving from the bottom up define semantics, access and interface, listed on the left to accommodate more complex data, where the functional access is at the top. Where the five cases fit is shown at the far left.

Cases	Views	Layers
1, 2, 3, 4, 5	• "Real world" • User/Domain	Application (Functions)
1, 4, 5	• Complexes • Hierarchies • Series	Aggregation Specialization
1, 4, 5	• Semantics for arrays • Samples or tables	Field (e.g., Fiber Bundle)
5	• Virtual organization • Dimensionality, rank & attributes	Multi-Dimensional Arrays
	• Physical access • Distribution and locality	I/O, Communication, Physical Storage

Table 1. Layers of Function-Based Data Model.

This effort has focused mostly on the upper two layers as shown in Table 1 to provide a conceptual specification. An abstraction at each level is presented in a simple fashion, but efficiencies in implementation can be addressed at the lower levels. As an illustration, consider the following two figures, which show some of the underlying components and their relationships.

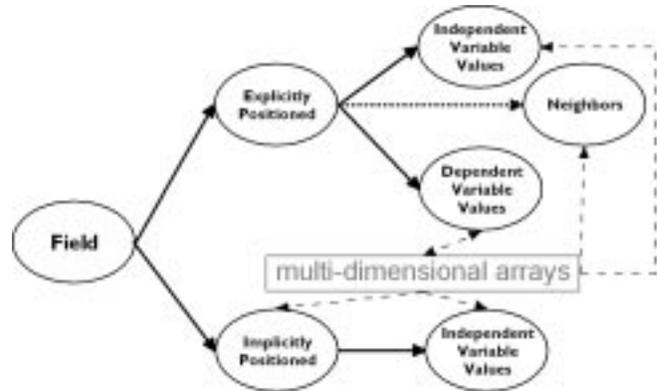


Figure 2. Taxonomy of the Field Layer.

Figure 2 illustrates one taxonomy of the field layer, which can be decomposed into meshes on which the data are sampled that are either implicitly positioned (i.e., regular) or explicitly positioned

(i.e., irregular). In the latter case, the mesh may be topologically regular, in which only the locations of the sample points need be specified, or topologically irregular, in which connectivity information is required. In all cases, the actual data values on the sample points are defined. All of this information is stored as a collection of multidimensional arrays (i.e., the array layer). The metadata associated with the arrays provides the semantics to define a field.

Figure 3 shows sample taxonomies for the aggregation layer, which describes classes of data that can be broken down into simple fields. They may range from a (time) series (a sequence of instances of a field) to cases which can be spatially decomposed into sets of sub-fields such as a multizone grid or a collection of spatial partitions for parallel processing. Adding hierarchy enables a description of an adaptive mesh. Some data may be sampled over a topologically complex mesh. In this case, it may be easier to decompose it into a set of simpler meshes, each of which is of the same class.

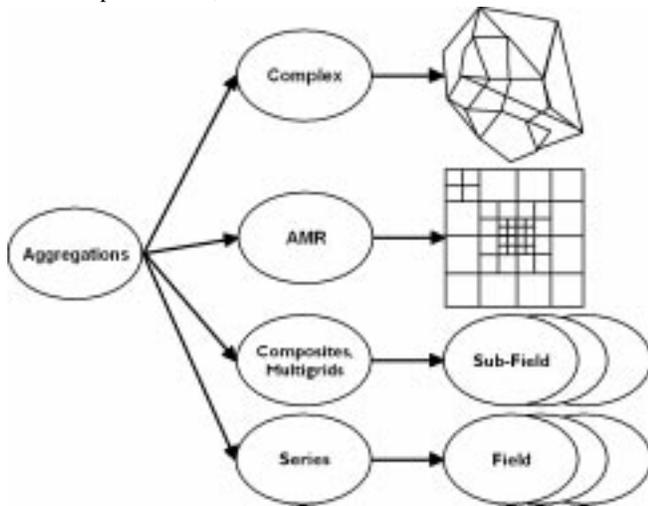


Figure 3. Taxonomy of Aggregation Layer.

3. EXAMPLES

To demonstrate the flexibility of this approach and to test the applicability of this model, a number of applications have been identified for each of the five cases. Only a handful are shown herein. Visualization and interaction are used to verify the decomposition and mapping. The data model and functionality of Data Explorer is extended to provide the function-based interface. Some of these extensions are usable across different cases and applications.

3.1 Case 2

Figure 4 shows different coordinate systems and sampling for a single scalar data set, which is total column ozone in the earth's atmosphere that is irregularly sampled on a two-dimensional manifold over time. The image only shows a single time step out of a long series of daily observations. Essentially, two equivalent class mappings are presented via generalized cartographic projections to three dimensions, which is a Case 2 example. The window in the upper part of the center of the figure, shows the full data set, mapped to color, opacity and radial deformation. The window at the lower right shows only half (the southern hemisphere). In both cases, other data are registered in the final rendering and the same mapping applied for annotation purposes (topography and coastline data, respectively). The field is discretely sampled in one dimension and shown as a plot. The remapping is packaged as a new function to create this application. The user only needs to specify the field representing the original data and the new space.

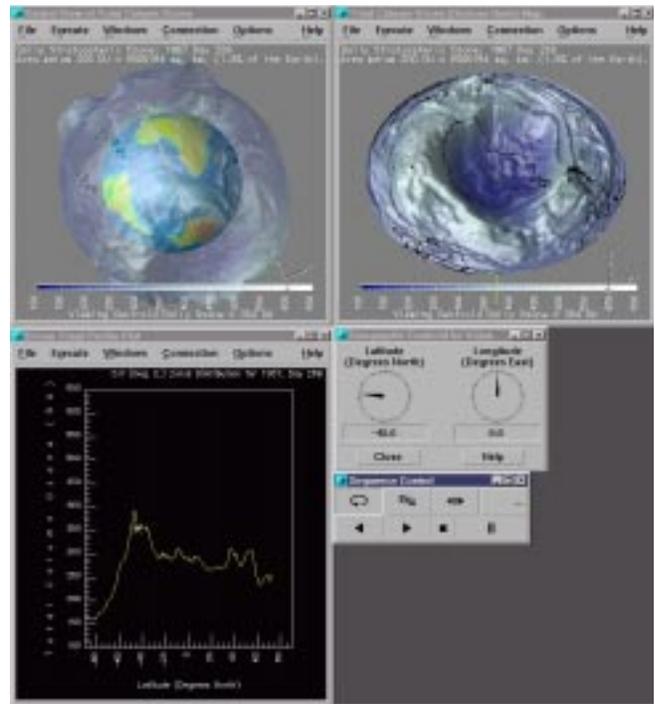


Figure 4. Different Spaces and Sampling.

3.2 Case 2 and Case 4

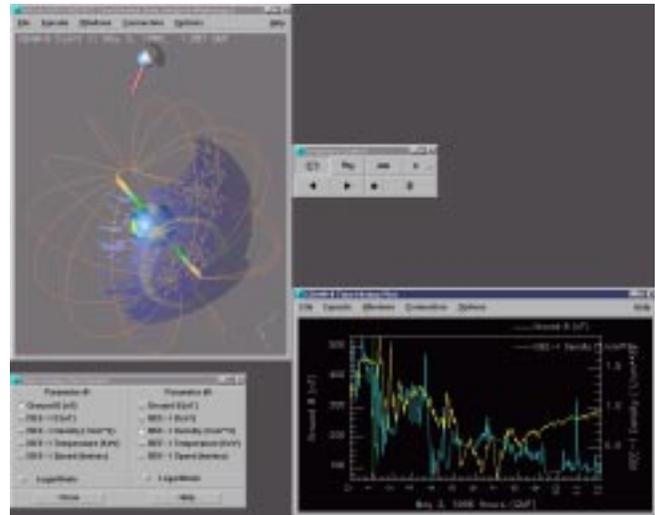


Figure 5. Fusion of Irregular Samplings in Space and Time.

Figure 5 shows several data sets, both scalar and vector, that are sampled irregularly and differently on distinct three-dimensional and two-dimensional manifolds with various irregular sampling in time. The image only shows a single time step out of a series for each of the data sets, ultraviolet intensity, proton density, temperature and speed, topography, and three different magnetic field data sets. Essentially, one equivalent class mapping is presented via a generalized cartographic projection to three dimensions. Unlike the previous example, these may be either Case 2 or Case 4 because some of the data sets are discretely sampled with no information on the relationship between the samples. Each variable is processed separately by using the same remapping function, with which the user specifies the original data and the new (spherical) space. In the application that utilizes this function, several choices of realization mappings are offered for the different data sets that are registered in the final rendering in the remapped (earth-centered, spherical) coor-

dinate system. In addition, two of the data sets are plotted at the lower right as a function of their original time sampling.

3.1 Case 1 and Case 3

Figure 6 illustrates what appears to be conventional representations of two data sets sampled irregularly on a three-dimensional manifold over time. The image only shows a single time step out of a series of computed results. But there are some important distinctions. The first is that the domain on which the data are defined is symmetric, with only one-fourth being specified by the manifold. The second is that while one of the fields is a traditional interval data set (density), the other (material) is not. Material is categorical, specifically nominal (i.e., there is a “name” associated with each sample space that may not be related to other sample points). Therefore, for density this is Case 1 while for material it is Case 3. Hence, access to the categorical data is packaged as a new function to provide the equivalent of traditional field as well as calculation of a derived one for sample points where the material is not homogeneous. In both cases, the relationship between the sampling manifold and the domain is hidden within the function. The application shown in the image allows the user to specify various realizations and interact with them. In addition, the fields may be queried by invoking the new functions directly. The results are shown both as values in the three-dimensional scene as well as plotted in the lower right. In addition, the material function enables remapping between those data and density, so that density surfaces can be color mapped by material or the density of specific materials can be illustrated.

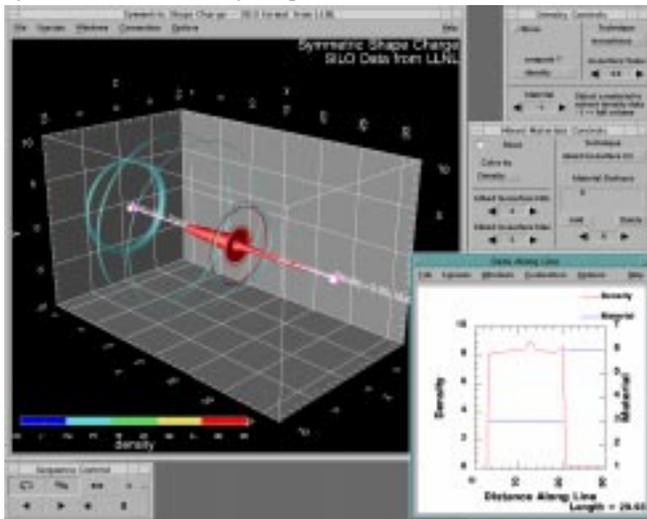


Figure 6. Sampling of Interval and Nominal Data.

3.1 Case 4 and Case 5

Figure 7 shows three-dimensional representations of several variables. They are from a set of some 60 parameters provided as a table from a data base of credit card transactions. Effectively these data are “sampled” by record from about 48,000 instances. These data are all categorical, either ordinal or nominal. Normally, this would be Case 5. However, some of the ordinal parameters may be interval-like. Hence, access is packaged as a new function, which provides the equivalent of traditional field (Case 4) via Delauney tetrahedralization. In this case, the user interactively chooses the mapping of parameters from the complete set to three spatial dimensions to form the basis of the independent variables. In this application, up to three additional variables may be selected as being dependent on the first three for creating planar mappings, which are then pseudo-colored contoured. In addition, the original data may be queried.

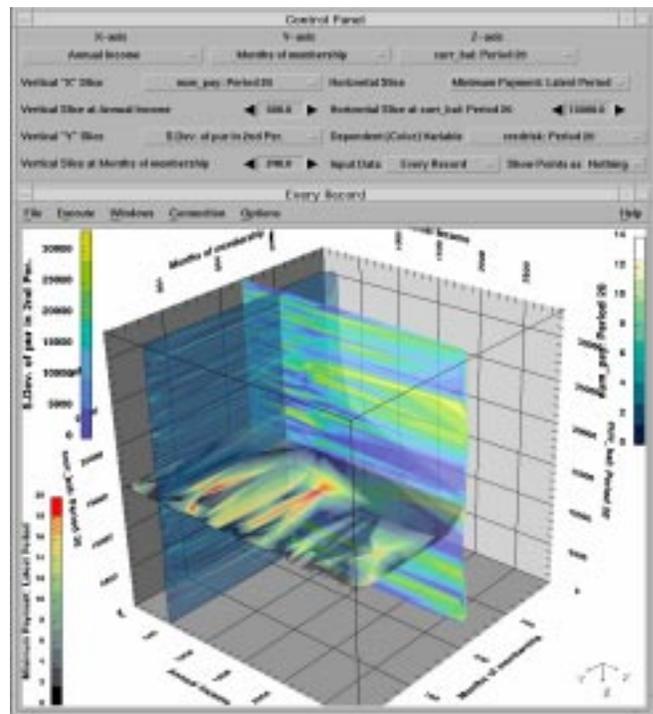


Figure 7. “Resampling” of Tabular Data.

4. CONCLUSIONS

Data models can hide the complexity of underlying computational systems for simulation, analysis and visualization by providing a common mechanism for access, utilization and interchange. A study of data model efforts and limitations in how tools that utilize them map to end user requirements has led to a taxonomy of conceptual models for visualization. In turn, this taxonomy has enabled a simple formalism to define a set of higher-level functions that map directly to how data may be used in visualization. To test these ideas, a collection of higher-order functions have been implemented by leveraging the capabilities of a lower-level data model. This demonstrates the feasibility and potential applicability of this idea. Current plans include the further extension and application of these higher-order functions and continuing to refine the formalism.

5. REFERENCES

- [1] Abram, G. and L. Treinish. *An Extended Data Flow Architecture for Data Analysis and Visualization*. **Proceedings IEEE Visualization '95**, pp. 263-269, October 1995.
- [2] Ambrosiano, J. **ASCI Common Data Model (CDM)**. Los Alamos National Laboratory, <http://www.ca.sandia.gov/asci-sdm/cgi-bin/sdm-framedisplay.cgi/asci-sdm/CDMlib.html>
- [3] Butler, D. M. and M. H. Pendley. *A Visualization Model Based on the Mathematics of Fiber Bundles*. **Computers in Physics**, 3, n.5, September/October 1989.
- [4] Haber, R., B. Lucas and N. Collins. *A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids*. **Proceedings IEEE Visualization '91**, pp. 298-305, October 1991.
- [5] Hibbard, W., C. R. Dyer and B. Paul. *A Lattice Model for Data Display*. **Proceedings IEEE Visualization '94**, pp. 310-317, October 1994.
- [6] Hibbard, W. *VisAD: Connecting people to computations and people to people*. **Computer Graphics**, 32, n. 3, 1998.
- [7] NCSA. **Hierarchical Data Format (HDF) Reference Manual**, V. 4.1. University of Illinois, June 1998.
- [8] National Space Science Data Center. **CDF User's Guide**, V. 2.6. NASA/Goddard Space Flight Center, March 1996.
- [9] Unidata Program Center. **NetCDF User's Guide**, V. 3.4, March 1998.