LA-UR- 99-722

Approved for public release; distribution is unlimited.

CONF-990707--

Title: Development of a Common Data Model for Scientific Simulations

Author(s): John Ambrosiano, Los Alamos National Laboratory
David M. Butler, Limit Point Systems, Inc.
Celeste Matarazzo, Lawrence Livermore National Laboratory
Mark Miller, Lawrence Livermore National Laboratory
Larry Schoof, Sandia National Laboratory

Submitted to: 11th International Conference on Scientific and Statistical Database Management
Cleveland, OH
July 28-30, 1999

MASTER

Los Alamos
NATIONAL LABORATORY

DISCLAIMER

Development of a Common Data Model for Scientific Simulations

John Ambrosiano, Los Alamos National Laboratory
David M. Butler, Limit Point Systems, Inc.
Celeste Matarazzo, Mark Miller, Lawrence Livermore National Laboratory
Larry Schoof, Sandia National Laboratory

Abstract

The problem of sharing data among scientific simulation models is a difficult and persistent one. Computational scientists employ an enormous variety of discrete approximations in modeling physical processes on computers. Problems occur when models based on different representations are required to exchange data with one another, or with some other software package. Within the DOE's Accelerated Strategic Computing Initiative (ASCI), a cross-disciplinary group called the Data Models and Formats (DMF) group, has been working to develop a common data model. The current model is comprised of several layers of increasing semantic complexity. One of these layers is an abstract model based on set theory and topology called the fiber bundle kernel (FBK). This layer provides the flexibility needed to describe a wide range of mesh-approximated functions as well as other entities. This paper briefly describes the ASCI common data model, its mathematical basis, and ASCI prototype development. These prototypes include an object-oriented data management library developed at Los Alamos called the Common Data Model Library or CDMlib, the Vector Bundle API from the Lawrence Livermore Laboratory, and the DMF API from Sandia National Laboratory.

1. Introduction

Computational scientists employ an enormous variety of discrete approximations in modeling physical processes on computers. Within a given model, discrete representations and numerical algorithms are chosen to achieve the greatest fidelity and computational efficiency.

The range of computational representations in simulation is extremely broad. Some simulation variables are finite and discrete, while others are approximations of functions on continuous spaces. In the latter case, continuous problem domains must be first discretized. Discretization often takes the form of a grid of sample points or alternatively a decomposition into cells or elements. However, other representations may also be

employed. These may include representations such as spectral or pseudospectral expansions, e.g. Fourier or wavelet transforms.

Serious problems can occur when models based on different representations are required to exchange data, or when input and output data related to simulations must be interpreted by other applications. For example, initial problem geometry and grid might be generated using commercial CAD and grid generation software. This information must then be communicated to the simulation code. A similar problem may occur in transferring simulation output to commercial visualization or statistical analysis software. Data exchange problems can also arise when individual sub-models are incorporated into larger, collaborative, multi-discipline frameworks. The trend toward building large-scale, high-performance, multi-discipline problem-solving environments (PSEs) has promoted data exchange problems to near-crisis levels.

Examples of ambitious, multi-discipline PSEs are to be found in the Environmental Protection Agency's emerging framework for integrated air and water quality models [Novak, et al, 1998] as well as within the Department of Energy's Accelerated Strategic Computing Initiative (ASCI) [Larzelere, 1998]. The aim of the EPA framework is to enable more realistic simulations of coupled ecosystems. The goal of the ASCI program is to support science-based stewardship of the nation's nuclear stockpile under conditions of a comprehensive nuclear test ban.

The essential problem of sharing simulation data can be stated as follows. Simulations are intended to be models of physical reality that mimic some essential features of the real system so as to predict aspects of its behavior. Data sharing can be challenging for many reasons, but two very important ones are:

- The complexity of a real system is many times greater than that of a computer program. Real systems are often infinite and continuous, whereas a computer simulation is finite and discrete.
- The real system is at least two levels of abstraction removed from the computer program. The first level being the step from reality to a mathematical model, and the second being a step from mathematics to computer algorithms and data structures.

The second of these factors accounts for one of the main problems in communicating simulation data from one model or application to another. In the transformation from physical system to mathematical model, and the subsequent transformation from mathematics to data, the semantic links are often lost. Therefore, the application receiving the data often has no context in which to interpret the numbers and other data items given to it.

Within the DOE ASCI program, a cross-disciplinary, inter-organizational group has been tasked with developing solutions to these problems as part of ASCI's Scientific Data Management (SDM) effort. This group, called the Data Models and Formats Group (DMF), has been working to develop a common data model and supporting software. In addition to the complexity and diversity of representations that plague every ambitious PSE effort, the ASCI program must also contend with unprecedented scale and performance requirements. ASCI applications will routinely compute hundreds of simulation variables comprised of literally billions of computational elements. This puts ASCI application output squarely in the terascale range. Data modeling and tool development must therefore not only support a broad range of model representations, but must integrate them with efficient parallel I/O and communication systems of tremendous capacity and throughput.

This paper briefly describes the ASCI common data model, its mathematical basis, and ASCI prototype development. These prototypes include an object-oriented data management library developed at Los Alamos called the Common Data Model Library or CDMlib, the Vector Bundle API from the Lawrence Livermore Laboratory, and the DMF API from Sandia National Laboratory.

2. The ASCI Common Data Model

A useful data model must take into account the relationship between physical systems and mathematics as well as the relationship between common mathematical entities in simulations and discrete representations of them employed in computer algorithms. In this way the semantic connections can be preserved and exploited for data transformation and management. To satisfy the requirements of the ASCI program, such a model must be well integrated with parallel I/O and communication.

2.1 Semantic Levels of the Data Model

The current model is comprised of three levels of increasing semantic complexity. A simplified view of the DMF conceptual architecture is shown in Figure 1. Each layer is supported by abstractions in the layers below.

The top layer contains entities commonly employed in simulations and therefore easily apprehended by users. These would include, for example, various categories of meshes and mesh-based variables. Descriptions of parts and other geometric objects would also be supported here. Simple finite sets, lists, and maps can be made available as well, as can ordinary scalar variables and other generally useful features such as descriptive text fields. To help users manage collections of objects at this level, we would

likely include a simple hierarchical aggregation mechanism such as an object set or object group class whose contents can include other object sets or groups.

In the middle is a collection of abstractions drawn from fundamental principles in set theory and topology called the Fiber Bundle Kernel or FBK. Its role is much the same as that of the widely used relational data model and has some features in common with it. The name derives from a series of mathematically motivated data models first proposed for scientific visualization [Butler and Pendley, 1989; Butler and Bryson, 1992]. In addition to the well-known Cartesian product sets of the relational model, the FBK introduces important new structures needed to capture the topology of simulation problem spaces and the often complicated mappings defined on them. The existence of this layer provides a powerful set of components for reuse in the layers above, and allows for optimization of storage and retrieval over a very broad range of circumstances.

The bottom layer of the model contains the abstractions that are closest to actual computer data and record management. It provides abstractions of atomic data types, data structures and simple containers. It can also support a persistent object model based on these. Standardized communication protocols and I/O operations live here.

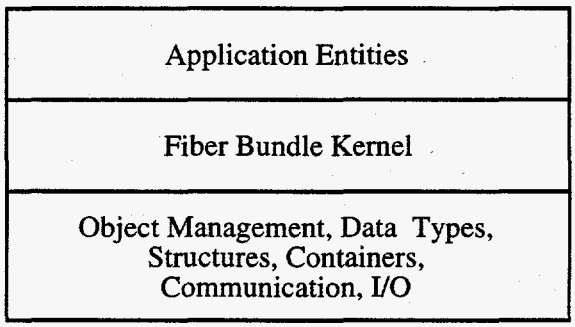

Figure 1. Conceptual layers in the ASCI Common Data Model.

In the current software design, HDF5 has been adopted as the primary I/O subsystem [HDF5, 1998]. HDF5 is a file format and I/O library for high performance computing and scientific data management developed by the National Center for Supercomputing Applications. HDF5 provides cross-platform portability of the data and its access methods within a simple self-describing, array-oriented data model that is easy for scientists and software developers to use. HDF5 is a newly redesigned version of the HDF format and library [HDF, 1998] that still enjoys widespread support in a scientific computing community that includes NASA, and the Department of Energy as well as scientific visualization software vendors. Two special features of HDF5 that are of particular importance for ASCI are its very large capacity, and its direct support of parallel I/O functionality via the MPI I/O standard [MPI2, 1997].

2.2 The Fiber Bundle Kernel

In this section we describe the mathematical elements of the Fiber Bundle Kernel. The key concepts are:

- Fiber bundles and manifolds
- Cells and cell complexes
- Sets and families of subsets
- Lattices based on subset inclusion
- Families of functions on subsets of a topological space (sheaves) and their representation spaces

Many of these mathematical concepts were introduced to the DMF effort by Butler [Butler, 1999]. The proposal to make cell complexes an integral part of the model derives from earlier applications to environmental modeling [Ambrosiano, et al, 1996].

Fiber bundles. Quantities of various kinds (e.g. temperature, pressure) that are functions of the problem domain are key features of simulations. Therefore a good encompassing model for them is important. In order to allow functions to be defined on subsets of the problem, and to allow the possibility that multiple values may exist for the same point where subsets overlap, the usual idea of a function must be broadened. This generalization is a called fiber bundle. Figure 2 illustrates a fiber bundle's construction.

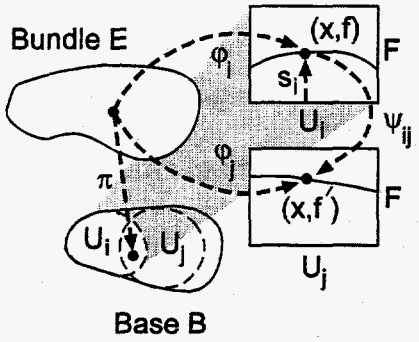

Figure 2. A fiber bundle.

Fiber bundles are able to represent globally complex mappings on topological spaces in a way that is locally simple. Each subset U_i of the domain space B (called the base) has a local representation of the mapping that resembles an ordinary function. This view, called a section, is a map from a point in the subset to a corresponding point the Cartesian product of the subset and the range. Such a view is provided by a topology-preserving map (homeomorphism) φ called the local trivialization. The standard term for the range set in this context is the "fiber." Where multiple sections coincide, a mapping ψ_{ij} exists from a fiber value in one section to a different fiber value in the other. These mappings form a group called the structure group. There is an additional ingredient, namely a projection operation π to take us back from anywhere in the bundle to a unique point in the base. That there exists a unique inverse image in the base for any point in the fiber

bundle, is the feature that most resembles the functions to which we are accustomed. The collection of local trivializations, sections, projection, and structure group mappings, is what we call a fiber bundle.

While fiber bundles seem exotic, they occur with surprising regularity in practical modeling applications, particularly in parallel computing. When a problem is decomposed into subdomains for parallel computation, the subdomains often carry with them pieces of the adjoining problem space in the form of ghost points or ghost cells. The collections of data values computed on separate subdomains resemble the sections of a fiber bundle. In overlapping regions, prior to synchronization, there may be multiple values of simulation variables for the same point in the problem space. Thus parallel application builders already work with fiber bundles every day. Another common example is a mesh-based problem containing separate materials with interfaces at mesh cell boundaries. Each material region is a subset, and variables like density are actually separate sections whose values are likely to be different on interface cells (e.g. faces) shared by adjoining regions. The fiber bundle construction provides the flexibility to accommodate multiple function values at these interfaces, whereas the usual definition of a function does not.

Manifolds. Many readers are already familiar with manifolds. A manifold is another example of a complex entity described in terms of locally simple views as shown in Figure 3. Here each subset S_i of a topological space is allowed to have its own local coordinate map φ_i that makes the subset seem to be a simple Euclidean space. Where two subsets S_i and S_j overlap, there exists a transformation T_{ij} of coordinates from one local map to the next so that local coordinates can transition smoothly from one subset to the

next. The coordinate map associated with a subset is called a chart, and the collection of all the charts is called an atlas.

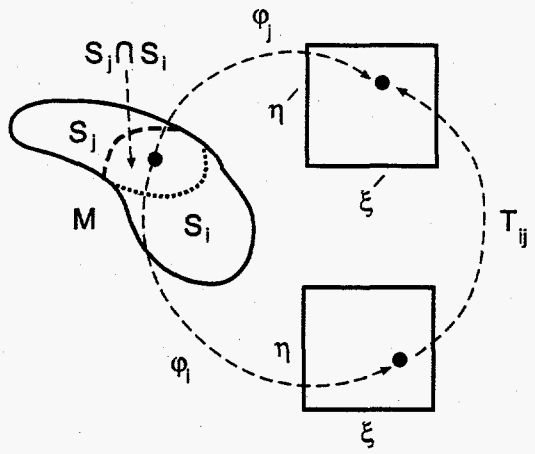

Figure 3. A manifold M with its coordinate maps and transformations.

Manifolds are more common in computations than one might think. Aerodynamic modeling is a prime example. In simulating flow over an airplane's surface, it may be very difficult to construct one mesh and one set of mesh coordinates that covers the problem domain. Instead one often covers the surface of the airplane with overlapping regular meshes, each with its own local (logical) coordinates.

Cells and cell complexes. The idea of cells arises in the context of dividing a toplogical space into pieces. Such an idea is natural in computational modeling where the problem space is often divided into subdomains. A cell is a closed subset of a Euclidean space homeomorphic to the closure of a subset of some topological space [Thurston, 1997]. The homeomorphisms that define cells are often chosen for convenience to map the subsets into polyhedrons.

Cells are characterized by their dimension. In three dimensions a zero-cell would be called a vertex, a one-cell an edge, and a two-cell a face. A cell complex is a partition of

the space into cells in such a way that the boundary of any n-cell is contained in the union of all cells of dimension less than n. The partition of a portion of the Euclidean space R^2 into polygons is illustrated in Figure 4.

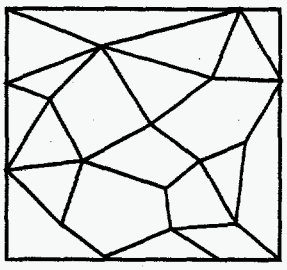

Figure 4. A cell complex of polygons in R^2

Each two-cell in the figure is bounded by one-cells, and adjacent two-cells have common boundaries. The one-cells are themselves bounded by zero-cells. The total collection of all these cells (of dimension, zero, one, and two) constitutes the cell complex. Cell complexes discretize a continuous problem domain, having an infinite number of points and an arbitrary collection of open subsets, into a finite collection of closed subsets related to one another in an explicit fashion. The intersections of cells in a cell complex are contained in the cell complex by construction, and the unions of cells are simple to describe. Cell complexes are, by construction, manifolds whose structure is straightforward and explicit.

Cell complexes are a good model for many discretizations occurring in practice. Computational meshes are easy to interpret in terms of a cell complex model, and many B-rep descriptions of geometry based on polygonal facets or NURB patches can also be interpreted in this way. There is also a close relationship between finite element discretizations and cell complexes. However, the cell complex model is much broader.

Subsets and Inclusion Lattices. A family of subsets is an instance of a partially ordered set or poset. A poset is any set for which an ordering relation among the elements exists. The ordering relationship among elements of a family of subsets is subset inclusion. Let S_1 and S_2 be subsets of a set X. We say that $S_1 \geq S_2$ if $S_1 \supseteq S_2$. A family of subsets (or the elements of any poset) can be arranged in a directed graph [Szasz, 1963]. Such a graph is constructed by assigning a node to every element and connecting nodes by arrows using the following rule. If an element is larger than another element, an arrow connects the smaller element to the larger one. An arrow connects two elements x and y if and only if there is no other element z such that $x \geq z \geq y$. Figure 2 shows the poset graph of a partially ordered set of elements $\{a,b,c,d,e,f\}$. From the graph we can see that the set has a maximum equal to a, but no single minimum. Also some elements such as d and f are not directly comparable, although d and f can each be said to be smaller than b. When a poset is constructed from a family of subsets, the entire set and the empty or null set ϕ are both included in the graph at the top and bottom respectively. That is, all sets are less than or equal to the entire set, and all sets are greater than the null set.

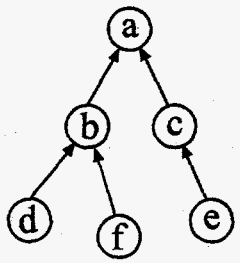

Figure 5. A poset graph for the set {a,b,c,d,e,f}.

Posets are related to a class of algebraic structures called lattices. For this reason we call the poset graphs for families of subsets inclusion lattices. The inclusion lattice construction provides a practical as well as a mathematically sound strategy for

representing the topology of a space in terms of a covering of subsets. For example, the subsets represented by a cell complex can be arranged in a graph such as Figure 5. Once this is done, it is straightforward to exploit knowledge of cell unions, intersections, and inclusion relations by simply traversing the graph. There is also another useful feature. Members of a lattice that cannot be expressed as the union or join of other members are called the join-irreducible members. These constitute a basis for representing other members according to the Birkhoff representation theorem [Birkhoff, 1967]. In other words, there exists a minimal set of elements for expressing topological relationships in lattice form.

Presheaves, Sheaves, and Section Representation Spaces. In the approximation of functions or fiber bundles on continuous domains, two discretization steps can be identified. The first is the decomposition of the space into the cells of a cell complex. Once this is done, a discrete manifold structure exists by construction. On this manifold one can define a single global coordinate space or, if one wishes, an atlas of local coordinate maps. This structure also supports the definition of either a single global function, or alternatively, a finite set of fiber bundle sections with the cell complex as the base. Sections can in principle be defined on any cell of any dimension. For example, in the cell complex of Figure 4, sections might be defined on each of the polygons as well as each of the edges and vertices. Each section is potentially continuous.

A second discretization is necessary to approximate continuous sections by a finite set of values. For example, a section defined on a polygon of Figure 4 could be approximated by a constant function. The value of the function everywhere in the cell would in that case require the specification of only one number.

This construction can be generalized by allowing each subset of the domain (e.g. as represented by a cell) to be associated with a family of sections restricted to the subset. A mapping from subsets of a topological to a family of sections is technically known as a presheaf of sets. Additional properties, having to do with how such a family behaves when restricted to open coverings of each subset, may qualify some presheaves as sheaves of sets [Tennison, 1975]. In practical terms, a (pre)sheaf construction allows each cell in the model to be associated with a prescription that identifies a particular family of sections. If this prescription is expressed in terms of a discrete set of values (as in our constant function example), then a second level of discretization is obtained. We call these discrete values the "degrees of freedom" for the section family and their range (e.g., type, allowed values) defines a Cartesian product space called the "section representation space." This space is different in general than the fiber of the section being approximated. Choosing specific values from the section representation space selects a particular function representation from the family of sections.

In spite of the unfamiliar terminology, this construction should be familiar to modelers. In mixed finite element approximations, cells of different types, such as quads and triangles are associated with different finite element basis functions. These are really families of functions. The integration points of finite elements correspond to our degrees of freedom. It is important to understand that the generalization discussed here, while covering familiar examples like finite elements, is actually much broader and can accommodate virtually any computational representation including spectral and pseudospectral expansions. The sheaf, introduced in the DMF model by Butler, is the most recent data model addition and one that provides a *bona fide* mathematical

interpretation for constructions that have been routinely employed in computational settings. We have recently learned that sheaves have lately been applied independently by Oliviera in the field of bio-informatics [Oliviera, 1999].

Integrating Mathematical Concepts in the DMF Model. What we have called the DMF Fiber Bundle Kernel is more than a miscellaneous collection of interesting mathematical concepts. It is a synthesis of particular mathematical elements into an integrated data model. Each of the entities introduced was driven by requirements and scenarios elicited during analysis of the ASCI computational problem domain. Several of the computational scenarios described above allude to this. To make some relationships more explicit consider the semantic map of Figure 5.

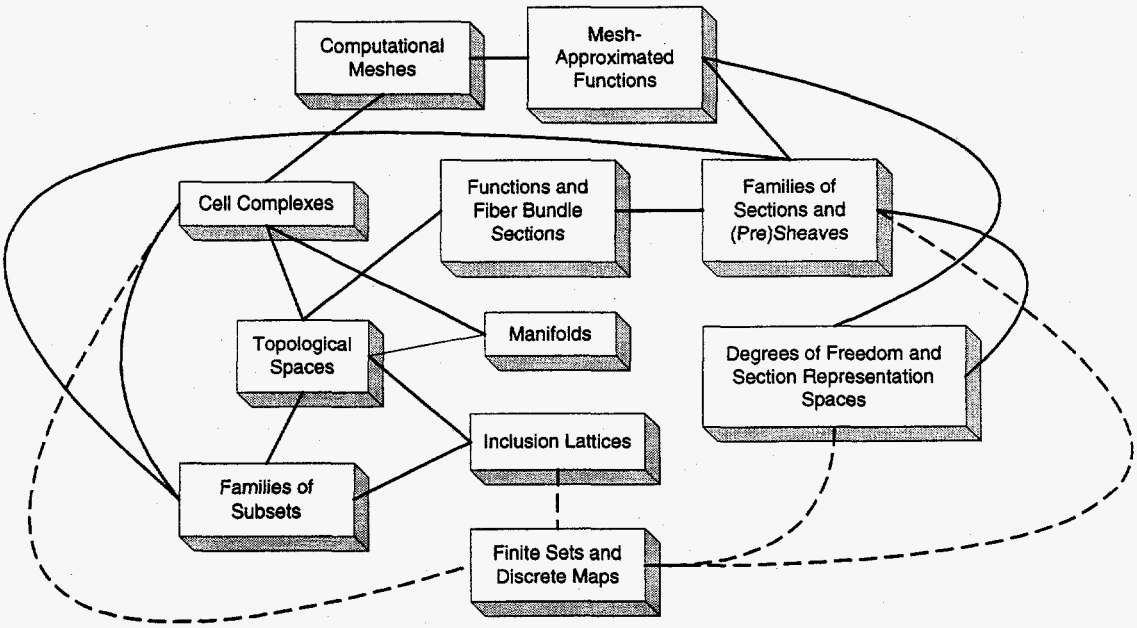

Figure 5. A semantic map of the data model.

Roughly speaking, this map shows semantic relationships between the various mathematical elements of the model. For example, the computational meshes and mesh-approximated functions are related to more fundamental entities such as cell complexes, manifolds, and fiber bundles. These are in turn related to topological spaces and their

representation as families of subsets ordered by inclusion. Discretization of the section spaces brings in sheaves and section representation spaces with their degrees of freedom. All discretized entities can be interpreted in terms of finite sets and discrete maps. Finite sets and discrete maps are also a good match to computational data structures such as arrays, lists, and so on. Thus the integrated model establishes clear semantic relationships between highly abstract entities like fiber bundles, cell complexes, and sheaves, and relates them to well-known discrete computational entities such as meshes and mesh-based variables. Discrete entities can then be related, via finite sets and discrete maps, to concrete data structures like arrays.

3. Data Model Prototypes

The DMF group has made considerable progress, both in its theoretical efforts and in developing practical and useful prototypes. Each prototype contributes to the goal of achieving a common data model and supporting software based on the conceptual architecture of Figure 1. Each also produces usable products in the near term.

3.1 CDMlib

CDMlib is a DMF prototype developed at Los Alamos National Laboratory. The main goals of CDMlib are to develop specific mesh and mesh-based variable classes while exploiting underlying mathematical elements from the ASCI DMF model. CDMlib is a C++ class library that appears to users as an object-oriented run-time database with a set of pre-defined object classes. Users can program applications using the C++ classes directly, or by using the procedural APIs provided for C and Fortran. DMF model entities are incorporated directly into CDMlib classes without a specific implementation of the Fiber Bundle Kernel layer. In this way CDMlib is able to explore a wide range of

possible representations and methods for FBK entities within the same implementation. CDMlib is currently a serial library undergoing design modifications in order to add parallel features.

CDMlib provides two basic categories of meshes: product meshes and unstructured meshes, as shown in Figure 6. Simulation fields are defined in terms of these. The meshes conform to the cell complex model. Thus, the data are mapped as discrete values onto some subset of the mesh cells, e.g., zones, faces, vertices, etc. In this way CDMlib supports a wide range of common mesh representations and data centering schemes. The library also includes an array object for general-purpose data storage, and a hierarchical arrangement of objects within the file based on an object set construction.

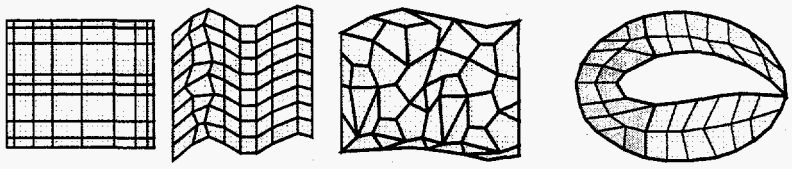

Figure 6. Several mesh object types supported in CDMlib. The first two meshes are variations on the theme of a product mesh or regular mesh. The second two are variants of the unstructured mesh type.

3.2 VB API and DMF API

The VB API and DMF API prototypes, developed at Lawrence Livermore National Laboratory and Sandia National Laboratory respectively, represent an attempt to implement the complete architecture of Figure 1.

The Vector Bundle API or VB API is based on early specifications of the Fiber Bundle Kernel that incorporate some, but not all of the elements described above. The term vector bundle comes from a specialization of fiber bundles to fibers that are vector spaces. The underlying data structures in VB API are relations or tables, and the

programmer uses the API to instantiate records in one or more of these tables. VB API entities currently include CELL, INCLUSION_MAP, FIBER, BUNDLE, and FIELD, as well as one or two others related directly to storage. Here CELL refers to degrees of freedom associated with a subset, and FIELD, is the term used for a fiber bundle section. The implementation is fully parallel, exploiting the collective parallel communication features of parallel HDF5. It is built on a supporting layer that manages table and data type descriptions and mediates between the upper layer and HDF.

The DMF API is a parallel prototype of the application layer of Figure 1. It is based on a procedurally-implemented object-oriented design. The objects currently included are SET, FIELD_TEMPLATE, STATE_TEMPLATE, STATE, and SEQUENCE. In this case SET takes on the same role as CELL in the VB API, and FIELD_TEMPLATE corresponds to BUNDLE. STATE and STATE_TEMPLATE, are entities introduced to contain whole collections of variables corresponding to a given base set. SEQUENCE is introduced in order to support the use of time or some other simulation parameter to denote a series of simulation states.

VB API and DMF API are designed to be integrated into a complete top-to-bottom parallel prototype of the proposed DMF architecture.

Acknowledgement

The authors wish to thank our many colleagues and ASCI application team members for their suggestions and advice in developing the data model. We are especially grateful to the members of the software development teams including Jim Reus, Tom Robey, Jim Holten, Jonathan Parker, Ted Reed, and Pat Medvick. This work was supported by xxxx.

References

Ambrosiano, John, Carlie Coats, Mark Reed, Atanas Trayanov, Richard Loft, Celeste Matarazzo, Tim Turner, Mladen Vouk, "Data Archetypes for the Fusion of Parallel Simulation Codes in Climate Modeling and Other Applications," *Proc. of the 2nd*

Workshop on Parallel Object-Oriented Methods and Applications (POOMA), Santa Fe, NM (1996).

Birkhoff, Garrett, *Lattice Theory*, Amer. Math. Soc., 1967.

Butler, D.M. and M.H. Pendley, "A Visualization Model Based on the Mathematics of Fiber Bundles," *Computers in Physics*, September/October (1989).

Butler, David M., and Bryson, Steve, "Vector-Bundle Classes Form Powerful Tool for Scientific Visualization," *Computers in Physics*, **6** (6), 576-584 (1992).

Butler, D.M., paper in preparation (1999).

HDF5, *Introduction to HDF5 Release 1.0*, http://hdf.ncsa.uiuc.edu/HDF5/doc/H5.intro.html, 1998.

HDF, *HDF4.1r2 Users Guide*, http://hdf.ncsa.uiuc.edu/training/HDFtraining/UsersGuide, 1998.

Larzelere, A.R., II, "Creating simulation capabilities," IEEE Computational Science and Engineering, Jan.-March, **5**, 1 (1998).

MPI2, *MPI-2: Extensions to the Message-Passing Interface*, http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html, 1997.

Novak, J.H., J.O. Young, D.W..Byun, C.J. Coats, G.L. Walter, W.G. Benjey, G.L. Gipson, S. K. LeDuc. Models-3: A Unifying Framework for Environmental Modeling and Assessment, Proceedings of the American Meterological Society 78th Annual Meeting, January 11-16, 1998, Phoenix, AZ.

Oliviera, Joseph S., private communication and unpublished notes (1999).

Szasz, Gabor, *Introduction to Lattice Theory*, Acad. Press, 1963.

Tennison, B.R., *Sheaf Theory*, Cambridge, 1975.

Thurston, William P., *Three-Dimensional Geometry and Topology, Volume 1*, Princeton, 1997.