THE DYNAMIC INFORMATION ARCHITECTURE SYSTEM: A HIGH LEVEL ARCHITECTURE FOR MODELING AND SIMULATION

14 November 1995

Decision and Information Sciences Division
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439-4832

Operated by The University of Chicago For the United States Department of Energy

Table of Contents

List of Figures

THE DYNAMIC INFORMATION ARCHITECTURE SYSTEM: A HIGH LEVEL ARCHITECTURE FOR MODELING AND SIMULATION

A White Paper Prepared By:
Advanced Computer Applications Center
Argonne National Laboratory
Argonne, IL 60439-4832

1. INTRODUCTION

The Dynamic Information Architecture System (DIAS) is a software framework intended to facilitate holistic management of information processes, including the construction of federation simulations from component models according to a user supplied context. These processes are modeled in DIAS as interrelated actions caused by and affecting the collection of diverse objects - which may range from abstract concepts represented by a simulation, through data sets to real world objects, and input from simulators. The domain of DIAS is flexible, determined by the objects available within DIAS and by the collection of models and other data processing applications which have been gathered by users to address specific information processing concerns.

The DIAS concept architecture was adopted in mid-1993 by the Joint Chiefs of Staff/J-8 for use in developing a prototype terrain reasoning and synthetic terrain generation system in the form of DEEM, the Dynamic Environmental Effects Model. DEEM is the first well developed application to be assembled using the DIAS. DEEM has been adapted to provide optimal interoperability among environmental effects models. Within other contexts, the architecture would be adapted in a different way. In addition, DEEM has been selected by the USAF Air Weather Service as the software framework for a multidisciplinary environmental modeling effort in support of theater-level mesoscale weather forecasting. The DEEM environmental taxonomy is also being used as the environmental component in the ARPA-funded Joint Task Force Advanced Technology Development Object Modeling Working Group and the Joint Simulation System (JSIMS). DEEM is being proposed to be used with FORSCOM's Mobilization Station Assessment Model for used in studying the impact of the environment on mobilization stations, developed for FORSCOM. Development efforts for the DEEM application based on the DIAS architecture have been sponsored by a number of agencies including the Joint Chiefs of Staff/J-8, US Marine Corps, Defense Modeling and Simulation Organization, USAF Air Weather Service, US Transportation Command, FORSCOM, and DOE.

The DIAS development effort is also a partnership involving other agencies and organizations. The DIAS program has benefited from cooperation and modeling contributions from various DOD partners including the USA Topographic Engineering Center, USA Waterways Experiment Station, and the USAF Phillips Laboratory Geophysics Directorate. Technology and applications produced from the DIAS program are freely made available to other government agencies. A DIAS Users Group meets on a quarterly basis to discuss developments, goals, and future efforts.

DIAS is a software framework into which models and information processing applications and databases can be placed. This object oriented system can be thought of as containing a library of reusable objects which have been designed for use in a wide variety of applications. DIAS also employs a sophisticated Context Manager that employs an expert system to assist the user in setting up a new "component" application to match the needs of the information management task under consideration. These determinations are made based on an analysis of the "context" of the scenario being considered. During the development of a new task, DIAS can dynamically change the resolutions being employed in order to efficiently provide the correct level of detail required for the application.

DIAS is also designed to be able to easily interface with existing models and information processing applications. In this way, it can import the various physics and process models that are required to provide the functionality for a given simulation. The majority of these models already exist within the DOD R&D community.

2. KEY FEATURES OF THE DIAS ARCHITECTURE

The DIAS architecture has been designed to provide interoperability among a wide range of models and information processing applications. Examples of the architectural features that support this are:

- DIAS is a fully object-based modeling system which supports distributed, dynamic representation of interlinked processes. The DIAS object library contains a variety of objects to represent a wide variety of processes.
- DIAS is designed to accommodate a wide variety of data types and formats. For example, it can accept terrain elevation and surface features in DMA, USGS, and TIGER formats; interpreted remote sensing imagery; CAD-type data of 3-D objects, and data from most Commercial-Off-the-Shelf (COTS) Database Management Systems.
- DIAS utilizes an expert system-based Context Manager to determine the requirements for a given application. The Context Manager supervises the creation of transient or persistent "Projection" Objects that are used for interprocess data transfer. The Context Manager also supervises the application process via a global Event Manager which provides the mechanism by which events are posted, activated, received, and activated upon by DIAS objects.
- DIAS is designed to operate under the UNIX operating system. DIAS operates in a mixed language environment. The "core" DIAS software utilizes C++ and SmallTalk. DIAS models are run in whatever language they were developed in; *e.g.*, FORTRAN or C.
- DIAS has been developed to provide for easy integration of legacy-type models and application tools, such as the wealth of DOD funded physics models. Once a model has been "registered", it can interact, as appropriate, with any DIAS simulation.
- Models or applications used within DIAS interact indirectly with one another via objects in DIAS's object library. This means that there is no disruption to the federation of actors when a model is added or removed.
- DIAS's Graphical User Interface system employs a GeoViewer module that is designed to provide the ability to display results in a spatially oriented manner.

3. OVERALL STRUCTURE OF THE DIAS ARCHITECTURE

Figure 1 shows the top-level architecture of DIAS. An external Graphical User Interface (GUI) is used to input commands as well as display results from a simulation. (It is noted that the GUI need not be used in a DIAS application. If DIAS is used as a callable object from another application, the commands to DIAS would be passed by the calling system and the GUI would not be required.)

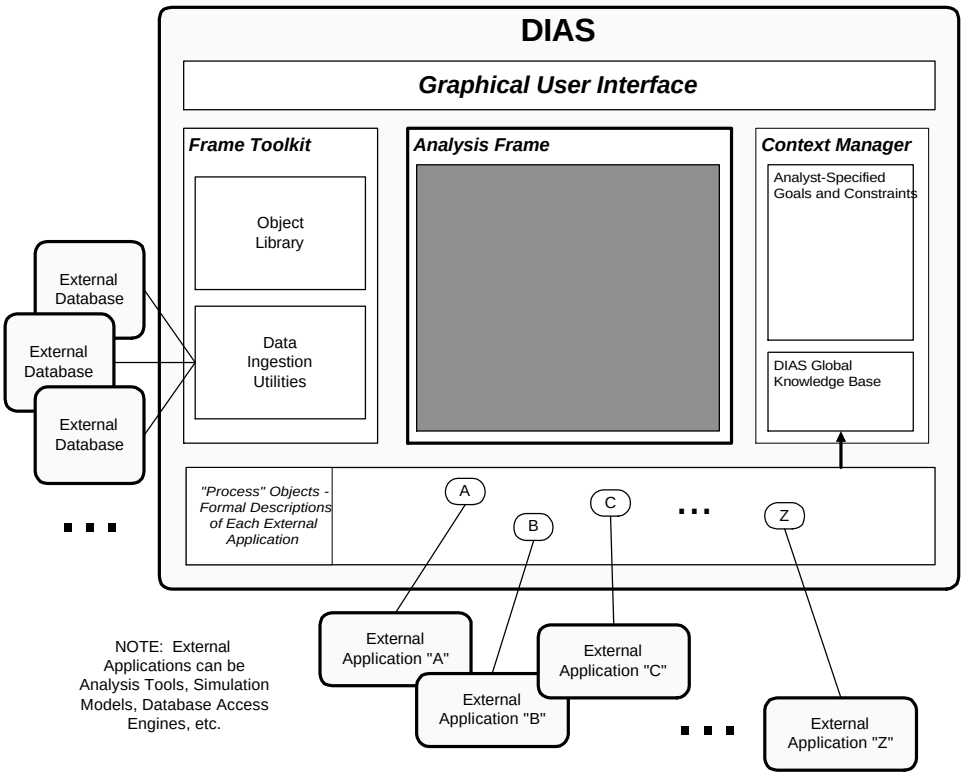


Figure 1. Top Level Architecture of Argonne's Dynamic Information Architecture System.

The DIAS Context Manager is used to interpret the requests and instructions supplied by the user and identify the DIAS resources to be utilized during the course of an application development, such as objects from the DIAS Object Library, databases, and models. During the execution of a simulation or analysis, the Context Manager controls the interactions between objects and models using a rules-based expert system. In the following subsections**,** major components of DIAS and their interrelationships are discussed.

3.1 Major Components and Their Interrelationships

3.1.1 DIAS Architecture Overview

DIAS is a fully object-based software system which supports distributed, dynamic representation of interlinked processes at variable scales of resolution and aggregation. A DIAS application is built from a collection of diverse "Entity" objects which may have real-world counterparts (Atmosphere, Forest, Stream, Railyard, etc.) or which may be other modules such as information processing applications. The state of an entity at any instant is defined by the values of state variables held as private data by the DIAS Entity objects. Various aspects of the behavior of the entity are addressed by models or other modules, either external or internal, which are accessed by methods (functions, subroutines) of these objects. These methods are invoked in response to Events which are selectively broadcast by the Event Manager. There is currently a DIAS focus on object classes related to environmental representation because of the use of DIAS to develop DEEM for the JCS/J-8 and other customers.

A DIAS "Frame[*] " object is created for each area, or subject, of interest (AOI). Each Frame object contains hierarchies of "Entity" objects representing landforms, soils, vegetation, infrastructure elements, etc.[†] Figure 2 gives a schematic representation of how a DIAS Frame is populated by the DIAS object library and how the object attributes are supplied via the DIAS data ingestion utilities. These data ingestion utilities have been developed to be able to supply the object attributes from a variety of external data sources. (Section B.2.2 in Appendix B describes the data ingestion pathways employed in DEEM, the DIAS instantiation used for environmental studies.) Each Frame also includes the means to access any objects within its geographic bounds. In environmental applications, the Frame can access a global Ephemeris object for solar system geometry and the location of deep-sky objects as a function of time for illumination, tidal effects, etc.

DIAS Entity objects contain (by reference) a list of Aspect objects, each of which encapsulates a single aspect of the Entity's behavior. For example, a DEEM "Stream" Entity representing a section of a river can have many possible aspects, corresponding to its behavior in terms of hydrology (as a component of a drainage system), navigation (as a link in a waterborne transportation system), meteorology (in terms of moisture, heat, and momentum exchange with the atmospheric boundary layer) and visualization (for photorealistic, schematic or GIS-type displays), among others.

When a DIAS Entity receives an Event which cues it to perform some aspect of its behavior (*e.g.*, a request to the Atmosphere to account for surface moisture and heat exchange for a specified timestep), the Entity routes the Event to its appropriate Aspect object(s). Each affected Aspect responds by calling for the execution of a specific process encapsulated in a DIAS "Process" object. **A key feature of the DIAS architecture is that the Process chosen to implement each Aspect in a simulation is selected *at run***

[*] Appendix A contains a glossary of DIAS terminology.

[†] See Appendix B for a discussion of the entity objects used in DEEM, the environmental instantiation of DIAS.

***time*, based on the *context* of the application.** This feature brings important benefits in terms of flexibility and interoperability.

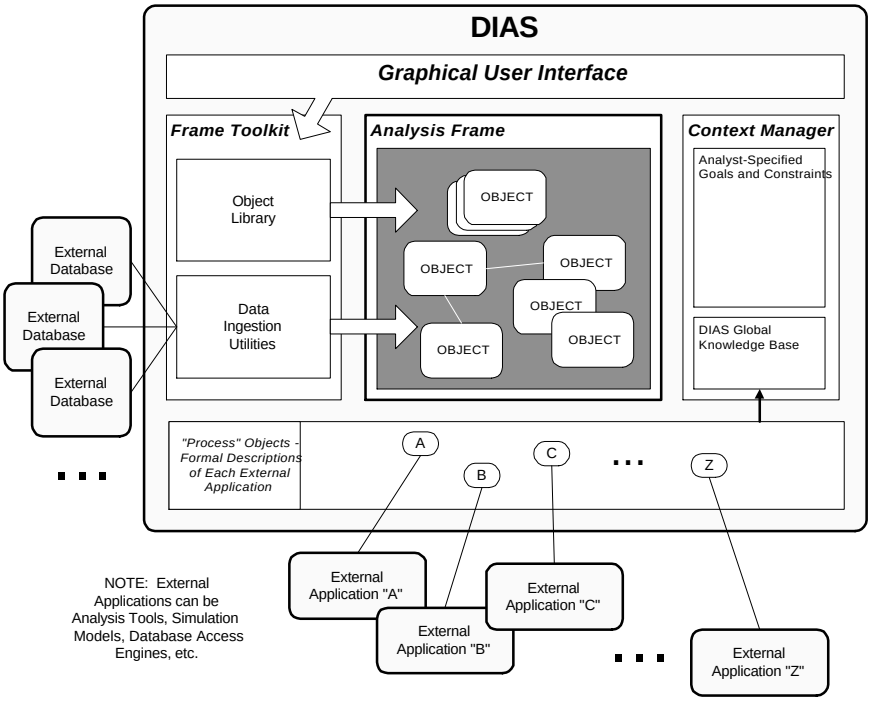


Figure 2. Representation of How a DIAS Frame is Populated by Objects and the Attributes Supplied by External Databases.

Dynamic processing of the contents of a DIAS Frame is accomplished within a modeling context specified by the user via the DIAS Context Manager object. The core of the Context Manager is a rules-based expert system. The Context Manager interactively accepts user requirements for classes of behaviors which need to be represented, such as output requirements, degree of fidelity, spatial and temporal resolution, etc. It then uses its knowledge base to facilitate the accomplishment of the user's goals within the context of (1) the user's request, and (2) the modeling capabilities presently "registered" with DIAS via distributed "Model" and "Process" objects. Figure 3 gives a schematic representation of how DIAS matches the user-supplied goals and constraints with the resources available (objects, applications, knowledge bases, etc.)

The DIAS Model objects act as proxies for actual model code or information processing applications. These objects, which contain one or more Process objects, present formal descriptions of their respective model codes, in terms which allow the Context Manager to reason about them in constructing the dynamic linkages for a specific simulation.

Each Process object addresses a single specific aspect of behavior. The Context Manager's expert system attempts to optimally associate and link available Processes with corresponding Entity Aspects to meet the constraints supplied by the user. The user may specify a context in any level of detail, including stipulation of specific Processes to be used to address specific Aspects, if desired. The Context Manager's expert system will

assemble a set of Aspect-Process dynamic linkages which will meet the goal if possible, and will inform the user if the goals are presently unattainable as stated.

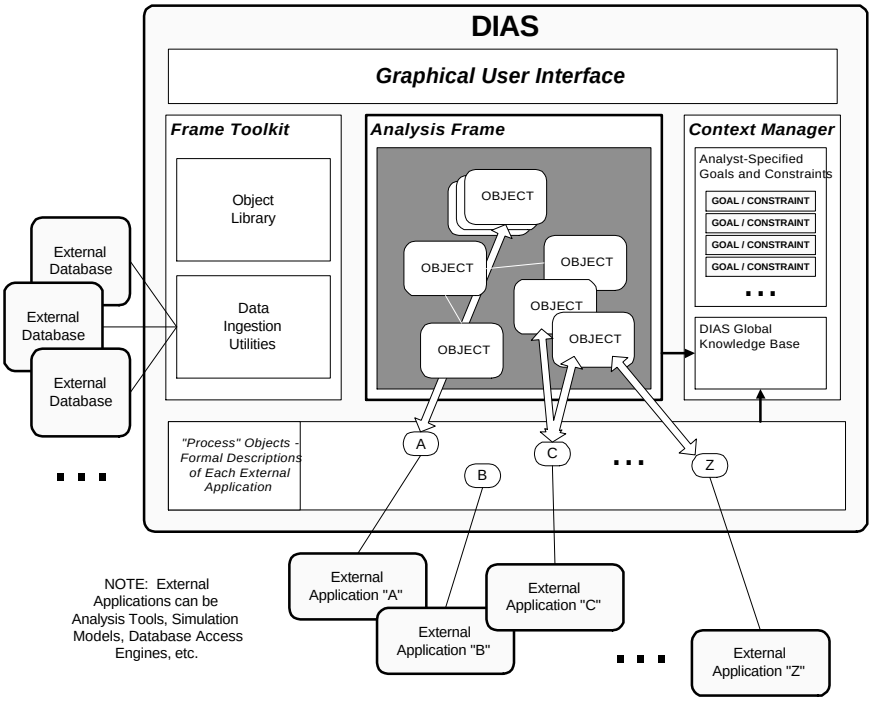


Figure 3. Representation of How DIAS Links User Goals and Constraints to DIAS Resources, Such as Objects, Applications, and Knowledge Bases

The Context Manager examines the data input and output needs of the specific Processes chosen for the application under development and builds "Projection" objects, which serve as flexible, *ad hoc* containers for any necessary transfers of data among DIAS Entity objects and between these objects and external models or modules (via Process objects). The Context Manager also selectively enables an appropriate, context-specific subset of the triggering mechanisms by which objects create DIAS Events and log them with the Event Manager. These Events, which may have specific Projection objects associated with them, are used to expedite and synchronize data transfer.

A schematic diagram of the generic linkage between Entity Aspects and Processes is shown in Figure 4. An example of such a linkage, which has been implemented for Aspects of the Atmosphere object in DEEM, is provided in Figure 5. In Figure 5, the Atmosphere is being asked to express two of its aspects:

1. Evolve atmospheric state (3D winds, temperature and moisture profiles, clouds, precipitation, etc.) forward to a succeeding time step.
2. Update atmospheric surface exchange (moisture and sensible and latent heat fluxes throughout the atmospheric surface layer) for a succeeding time step.

Note that the time steps for these two activities are, in general, not the same.

For the modeling context shown in Figure 5, both of the active Aspects of the Atmosphere have been associated by the Context Manager with Processes carried in external models. The "Evolve Atmosphere" Aspect has been associated with the "Evolve Atmospheric State" Process of the NCAR/Penn State Mesoscale Model Version (MM5). The

"Surface Exchange" Aspect has been associated with the NCAR Biosphere/Atmosphere Transfer Scheme (BATS) model. The DIAS dynamic linkages are fully distributed -- the actual MM5 and BATS codes can exist at any accessible node on a network. The Projection objects shown in Figure 5 carry the requisite Atmosphere parameters between the Atmosphere object and the Processes which modify the state of the atmosphere. (Note that, although the MM5 model is itself capable of modeling surface exchange at one level of detail, the current modeling context in the example calls for use of the more detailed BATS surface exchange process instead.)

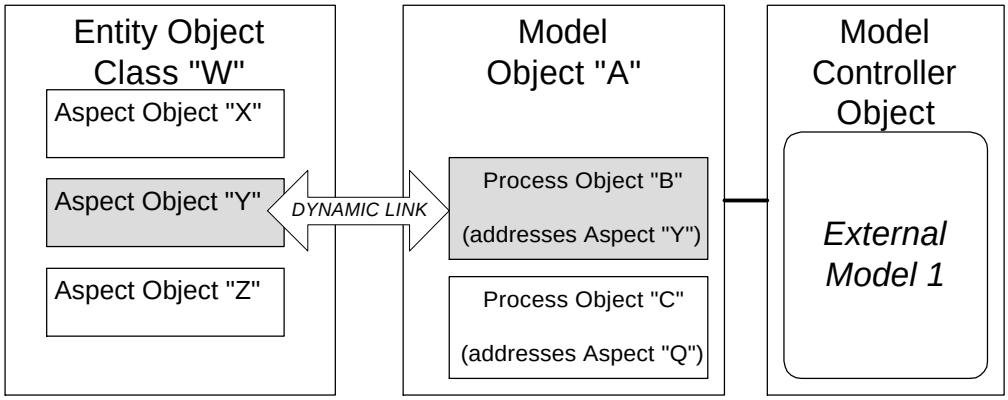


Figure 4. A Schematic Diagram of the Generic Linkage Between Entity Aspects and Process.

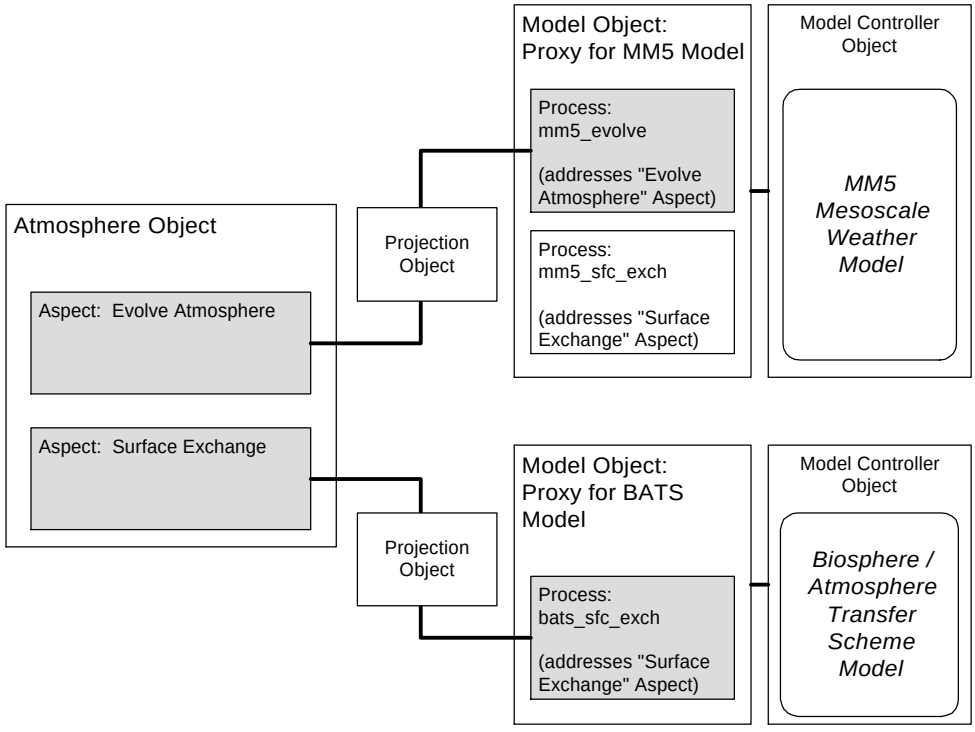


Figure 5. An Example of the Linkage Between Entity Aspects of the DIAS Atmosphere Object.

For the modeling context shown in Figure 5, both of the active Aspects of the Atmosphere have been associated by the Context Manager with Processes carried in external models. The “Evolve” Aspect of the Atmosphere has been associated with the “Evolve Atmospheric State” Process of the NCAR/Penn State MM5 mesoscale meteorological forecast model. The “Surface Exchange” Aspect of the Atmosphere has been associated with the NCAR Biosphere/Atmosphere Transfer Scheme (BATS) model. The DIAS dynamic linkages are fully distributed -- the actual MM5 and BATS codes can exist at any accessible node on a network. The Projection objects shown in Figure 5 carry the requisite Atmosphere parameters between the Atmosphere object and the Processes which modify the state of the atmosphere. (Note that, although the MM5 model is itself capable of modeling surface exchange at one level of detail, the current modeling context in the example calls for use of the more detailed BATS surface exchange process instead.)

The modeling linkage described in this example, which has been implemented in DIAS for the USAF, is in fact a very important one, because it provides a mechanism for addressing feedback between atmospheric and surface processes -- rain which falls to earth will subsequently evaporate and change local moisture and temperature profiles, ultimately affecting winds and future precipitation. The models used in this particular example, MM5 and BATS, can provide relatively high resolution and fidelity, at substantial computational cost. Such a model linkage would be most suitable to high-resolution applications, such as weather forecasts for tactical air operations. If the user was more interested in agronomic applications, for example, the Context Manager would have chosen a lower-fidelity model for evolving the atmosphere, if one was available. In that case, a Process which simply accessed daily temperature and precipitation records from a surface observing station might be a better choice than using MM5 (*i.e.*, a data table could be used, rather than a model.)

Under DIAS's software architecture, the user is given a rich set of real-world objects that can be imbued with diverse behaviors by being dynamically associated with modeling capabilities in various disciplines and at various levels of fidelity. In practice, the scope and power of DIAS as an information management tool is a function of the ensemble of internal and external models that are available to the Context Manager. For maximum flexibility and interoperability, we have chosen to limit the internal models carried within the DIAS application to a few relatively uncontroversial functions, mostly connected with object visualization. Thus, the external models must provide the bulk of the modeling functionality. It is expected that different users will have different suites of models; in general, the same set of Entity object classes will support all of these, although it will doubtless be necessary to expand the object schema from time to time by adding new Entity subclasses, new Aspects, and new private data elements.

External models are “registered” so that they can be used by DIAS by creating a Model object and subordinate Process objects for the external model code or information processing application. The Model object stands as a proxy for the external model. It includes a Model Controller object (described in Subsection 3.1.2.3) which provides DIAS's explicit linkage to external models' source code and data structures. The Model object carries the physical location of the model, and a formal description of the separable processes which the model contains, in the form of Process objects. (Models which are never

run piecemeal will have just one Process.) Each Process object carries (1) the identity of the Entity Aspect which it is capable of addressing, (2) a formal data dictionary of input and output parameters, and (3) expert system rules and facts which describe constraints to the Process' operation (*e.g.*, applicable ranges of spatial resolution, etc.). The DIAS Context Manager uses these formal definitions in reasoning about which models should be used to meet the user's stated goals, and how the data transfers (via Projection objects) should be structured and synchronized.

3.1.2 DIAS Infrastructure Object Class Characteristics

A number of DIAS object classes are used within the DIAS architecture. In this section we summarize the primary object classes and give examples of how they relate to a DIAS simulation. Table 1 lists the primary DIAS object classes and the following subsection contain descriptions of each object class.

Table 1. A Summary of the DIAS Object Classes.

Object Type	**Description**
Frame	The object class representing the area of interest in a DIAS simulation.
Entity	An environmental object (*e.g.*, such as a river, forest, road, atmospheric layer, etc.) with many possible aspects of dynamic behavior.
Aspect	An object class that is an expression of a single aspect of an Entity's behavior. An example of an aspect is the exchange of moisture and energy between the atmosphere and the surface.
Context	An object class that is a collection of metadata which governs the protocol by which environmental models operate on the DIAS state variables.
Event	An object which cues an Entity object to perform a stated aspect of its behavior.
Model	An object encapsulating the means of assessing a particular environmental model and a formal descritpion of its function via a network of Process objects.
Process	An object class that is a single portion of an environmental model, either external to on embedded within the DIAS framework, which addresses a particular Entity Aspect.
Projection	A DEEM object class that is a transformation of DIAS data for a specific use. A projection object can either be transient or persistent.
Model Controller	A component of Model object that constitutes a shell around an external model to allow it to converse with DIAS Entity objects.

3.1.2.1. Frame Object Class

A DIAS **Frame** object is created for each area of interest. Each Frame object contains hierarchies of "Entity" objects representing landforms, soils, vegetation, infrastructure elements, etc. Each Frame includes the means to access any objects within its geographic bounds, and accesses a global Ephemeris object for solar system geometry and location of deep-sky objects as a function of time, for illumination, tidal effects, etc.

3.1.2.2 Entity and Aspect Object Classes

The Entity class is the base DIAS class for all objects which are to be modeled. Each Entity may be composed of zero or more sub-Entities, which allows for the modeling or execution of objects at various levels of aggregation. A wide array of specialized environmental Entities have been derived from the parent class, and many more are on the way as the DEEM application is developed further.

Each Entity contains an embedded Locus object, which specifies the reference location of the Entity in two or three dimensions, as a point, polyline, polygon, or voxel volume. The Locus also specifies the datum for the Locus -- locations can be defined with respect to (1) the Locus of a parent Entity, (2) the Locus of the Frame to which the Entity belongs, or (3) directly georeferenced (*e.g.*, latitude and longitude or geocentric).

Each Entity also contains an embedded Aspects Vector object. The Aspects Vector is composed of an array of lists of Aspect objects. Each element in the array represents a category of object behavior. In the case of DEEM, this would include Meteorological, Hydrological, Visual, etc. The Aspect objects carried in each list correspond to particular aspects of the object's behavior in that category. For example, the Aspects of a Stream Entity carried on the "Hydrological" list might include such issues as rate of infiltration/exfiltration of groundwater, capacity for silt suspension and transport, and streambed scouring dynamics, among others.

Each Aspect object carries a list of all of the types of DIAS Events to which it can respond. The Aspect will have an event handler method for each such Event. Each Aspect object also carries a pointer to the Process object which the Context Manager selects to implement the Aspect's behavior. Aspects are capable of creating new Events and filing them with the Event Manager, in addition to receiving and consuming Events.

3.1.2.3 Context Manager and Context Object Classes

The DIAS Context Manager object class incorporates an expert system shell. The present prototype utilizes NASA's CLIPS shell, a forward-chaining, object-oriented, rule-based programming language (written in C), which is very flexible and fast.

A DIAS Context Manager object is associated with each Frame object. The Context Manager operates on a Context object which is built by a user for a specific analysis/display context. Context objects contain the metadata which indicate which processing options and pathways have been selected to meet a specific user-defined set of goals. Specific Contexts built to serve specific purposes can be built, edited, copied, archived, and retrieved as needed.

In setting up an application or simulation, the Context Manager:

- Supervises a user's interactive specification of the analysis context (desired form of results, processes and data sets to be employed, spatial and temporal resolution, etc.). This specification process results in an *ad hoc* rules base of analysis context information which is then carried in a Context object.
- Based on the user's specifications rules base, and the separate rules bases of eligible Processes, the Context Manager nominates Processes (from external or internal modeling modules) to address the requisite behaviors.
- Assembles a combined simulation context rules base in the Frame's Context object, consisting of user's specifications and the various Process rules, to govern execution of a simulation.

Before a context is established, there exists a full set of available Entity subclasses and their associated Aspects, as well as a full set of available Event types, each with a list of Aspects which have event handlers for responding to them. One result of the constraints applied in the context specification process is that only certain of the Event types will be active and only certain of the allowable Aspects will have their event handlers for these events enabled. For example in DEEM, if the context of a simulation called simply for visualization of a scene, the Visual Aspects of a Stream object in the Frame would have their event handlers enabled, but Hydrological Aspects such as seepage, siltation, etc. and the associated Hydrological Events would not be needed. The Context Manager is responsible for setting the activation status of each Event type and Aspect. In executing an application or simulation, the Context Manager:

- Supervises creation of transient or persistent Projection objects for inter-Process data transfer, based on the data dictionary requirements and other constraints in the combined context rules base.
- Via DIAS's global Event Manager, the Context Manager supervises the actual process by which a simulation is viewed or evolved in time. Specifically, the Context Manager:
  - ⇒ Sets up the mechanism by which Events are posted, activated, and received and acted on by DIAS objects,
  - ⇒ Triggers the initiating Event, and
  - ⇒ Manages the simulation event queue, including the handling of external and user events.

Execution of the analysis or simulation is handled through the posting and distribution of events. Objects interested in a particular event will notify the global Event Manager and provide a event handler to handle the event. The Context Manager maintains the DIAS Event Queue and raises events at their appropriate time. The Event Manager is responsible for distributing the event to all interested objects by invoking the object's event handler. New events are typically generating by objects when they are finished handling an event.

3.1.2.4 Model Controller Object Class

As noted earlier, each external model or application accessible by DIAS is represented by a Model object, which will include one or more Process objects. A Model Controller object is a component of a DIAS Model object. Each Model Controller object constitutes a shell around an external model or other information processing application which allows the model to participate in a DIAS simulation. The Model Controller includes a rules-based expert system component (presently in the CLIPS language) in which procedures called from rules can execute portions of the external model code. Other rules and procedures of the Model Controller are responsible for translating between the model's internal data representations and the corresponding DIAS forms. Formal input and output data dictionaries for models, carried in the Process objects, are expressed in terms of DIAS "Parameter" objects, which have a common meaning throughout DIAS. The DIAS form of the input and output data streams is defined in terms of collections of Parameter objects carried in Projection objects.

To complete the linkage between the DIAS and an external model or application, it is necessary to insert monitor modules into their source or control sequence code. The monitors are needed to:

(1) Execute particular portions of the model code, corresponding to DIAS Processes. (Note that the DIAS grammar for representing Processes addressed by a model allows for the possibility of Process interdependence, including nested Processes.)

(2) Inject DIAS data as input (after being translated by the Model Controller into the model's native form).

(3) Accept model data as output. The Model Controller is responsible for subsequently translating the model information into DIAS form.

It should be stressed that the translation between a model's internal data representation and the DIAS form takes place in the Model Controller rules and processes, not in the model code itself. This approach is intended to keep the code changes required in the external models or applications to a minimum, to facilitate configuration control.

The bulk of the effort needed to accommodate a new model within DIAS is related to the creation of the expert system rules and procedures needed to translate between data representations. This effort is unavoidable, but need only be expended *once per model*. With the aid of the Model Controllers, external models communicate with DIAS in DIAS's common data language, rather than being allowed to communicate directly with each other. As a result, the addition of a new model into the DIAS framework, or modification of a model already "registered" with DIAS, requires no modifications to any *other* software components. Thus, a DIAS installation with N models registered will have exactly N unique model interfaces; if models could communicate directly with each other, up to N^2 such interfaces would be needed.

3.1.2.5 Projection Object Class

The DIAS Projection object class represents an intelligent, flexible container object used to pass data among distributed DIAS objects, including real-world Entity objects and the Process objects which act as the interfaces to various external models or information

processing applications. The purpose of a Projection is to recast DIAS data into a form directly usable by a particular DIAS Process. Projections may be transient or persistent. For example, while small Projection objects used to convey effects of a munition on terrain may be created and deleted frequently in the course of a simulation, a Projection which holds large arrays of surface characteristics (*e.g.*, elevation and surface roughness parameter) at the resolution and grid specifications preferred by a weather modeling Process may persist for the duration of a simulation, and may in fact be retained in the object database long after a simulation is over.

A Projection can contain a variety of named data objects ranging from very simple scalar data elements to very complex objects. Projections can also hold hierarchies of subordinate Projections.

Projection objects contain the necessary mechanisms to reformat their data contents to generate new Projections needed for other Process objects. Projections have the following capabilities:

- Insert and retrieve named parameters of various types.
- Encode/decode themselves for storage or transmission to a remote Process utilizing a variety of communication mechanisms.
- Transform parameters via a collection of standard mapping functions. Such functions implemented or planned include:
  - ⇒ Changing spatial resolution in gridded data.
  - ⇒ "Rasterization" of vector GIS-type data to obtain area-weighted parameter values (*e.g.*, surface roughness, leaf area index, etc.).
  - ⇒ Transformations involving more complex processing, such as deriving gradients from gridded fields.

3.1.3 DIAS Object Database

Since DIAS is an object-based system, it is perfectly suited for use with an Object Database Management System (ODBMS). While most currently available ODBMSs are proprietary, the gains received from not having to write extra reformatting, memory management, and synchronization code greatly outweigh the expense of the system. The Versant Object Database Management System is currently used in DIAS. This ODBMS has its own caching mechanism which is transparent to the programmer of the system as well as to the user. Objects (which contain all the data that the system needs) are referenced within the code as if they were in memory at all times. These same objects are saved as persistent objects within the ODBMS, thus encapsulating the data with the object. The following benefits are derived from using an Object Database Management System in place of a traditional database system/file system:

- Schema Support
- The database schema "language" is the same as the programming language (*i.e.*, C++), and the schema is automatically captured from the application code; thus, eliminating the well-known "Impedance Mismatch" problem.

- Object structures do NOT have to be "flattened" into another representation, but can be stored in their native object format. This eliminates the cost of maintaining two representations of every object.
- No need for additional input/output or synchronization code to remap objects from the database into their native object format. Objects are referenced as if they are in memory at all times.
- Performance
- Elimination of the need to reconstitute complex, normalized objects through many, expensive join operations.
- Client-side object caching makes accessing the objects efficient (most recently referenced objects remain within the cache).
- Other similar projects that have used ODBMS have demonstrated improvements ranging from 10 to 100 times faster than the relational approach.
- Unique Features
- Support for design transactions allows transactions which can span multiple database sessions which can last several days with full commit or rollback capability.
- Support for versioning of objects.

3.2 Key Assumptions and "Golden Rules" to Guide Design

The following short list of guidelines characterize the concept and design of the DIAS software system:

- A DIAS analysis environment is built from objects (Entities) which have real-world counterparts or which supply data as from a database.
- The state of the environment is captured within state variables held as private data by the DIAS objects.
- The dynamic behavior of the environment is captured by models or information processing applications either external or internal to DIAS, which are not tightly coupled to the DIAS environment Entities -- the coupling is context-dependent and is determined at run time.
- Actual objects in the environment can respond to and affect their surroundings in many diverse ways. Thus, it is natural and appropriate for each real-world object to have the potential for several types and categories of interaction. This is realized in DIAS through the ability of Entities to carry multiple Aspects in various categories (*e.g.*, in DEEM: Meteorological, Navigation, Botanical, Visual, etc.), each implementing a particular environmental behavior through an associated Process object.
- The goals and constraints which pertain to an intended analysis or simulation can be formally expressed in such a way as to allow an expert system to deduce a viable solution. This is the premise upon which the operation of the Context Manager is based.
- Details of external models and Commercial Off-The-Shelf (COTS) software packages should be as transparent to DIAS as possible. Examples of this design tenet in action include:

⇒ DIAS treatment of external models. External applications may have as extensive a store of private data as they need to perform their functions. These data stores are of no concern to DIAS. The only data of interest to DIAS are those which are called out as DIAS Parameters in the formal definition of the model in the Model and Process objects.

⇒ DIAS treatment of COTS software packages for visualization. DIAS can presently display 3D perspective or map-style views of the objects in a Frame via any of several COTS visualization packages. When a DIAS Entity is asked to display itself, the Process called by one of its Visual Aspect objects writes graphic primitives (polygons, quadrilateral meshes, etc.) in a common format to the DIAS Picture object owned by the current Frame. The Picture object "knows" which graphics package is to be used in the present context and packages the graphics primitives appropriately. Thus the Entity and Aspect objects need not have any knowledge of which graphics engine is being used.

3.3 Technical Reference Model

The top-level design paradigm for DIAS is that DIAS is built from objects which have real-world counterparts. The state of the environment is captured within state variables held as private data, or attributes, by the DIAS objects. The methodology used to view DIAS is based on the Object Modeling Technique. (The reader is referred to Rumbaugh *et al.*[1] or Booch[2] for a discussion of the Object Modeling Technique.)

[1]Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991*) Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, pp 500.

[2]Booch, G., (1991) *Object-Oriented Design*, Benjamin/Cummings, Redwood City, CA.

4. ELABORATION OF KEY ARCHITECTURAL CONSTRAINTS

4.1 Independence From Data Formats

The formal, general way in which data representation has been approached in DIAS, in the Context reasoning process and in the definition of Projection objects to carry arbitrary loads of data, supports a potentially valuable capability to load data into applications built by DIAS via virtually any format without recoding any of the ingestion routines, as long as the layout can be represented in terms of DIAS Parameters and Projections. This has been tested and demonstrated in the DEEM application, as the following example shows.

4.2 Synthetic Terrain Data

Synthetic terrain data can be required in simulations for non-specific locations or when data for a given location are lacking. This has been tested and demonstrated in the DEEM application, as the following example shows.

For high-resolution modeling work, we will want to expand the DEEM functionality for synthesis of missing data and synthesis of data at a higher level of detail. For example, a polyline section of a railroad should be expandable (via AI and domain knowledge) into (1) right-of-way, with cuts and fills represented explicitly in high-resolution topographic patches (DEEM linear Bumps), appropriate changes in soil and subsoil and vegetation cover, and incidental structures and signage; (2) rails, ties, turnouts, crossovers, ballast, etc.; (3) bridge and tunnel structures where appropriate; and (4) rail link object(s) for a transportation net. This general line of reasoning can be followed for virtually any component of a landscape.

Most GIS sources are deficient in some way or another for use in modeling, creating the need to fill in data in various ways. A key example is topography as it relates to water bodies. In both DMA ITD and USGS DLG, streambed, lakebed, etc. elevations are generally not provided. If water level changes, DEEM must have an estimate of bed topography to be able to model water flow and open water coverage at mid to high resolution or to model vehicle fordings of streams. A solution which we will probably attempt is to provide “bed templates” with each section of shoreline, which can be used by a DEEM Bump object to subdivide the terrain surface and alter the elevations to scour out a suitable bed when soundings are unavailable.

4.3 External Models

DIAS's principal aim is to facilitate holistic modeling by allowing essentially any model or information processing application to work together harmoniously in the same frame of reference. The inventory of environmental models being interfaced via the initial DEEM application, to address specific sponsor concerns, includes models for mesoscale meteorology, surface moisture and energy exchange, distributed surface hydrology, plant growth and forest ecology, trafficability and weather effects on terrain, visibility and obscuration, and photorealistic visualization. Additional classes of behavior can be added with relative ease, as can additional models with overlapping functionality and scope of application.

4.4 User Interface

An assumption has been made that most DIAS applications will have a spatial context, or one which the user will wish to view in some kind of spatially oriented fashion. The DIAS user interface provides functionality in the following areas:

1. Frame Set-Up: interactive control over data ingestion and editing for population of DIAS Frames (areas of interest) for modeling information analysis.
2. Analysis Set-Up: interactive specification of analysis context, via the Context Manager.
3. Analysis Control: interactive starting, pausing, resuming, and ending of a simulation or information analysis task, with opportunity for intervention (edit), inspection and visualization during a run.
4. Inspection of Intermediate and Final Results.

The DIAS GeoViewer Module, which is described below and its ancillary Control and Visualization components, address the first, third, and fourth of these requirements. The remaining requirement is handled by the Context Manager interface.

4.4.1 DIAS GeoViewer

The DIAS GeoViewer module was designed to provide GIS functionality to navigate within a DIAS Frame and to create, query, view, and manipulate objects within that Frame. The design incorporates object oriented technologies to provide a flexible high performance data model tailored to spatially related DIAS objects. Tools are provided to perform tasks involving typical GIS functionality, data ingestion, linkage to external models and integration with other DIAS components. Figure 6 shows an example of the GeoViewer displaying spatial data for a region of Korea.

The GeoViewer provides two mechanisms for data ingestion. The first, data preview, is the ability to plug in modules that can read arbitrary native file formats. This mechanism builds GeoViewer objects that are representative of the spatial aspects of the native data. The plug-in modules can be derived from existing readers in languages supported by a particular hardware platform. Using the data ingestion tool, which is described below, the user could optionally choose to build DIAS objects via this mechanism that represent all facets of the native data. The data ingestion tool also allows the user to cull unwanted native data. The second mechanism translates the spatial aspects of DIAS objects into GeoViewer objects.

The DIAS GeoViewer incorporates an object oriented GIS engine. This engine was designed to provide fast access to tens of millions of objects while minimizing the demands on hardware resources. Key DIAS components comprising this technology include (1) a light weight “GeoObject” optimized for efficient spatial representation while providing complete access to the full DIAS object and (2) a quadtree-based spatial indexing system that has been extended to incorporate adaptive, data-driven pruning.

Underlying the GIS engine is an object store which has been extended to provide three levels of adaptive caching which allows commonly used objects to remain in memory and removes object clustering. This functionality can also be provided through third party

ODBMSs or other plug-in modules. Overall, the GIS engine achieves approximately O(log(n)) performance for spatial queries.

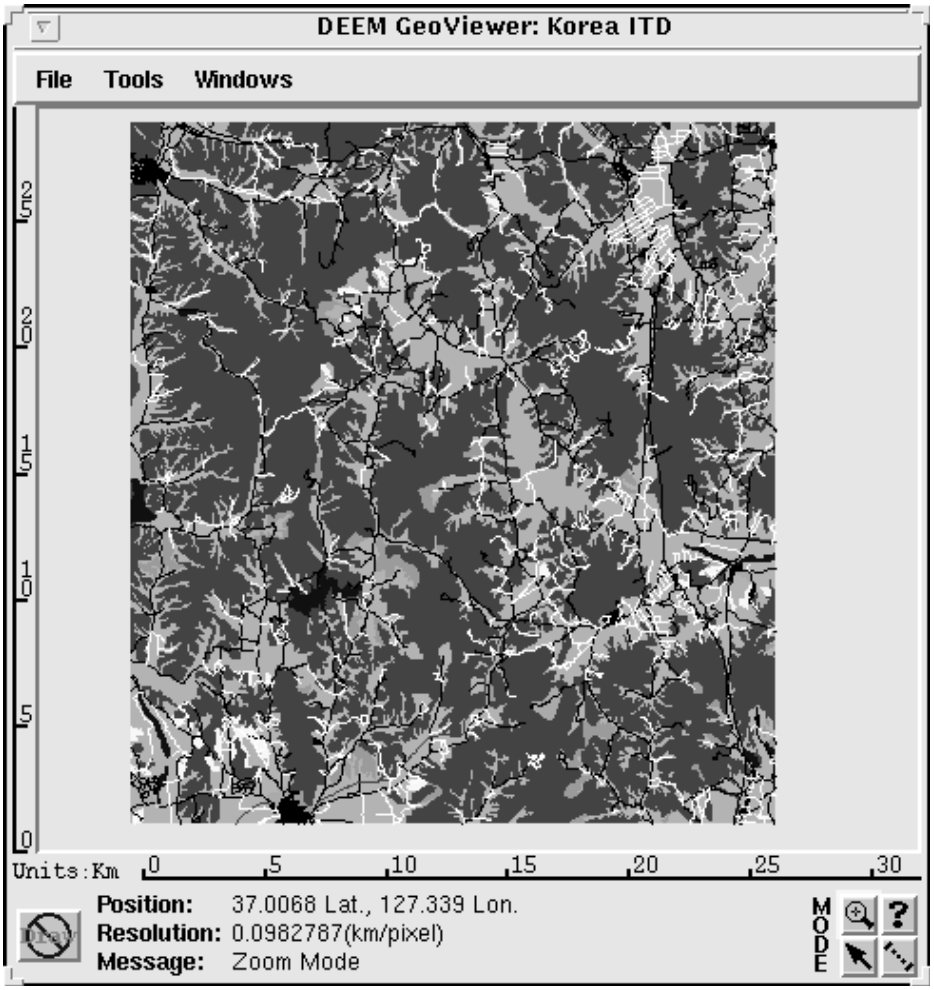


Figure 6. A Sample Display From DIAS GeoViewer. In the Example Shown, the Data are for a Region in Korea.

The GeoViewer's Layer Manager provides organization and visualization information to GeoViewer objects. All GeoViewer objects are assigned a layer which is contained within a hierarchy of other layers. The structure of this hierarchy is driven by the logical organization of the data of the DIAS Frame. Object visualization information describes how and when objects are to be visualized and is stored as a discrete sampling of a continuum of object representations. These data can come from three sources. An object can take its visualization information from the layer within which it currently resides. The point of the continuum is usually defined by the current view on the user screen, but can be explicitly set by the user. Visualization information for a layer can be inherited from parent layers via the layer hierarchy. If no visualization information is defined via this inheritance, then default information will be used. Optionally, an object can specify how to draw itself based on arbitrary object/user defined data.

The GeoViewer user interface provides a number of tools, including typical GIS tools such as zoom, pan, point query, relational queries, distance calculations and a host of others. Additionally, the DIAS GeoViewer provides an enhanced query facility that extends relational calculus to include object method calling and spatial relationships. This allows arbitrary processing to be performed as part of the query and does not limit queries to predicate based results.

Several tools are provided for layer management. These allow the user to explicitly view the layer hierarchy and modify the continuum of discrete visual representations. Other capabilities include browsing and editing of GeoViewer and DIAS objects and the relations between them. This is facilitated via the self-describing object meta-information and the dynamic editing object inspectors which are built at run time using that meta-information.

The GeoViewer is linked in with other user interface components such as the Analysis Control (providing controls for analysis start, pause, resume, end) and the photorealistic rendering module. The user can quickly set up a scene containing desired objects and their representations within the GeoViewer and have that scene passed to the photorealistic rendering component. (Note that the visual representation of objects in the GeoViewer need not be the same as that specified for the photorealistic renderer.) The GeoViewer data ingestion data preview mode in practice uses the DIAS data ingestion module, which is a separate module from the GeoViewer.

In the DEEM application, behavior of objects in the terrestrial environment includes their physical appearance. Each Entity carries with it in its associated Visual Aspect object(s) the metadata needed to determine how it should be represented graphically, in various modes and at various scales.

DIAS's visual behavior scheme explicitly includes both GIS-type and CAD-type features. It is also be able to generate CAD-type data representations from specifications and metadata carried in a simpler GIS-type data structure. For example, a “Forest” Entity with a polygonal geographical locus might automatically generate highly detailed, botanically correct trees and shrubs in an ecologically correct spacing and size and species mix, for purposes of rendering a close-up perspective scene. The same forest feature would generate a simple colored and textured polygon, if asked to portray itself as seen from high overhead.

DIAS supports scientific visualization and mapping graphics as well as photorealism. DIAS will generate imagery using any of a suite of advanced rendering software packages (presently including HOOPS™, and PHIGS FIGARO PRO™). These packages provide the advanced features such as raytracing and texture mapping needed to produce realistic visualization results. The details of the rendering process are encapsulated in the DIAS “Picture” object, so the DIAS Visual Aspect objects which actually generate the components of visual scenes are unaffected by the choice of renderer.

4.4.2 Context Manager Interface

In setting up an analysis, the user will be presented with the opportunity to create a new Context Object for the Frame, or select an existing Context from a Context Library,

for customization or use "as-is." The ability to build Contexts from scratch is a powerful one, but one which calls for a good deal of sophistication on the part of the user. We anticipate that, for some installations, it will be desirable to make only a limited menu of pre-built Context "templates" available to most analysts. The process of interactively defining a Context involves the following steps:

1. Selecting the goal. The goal will generally be expressible in terms of particular datasets which must be available at intervals during and/or at the end of the simulation, or in terms of particular analysis tools (with specific implied data requirements) which the user wishes to run. If the user wishes to allow the Context Manager's expert system to work out all the details based on goals alone, the user need only request that the Context Manager chain from the goal through the various modeling options and constraints to establish a viable Context. As with any other user interface to an expert system, it is very important that the DIAS context specification interface be able to clearly articulate the Context Manager's *reasons* for all of the decisions it makes regarding selection and linkage of modeling Processes, etc.
2. If the user wishes to take more control of the Context determination, he or she can provide additional constraints in terms of preferred Processes to be used to address particular Aspects, spatial and temporal resolution requirements, etc., at any points in the modeling linkage. Again, the interface must be able to articulate to the user when his or her choices are untenable, and when, due to dependencies inherent in the system, a selection forces other selections to be made as well.

I be expected that, unless the Context is specified by the user to such a rigid degree that the Context Manager has no choices left to make, there may be several Contexts capable of meeting the user's stated goals. To address this issue, it is our intention to switch to a fuzzy logic expert system approach (probably using the fuzzy logic extension of the CLIPS shell), to allow the Context Manager to evaluate the relative "goodness" of candidate Contexts according to various criteria (fidelity, CPU requirements, etc.), and then allow the user to choose among them.

4.5 Interfaces for Distributed Processing

Given the computational requirements of the type of model federation and other analysis applications DIAS is likely to be used for, including its core capabilities, visualization requirements, and external model processing, it was determined that distributing applications into many processes would be the only way to achieve our design goals. Traditional distributed computing environments typically involve a number of coordinated processes, each running on a separate platform, working together to accomplish tasks. Within an object-oriented framework, we would like to be able to distribute objects of various granularity across the network in as seamless a way as possible. This requires an infrastructure that allows objects to communicate without regard to specific platforms, implementation languages, and addressing mechanisms. There are several approaches which can be taken to achieve this goal. The following subsections describe some technologies that are available or planned to implement distributed computing environments for DIAS.

4.5.1 Parallel Virtual Machine (PVM) Library

Parallel Virtual Machine (PVM) is a software package that allows a heterogeneous network of serial and parallel computers to appear as a single large, parallel, distributed-memory computer. PVM provides heterogeneity at the application, machine, and network level, handling all the low level communication and architectural issues transparent to the applications programmer. PVM allows tasks to be started on the virtual machine, and provides a messaging mechanism for inter-task communication, allowing multiple tasks to solve some problems in parallel. PVM consists of a series of daemons which reside on each computer in the virtual machine, and a library containing functions for initializing tasks, communicating between tasks, and changing the configuration of the machines.

PVM is currently being used with the first DIAS application, DEEM, to provide Interprocess Communication. Since PVM frees the applications programmer from many of the low-level communication and architectural issues, it is a major improvement over other mechanisms such as sockets. However, since it is procedural in nature, rather than object-oriented, the programmer is still responsible for packaging and unpackaging of objects as they are transferred between tasks, as well as handling addressing issues. Another limitation of PVM is that it currently supports only a C and FORTRAN interface. Ideally, a distributed object environment should be able to work with a variety of languages.

4.5.2 Object Request Brokers

Object Request Brokers are the backbone of modern Distributed Object Environments (DOE), which simplify the task of building distributed applications by presenting the network as one large virtual machine in which remote objects appear to be local. The key advantage of Distributed Object Environments is that they free the developer from many of the low-level programming tasks necessary to achieve distributed objects, while at the same time providing many necessary services to applications which use them.

We are currently investigating the use of Common Object Request Broker Architecture (CORBA) compliant Object Request Brokers as a means of accomplishing a Distributed Object Environment. The CORBA standard is being developed by the Object Management Group (OMG), an international trade association representing over 250 leading information system and object technology corporations.

4.6 Applicable Standards

DIAS has been developed according to accepted programming and development procedures. Although DIAS has not had an application that has required it to be compliant with Distributed Interactive Simulation (DIS) application protocols, there are no fundamental reasons why DIAS cannot be made to be used with DIS applications.

5. ARCHITECTURAL MEANS FOR SUPPORTING CAPABILITIES

5.1 Interoperability

DIAS is intended to be a framework within which diverse models and information processing applications can simultaneously address various processes and phenomena. The DIAS architecture was therefore expressly designed to provide optimal interoperability among process models. Support for interoperability is derived from the following architectural features of DIAS, among others:

- Models interact with DIAS objects, not with each other. The interaction is formally defined in common terms (Projection objects and Parameter object data dictionaries). Since models do not interact directly with each other, there is no disruption to the suite of models when a model is added or removed.
- Models are not embedded in DIAS, but are, instead, associated with DIAS Aspect objects at run time. As a result, models can be freely swapped in and out of the DIAS framework.
- When visualization is required, DIAS objects present the information needed to visualize themselves to the DIAS Picture object, which hides the details of the graphics engine actually used in rendering. Thus, various graphics engines can be added or removed without affecting any DIAS objects other than the Picture object.

5.2 Object Reuse and Replacement

Since DIAS's architecture is based solidly on the object paradigm, taking full advantage of encapsulation, DIAS objects are completely “plug-compatibly” replaceable and reusable.

5.3 Scalability

As noted earlier in this document, external models registered for use with DIAS are required to express their I/O data needs in formal terms common to DIAS. No model “knows” about any other model, so there are no hidden inter-model dependencies. Thus, the number of model interfaces needed to support a suite of N models is N, rather than a value approaching N^2, as would be the case if models depended explicitly on each other. This architectural feature implies that DIAS will scale up gracefully as more and more models are added.

Furthermore, because the external models used to implement various environmental behaviors in DIAS are linked to DIAS objects by the Context Manager only at run time, and because DIAS explicitly supports distributed operation, the DIAS code will not become unmanageably large as more models are added. In fact, it will tend to grow very little as more models are added, since the only direct effect of adding another model is to add another Model object and one or more new Process objects to the rosters of available Models and Processes.

5.4 Time Management in a Federation of Models

With respect to time management, the DIAS architecture may be envisioned as a discrete event simulator, *i.e.*, events are executed with respect to their respective execution time irrespective of elapsed time. That is, time is maintained relative to the execution of events and their respective concept of time. Thus, execution is not dependent on a universal clock, nor is it dependent on a centralized timer. However, for simulations or exercises that require Universal Coordinated Time (UTC) or a similar capability, it would be implemented as a DIAS object providing a universal time to all models.

5.5 Configuration Management

Argonne utilizes standard configuration management procedures for the development of DIAS. Included in these procedures are strict version control procedures. At this time a formal configuration management protocol document has not been written but it is anticipated that one will be prepared in the future.

6. FUNCTIONALITY SUPPORTED BY ARCHITECTURE

6.1 During Simulation Set-Up

The functionality provided by the current DEEM prototype application of DIAS for simulation set-up activities for environmental modeling is presented here.

A broad range of data types are needed to fully characterize the state and behavior of a DIAS Frame, or area of interest. In DEEM, therefore, DIAS has been used to produce a design to accommodate a wide variety of input types and formats and presently accepts digital terrain elevation and surface features (in DMA, USGS, and TIGER formats at present), interpreted remote-sensing imagery, and CAD-format data on 3D objects.

The Context Manager expedites the user's interactive specification of the simulation "context" -- the goals, requirements, and constraints to be placed on a simulation.

The GeoViewer facilitates point-and-click querying and editing of objects in a map-based or photorealistic scene and provides the user with a spatially oriented view.

Other aspects of DIAS design directly support ease of data ingestion. For example, DIAS objects are "self-describing" -- edit screen layouts can be generated automatically from the object's internal structure, greatly decreasing the effort needed to build or modify the GUI. Also, the DIAS interface to external models, with Model Controller and Projection objects which translate between models' idiosyncratic data representations and the DIAS native data form, simplifies adding to or modifying the suite of models interfaced to DIAS.

6.2 During Execution

The user has control of the flow of an analysis via the Analysis Manager interface functionality. In the DEEM environmental simulation federation, it will be possible to pause at preset and/or arbitrary intervals to allow the user to "fly through" a simulation, inspecting (and, if desired, editing) the state of the simulated environment. We intend that the entire or partial state of the DIAS environment will be archivable at arbitrary checkpoint intervals for later analysis, if desired. All of the features of the DIAS GeoViewer for comprehensive inspection of the state of the simulation are available during execution as well as in the set-up phase.

6.3 During Post-Execution Analysis

As with the set-up and execution phases, all of the features of the DIAS GeoViewer for comprehensive inspection of the state of the analysis are available in post-execution analyses. The user will be able to browse through, visualize, and/or analyze the final state of the analysis environment, as well as any states checkpointed during execution.

6.4 Application Domains Supported

As noted earlier, DIAS's architecture allows essentially any models or information processing application to interoperate in the same frame of reference. DIAS is not limited by its architecture to a narrow range of resolution or fidelity. The collection of environ-

mental models being interfaced via the initial DIAS prototype represents only a small fraction of the potential scope of DIAS's application domain. The models already linked, or slated to be linked, with DIAS address mesoscale meteorology, surface moisture and energy exchange, distributed surface hydrology, plant growth and forest ecology, trafficability and weather effects on terrain, visibility and obscuration, and photorealistic visualization.

7. INTEGRATING MODELS INTO DIAS

In order for a model to be integrated into DIAS, the model must be "registered" with DIAS. Model registration consists of two steps: providing a formal description and augmenting the source code.

7.1 Providing a Formal Model Description

A formal description of a model is required in order to allow the DIAS Context Manager to reason about the model. The first step of the model description process is to obtain all of the metadata and information necessary to understand and characterize the model. This information is then used to create a new Model object. Then, the DIAS Entity Aspect(s) which the model will address must be identified.

Process Objects, one for each Aspect, must be created and added to the Model object. Each Process Object must specify which Aspect is being addressed. The formal input and output data requirements for each DIAS Process, in terms of DIAS Parameter objects, must be determined and added to the Process object.

Finally, any additional constraints on the model and its Processes must be determined and added to the rules base for the appropriate DIAS Process objects. For example, a model could be constrained in terms of spatial ranges or being valid for a specified time period, such as a given season.

7.2 Augmenting the Source Code

The model source code must be augmented in order to allow the model to participate in a DIAS simulation. It is stressed that models are integrated in the native language (*e.g.* FORTRAN, C,...) they were developed in. No attempt is made to rewrite the code in C++ or Smalltalk.

A Model Controller object must be created and associated with the DIAS Model object. The source code for the model must be then linked with the Model Controller. This is accomplished by converting the main program of the model to a subroutine or function that is called by the Model Controller; thereby making the Model Controller the main program.

Blocks of source code which correspond to the DIAS Processes included in the formal model description must be identified. Monitors are then inserted between these code blocks.

For the most part, the internal variables used within a model are of no concern to DIAS and are unaffected by the process of registering the model with DIAS. However, the variables which are directly involved in the interaction of the model with the DIAS environment, as formally defined in the Process input/output Parameter data dictionaries, must be identified. In some cases, the internal variable definitions will be consistent with a DIAS Parameter definition, so that data can be exchanged between the model and DIAS without translation. In other cases, it will be necessary to include a data translation filter in the DIAS Process Object to allow a model's internal variable to be expressed in DIAS terms.

7.3 A Model Integration Example

Figure 7 gives a schematic example of a model that has been integrated into DEEM. In the example shown, the integrated model is the Biosphere Atmosphere Transfer Scheme (BATS) that is used to perform surface exchange between the atmosphere and the surface.

On the left side of the Figure is the representation of the model object, the DEEM proxy for the BATS model. On the right side is the model controller which contains a software wrapper around the BATS model source code.

In the source code, two sections of code have been blocked out and tied to DEEM Processes. The first section of code involves model initialization and the second performs the actual model computations. The monitors in each block of code establish when and where the operations in the block are begun and completed. This information is provided to the corresponding Process object which contains event handlers that communicate with the overall federation simulation.

To date, a handful of models have been integrated with DIAS. Several models are scheduled for or are undergoing integration and others are being considered for future integration. Table 2 lists the models that are undergoing integration or are planned for integration during the next few months. The table also lists the developers of the models.

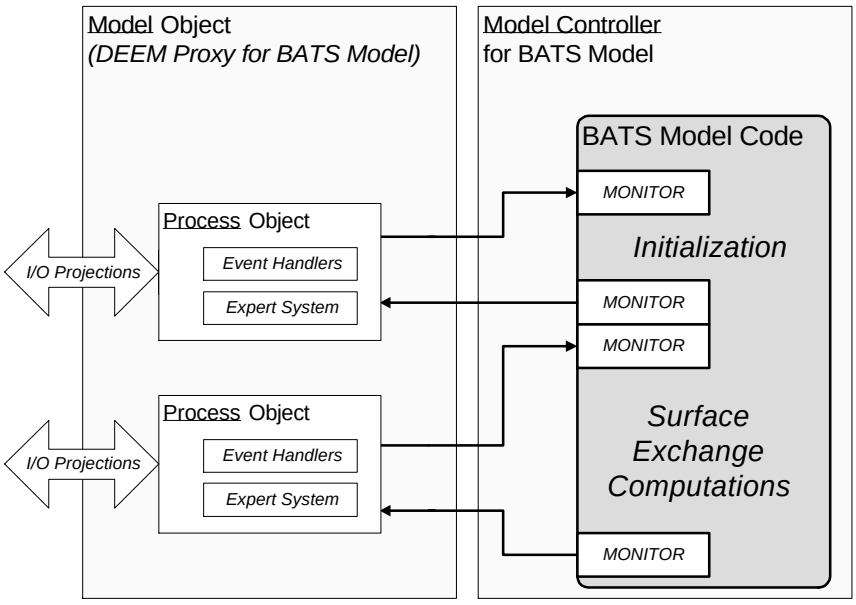


Figure 7. A Schematic Representation of How a Model is Integrated Into the DIAS Framework. In the Example Shown, the Model Integrated is the Biosphere-Atmosphere Transfer Scheme and is being Used in DEEM

Table 2. Models That Have Been or Are Planned, For Integration With DIAS. The Models Shown Will Be Used With DEEM

Model	**Integration Status**	**Developer**
Mesoscale Model Version 5 (MM5)	Completed In Prototype Form	Penn State /National Center for Atmospheric Research (NCAR)
Biosphere Atmosphere Transfer Scheme (BATS)	Completed In Prototype Form	NCAR
Soil Moisture Strength Prediction (SMSP)	Completed In Prototype Form	USA Waterways Experiment Station
Terrain Mobility Inference Algorithms	Completed In Prototype Form	USA Waterways Experiment Station
NATO Reference Mobility Model (NRMM)	Completed In Prototype Form	USA Waterways Experiment Station
Helicopter Landing Zones/Drop Zones Selection	Completed In Prototype Form	USA Waterways Experiment Station
Bivouac Sites/Assembly Sites Selection	Completed In Prototype Form	USA Waterways Experiment Station
Division Mobility Corridors	Completed In Prototype Form	USA Waterways Experiment Station
Mobility Corridors	Completed In Prototype Form	USA Waterways Experiment Station
Electro Optical Tactical Decision Aid (EOTDA)	Planned	USAF Geophysics Directorate
PLEXUS	Planned	USAF Geophysics Directorate

8. IMPLEMENTATION CONSIDERATIONS

8.1 Testing Compliance With Architecture

A basic tenet of the DIAS architecture is that a model will be ingested by configuring the knowledge-base. Implied in this configuration is the knowledge associated with ensuring the capability for data exchange with the model in a useful fashion; *i.e.* the semantics for the exchange will be built into the knowledge-base. Compliance with the architecture will be ascertained during the ingestion process. Once demonstrated, this compliance ensures that the model is interoperable with all other models in a given DIAS-based federation when a specific context requires the use of the ingested model's capability.

8.2 Technology and Risk Assessment

DEEM has been implemented based on DIAS to prove that a conceptual, architectural approach is valid. Thus, the current implementation may be envisioned as being prototypical of a generic architecture. From this perspective, the technological risks associated with adopting the DIAS architecture are similar to those associated with developing a production system from a prototypical system. The risks associated with such a development are well understood and, thus, the risk associated with adopting the DIAS architecture to a similar application is low.

The evaluation of any architecture with regard to this requirement must be demonstrated. In Appendix A, “Simulation Architecture,” of the DOD Modeling and Simulation Master Plan there is a list of ten technical goals for a generic architecture which we have used for an analysis of the technological risks.

8.2.1 Entity-Level Representation

DIAS is an object-oriented design. The technological risk is that associated with equating entities with objects. The risk for this goal is low.

8.2.2 Interoperability

Within the DIAS architecture, a model is integrated when the knowledge-base has been appropriately updated to accept the model. This not only assures interoperability, but enhances the coordination between potential disparate databases. Lack of such coordination is the primary cause for the marginal interoperability noted in current DIS based exercises. The risk for this goal is low. The risk associated with Verification Validation and Accreditation is low.

8.2.3 Reuse

DIAS objects are designed to be reusable from one simulation to another. External to this domain, there may be transformation issues related to individual implementations. The risk for this goal is low.

8.2.4 Portability

DIAS is a prototypical system and is reliant on minimally portable products, *e.g.*, Parallel Virtual Machine or CORBA. However, the technological risk associated with development of a production system from such a prototype is low.

8.2.5 Distributed Operation

The risk associated with this goal is usually considered within an implementation context. At one extreme, distributed operation may be conceived as involving heterogeneous networks or heterogeneous systems. Distributed operation within the architectural domain under which DIAS was developed presents a low risk. Outside of this domain, the risk would be the subject of a specific analysis.

8.2.6 Legacy Interface

DIAS has been designed for integration of legacy-type models. Consider a legacy model as being a DIAS model. Incorporation of new models requires establishing a set of rules to configure the Context Manager to accept the model. The technological risk associated with each new model is dependent on the complexity of the model, it's data requirements, and how well its behavior is documented. The risk associated with incorporating new models is conservatively estimated as being moderate.

8.2.7 Scalability

Scalability is typically a measure of the match between the inherent parallelism of a computation and its operational environment. Specifically, computation that is inherently parallel and is executing in an environment that is well suited for its paradigm will be highly scaleable. The dependencies in determining scalability concern the match between the inherent parallelism associated with the algorithm (and its implementation) and the target computer architecture. Generalized scalability of software systems; *i.e.*, scalability irrespective of the target architecture, is a topic of current research. Without a definition of a specific target architecture, it is difficult to ascertain a meaningful measure of the scalability for a particular application.

8.2.8 Broad Functional Applicability

This is also a current research topic and associated risk is high. Because federations can be assembled according to overall federation context, the architecture which supports this federation building process (DIAS) will have broad functional applicability.

8.2.9 Technological Evolvability

The risk associated with the technological evolvability of object-oriented design is judged as being low.

8.2.10 Use of COTS/GOTS Software

The use of COTS and Government Off-The-Shelf (GOTS) software can clearly be advantageous to a program because it avoids the un-necessary development costs as well

as bringing some accreditation from the use of "standard" software. DIAS has been designed to be able to incorporate COTS/GOTS software, and its first instantiation as DEEM has already successful integrated a number of COTS software packages.

However, there are two risk issues associated with the use of COTS/GOTS in any system. The first is that there can be distribution problems associated with the use of COTS/GOTS software. COTS software typically has to be licensed and when the integrated system is distributed to users, the issue of separate or runtime licenses must be addressed. The second issue involves obtaining support for the COTS/GOTS packages in case of errors or upgrades. Support for COTS software may not always be assured when the software is purchased and may require additional expenditures. In the case of GOTS software, there is no promise that support may be available (or even that the original developers can be found!) In summary, the total degree of risk associated with the use of COTS/GOTS is difficult to quantify but a moderate level of risk seems to be an appropriate assessment.

9. SUMMARY

The Dynamic Information Architecture System (DIAS) is a highly flexible software environment for modeling, analyzing, and displaying information from complex federations of models and information processing applications. DIAS has been sponsored by a number of agencies to support understanding the role of the environment in military and civilian operations through the development of DEEM, the Dynamic Environmental Effects Model. Although current DIAS applications have focused on environmental applications, the DIAS architecture is robust and can be applied to a wide variety of applications.

DIAS is a fully object-based modeling system which supports distributed, dynamic representation of interlinked processes. DIAS simulations can be performed at variable spatial and temporal scales of resolution and aggregation. The DIAS Object Library currently contains a large number of objects to represent a wide variety of natural and artificial elements of the environment that have been generated as a result of the developing DEEM application.

DIAS utilizes an expert system-based Context Manager to determine the requirements for a given simulation. The DIAS Context Manager supervises the creation of transient or persistent Projection objects for interprocess data transfer. The Context Manager also supervises the analysis process via a global Event Manager, setting up the mechanism by which events are posted, activated, and received and acted on by DIAS objects.

DIAS has been developed to provide for the easy integration of legacy type models, such as physics models from DOD laboratories and agencies. By having the ability to integrate legacy models, one reduces development costs and provides a degree of accreditation because the legacy models are generally well tested and validated. In addition, DIAS has been developed to utilize Commercial and Government Off-The-Shelf software packages as well as state-of-the-art software technologies.

Appendix A
Glossary of DIAS Terminology

Frame	The object class representing the area of interest in a DIAS simulation
Entity	An object (*e.g.*, such as a river, forest, road, atmospheric layer, ship, air craft, plan of action, Air Tasking Order, etc.) with many possible aspects of dynamic behavior
Aspect	An object class that is an expression of a single aspect of an Entity's behavior. An example of an aspect is surface exchange between the atmosphere and the surface.
Model	An object encapsulating the means of assessing a particular model or information processing application, and a formal description of its function via a network of Process objects.
Process	An object class that is a single portion of a model or information processing application, either external to on embedded within the DIAS framework, which addresses a particular Entity Aspect.
Context Object	An object class that is a collection of metadata which governs the protocol by which models or information processing applications operate on the DIAS state variables.
Projection Object	A DIAS object class that is a transformation of DIAS data for a specific use. A projection object can either be transient or persistent.

Appendix B
Description of Entities Developed for the DEEM Instantiation of DIAS

B.1 INTRODUCTION

As of mid-1995, the best developed illustration of the use of the DIAS is the Dynamic Environmental Effects Model (DEEM), which has been create to support the modeling of synthetic environments across a broad range of scale. A prototype of DEEM is operational at the USAF Global Weather Center, JCS/J-8, and at Argonne National Laboratory. Demonstrations of the DEEM System are available upon request.

In this Appendix, we provide a detailed description of the objects and entities that have been implemented within DEEM as an illustration of how DIAS can be used in practice and of the way in which entities are assembled inside a DIAS Frame to meet the needs of a given federation of models and information processing applications. Although most of the entities described in the Appendix are "environmental" in nature, a number of them, such as the Artifact, Structure, Vehicle, and Transport Net Objects, have general applicability across a broad spectrum of applications. In particular, the Transport Net Entity is very generalized and is capable of supporting the movement of a range of real and conceptual objects.

B.2 THE DEEM OBJECT/DOMAIN MODEL

The DEEM instantiation of DIAS is a software framework intended to facilitate holistic multidisciplinary modeling of terrestrial, aquatic, and atmospheric processes. These processes are modeled in DEEM as interrelated actions caused by and affecting the collection of diverse real-world objects represented within the Frame. The modeling domain is flexible, determined by the environment objects available within the DEEM prototype and by the collection of models which have been gathered by users to address their modeling concerns.

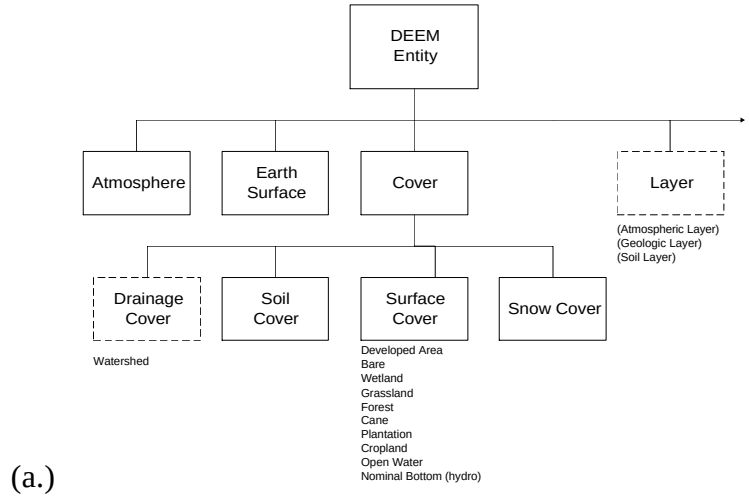
DIAS's inherent flexibility makes it somewhat difficult to draw an envelope around the domain it addresses. To illustrate some of the detailed aspects of how DIAS works, we have included a description of the environment object types in the DEEM prototype. In the discussion that follows, it is noted that many of the objects that have been created for the DEEM prototype have applications in studies without a specific environmental thrust. As a result, the "DIAS" and "DEEM" labels will be used interchangeabily. In addition, since the usefulness of any advanced modeling system also depends upon the availability and cost of data acquisition and preparation, we discuss in subsection B.2.2 some of the current data ingest pathways and tools that have been implemented for the DEEM prototype.

B.2.1 DEEM Environmental Entities

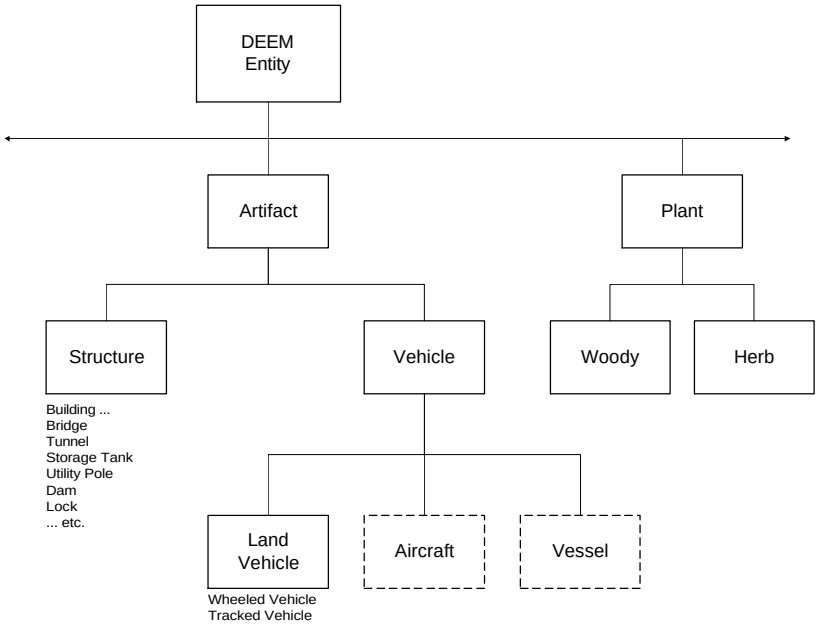
The mix of environmental Entity types available or under development in DEEM is described in this subsection. The ensemble of environment object (Entity) types presently available within DEEM is to a great extent a reflection of specific modeling concerns of

the sponsors of the DEEM development effort to date, and should not be taken to be comprehensive.

A simplified schematic diagram of the various object classes derived from the DEEM Entity class is shown in Figures B-1 (a.), (b.), and (c.) In these Figures, boxes with solid borders represent completed classes; boxes with dashed borders are under development. Lists of class names under a box indicate additional object subclasses derived from that class. A class name appearing at more than one location in the hierarchy indicates multiple inheritance -- the class is derived from more than one parent class. Some explanatory notes on the Entity subclasses shown in Figure B-1 are presented below.

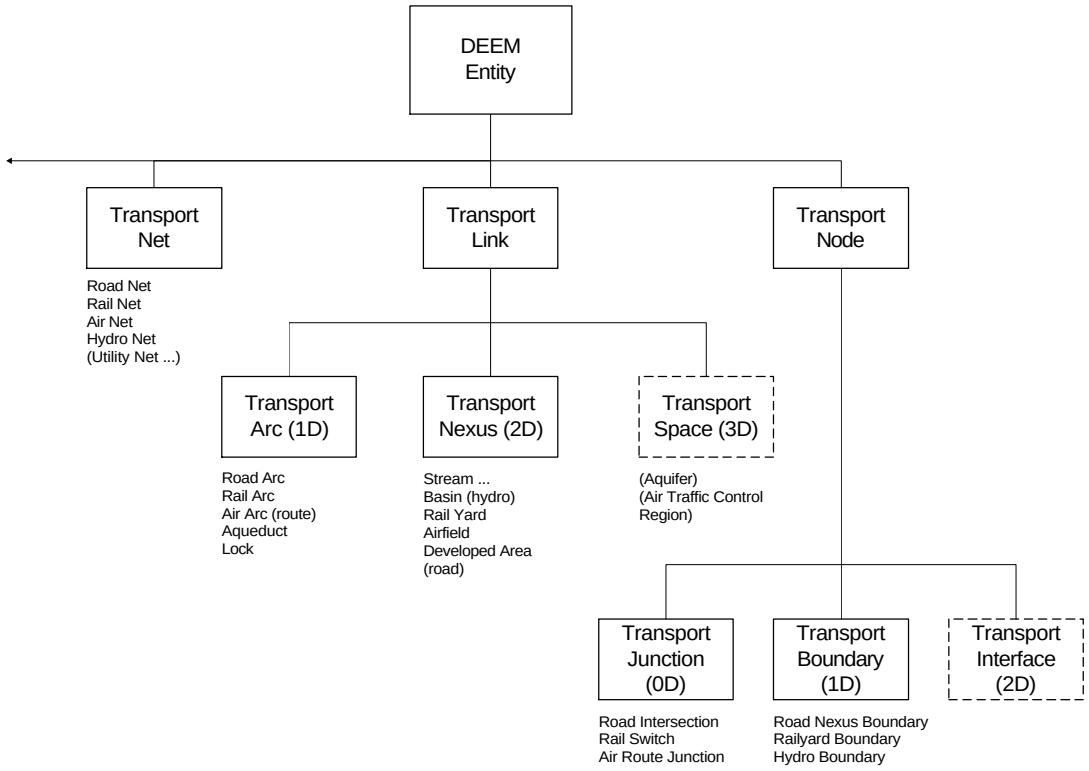


(a.)



(b.)

Figure B-1. Taxonomy of the DEEM Entity Class Objects. (Objects in Dashed Boxes Are Under Development) (a.) The Natural, (b.) the Artifact and Plant Branches



(c.)

Figure B-1. (Cont.) (c.) Transportation Branch.

Atmosphere: A dynamically reconfigurable 3D representation. State data include grids of meteorological parameters (wind components, temperature, humidity, clouds, precipitation, etc.), as well as airborne obscurant information and data needed to address atmospheric dispersion (*e.g.*, turbulent diffusivities).

Earth Surface: container class for all surface-covering Entities (Covers) in a Frame. Includes 2D grids of surface/moisture and energy atmospheric fluxes. An Earth Surface object:

- Is owned by a Frame object;
- Maintains lists of Cover objects of each type for the DEEM Frame;
- Applies precedence rules to "clip" polygonal 2D extents, and maintains hierarchy of "child" Cover objects;
- Provides data for gridded Projections of vector-based surface features.

Cover: Superclass for all objects considered to have 2D geographical extent. A cover carries type and precedence for proper processing by the Earth Surface.

Drainage Cover: A Cover object with hydrological significance. The only example thus far is a Watershed.

Soil Cover: Cover object carrying 2D extent of a soil layer. (Note that soil and subsoil layers could be expressed explicitly in 3D, or in 2D with a depth annotation. The way this is done depends somewhat on the application, and is a current research topic at Argonne.)

Surface Cover: Superclass for objects which can be considered to directly cover (drape across) the terrain surface. Subclasses include Developed Area, Bare, Wetland, Grassland, Forest, Cane, Plantation, Cropland, Open Water, and Nominal Bottom objects. The Open Water and Nominal Bottom objects were created to address the peculiar problems that can occur in trying to characterize a drainage feature, such as a Stream or Basin, which may have a fluctuating extent of open water surface (which are maintained in the Open Water object). The Nominal Bottom object is that portion of a drainage feature which is "normally" underwater, such as the region underlying an "Open Water" feature in a DMA Interim Terrain Data (ITD) dataset. In ITD, among other data sources, characteristics of lakebeds or seabeds are not specified. The Nominal Bottom must be created and assigned surface characteristics so that data voids do not occur if a water body recedes from its nominal banks during a simulation and portions of the Nominal Bottom region are exposed.

Snow Cover: This is a special subclass of the Surface Cover object which is used to address snow and ice fields. This class has the special characteristic that it overlies other Surface Cover objects. (All other Surface Covers except for Open Water are mutually exclusive in 2D extent.)

Layer: A Layer is a superclass for "stratified" objects which have a 3D extent in the Frame. Subclasses will include Atmospheric Layer (carried in lists in the Atmosphere Entity), Geologic Layer, and Soil Layer (see comment above on Soil Cover).

Artifact: This is the superclass for all manmade objects.

Structure: This is the superclass for all stationary Artifacts. It can be used to model buildings and installations of any type, but generally a subclass should be derived to handle different major types of structure. The Structure superclass makes use of a number of subordinate object classes:

- Section - an undifferentiated major subdivision of a Structure.
- Panel - a portion of Structure wall, roof, etc.
- Support - a structural support for Structure -- column, beam, etc.
- Trim - a non-structural element of a Structure.
- Opening - an aperture and associated support and trim for a window, door, etc. in a Structure.
- Membrane - a substance covering an Opening, such as a door, louver, or window glass.

These subclasses have been useful in some earlier structure modeling, but more special purpose subclasses may be needed to support serious civil engineering applications in DIAS.

Vehicle: This class is the superclass for any Artifact that can (and is intended to) move. Obviously, many subclasses will be derived from Vehicle. The DEEM Land Vehicle subclass presently carries all of the vehicle parameters required by the very high fidelity NATO Reference Mobility Model, developed by the USA Waterways Experiment Station.

Plant: Superclass for all objects which represent individual botanical plants.

Woody: Object representing a woody plant -- a tree or shrub. This class is quite well developed and utilizes internal models which address growth and visual appearance at various levels of detail. The Woody class is supported by a number of subordinate classes:

- Woody Type - carries the characteristics of the plant's species or variety.
- Growth Unit - a branch or twig "internode" between successive buds, a fundamental structural and botanical subdivision.
- Bud - an individual bud, carrying branching geometries and other information future leaves, flowers, and internodes which may develop from it.

Herb: An object representing an herbaceous plant. The Herb object is generally comparable to the Woody object class in terms of complexity.

Transport Net: Superclass for all Entities which facilitate the transportation of an object or a substance across a geographically distributed network. This highly abstract concept supports subclassed networks of many types, including:

- Road, Rail, Air, and Navigation transportation networks;
- Drainage (Hydrological) networks;
- Utility networks of various sorts -- pipelines, transmission lines, etc.

In a Transport Net, transport occurs along Transport Links which are connected by Transport Nodes. The concepts of Link and Node are also highly abstracted and subject to several topological variants (subclasses), subject to the constraint that all transport takes place along links and links are connected at nodes. Figure B-2 shows an example of a road net that has been constructed from various transport objects.

Transport Link: Superclass for those elements of a transport network along which transport from place to place occurs. Classically, links are thought of as one-dimensional objects (line segments). In DEEM, however, they can take on one, two, or three dimensions, depending on the type of network and the type of link represented.

Transport Arc: This is a "normal", one-dimensional link, connected to a node at either end. Subclass examples include:

- Road Arc - a section of road between intersections.
- Rail Arc - a section of railroad between rail junctions.
- Air Arc - a specific air route or flight path.
- Aqueduct - a strictly hydrological link.
- Lock - a hydrological and navigation link).

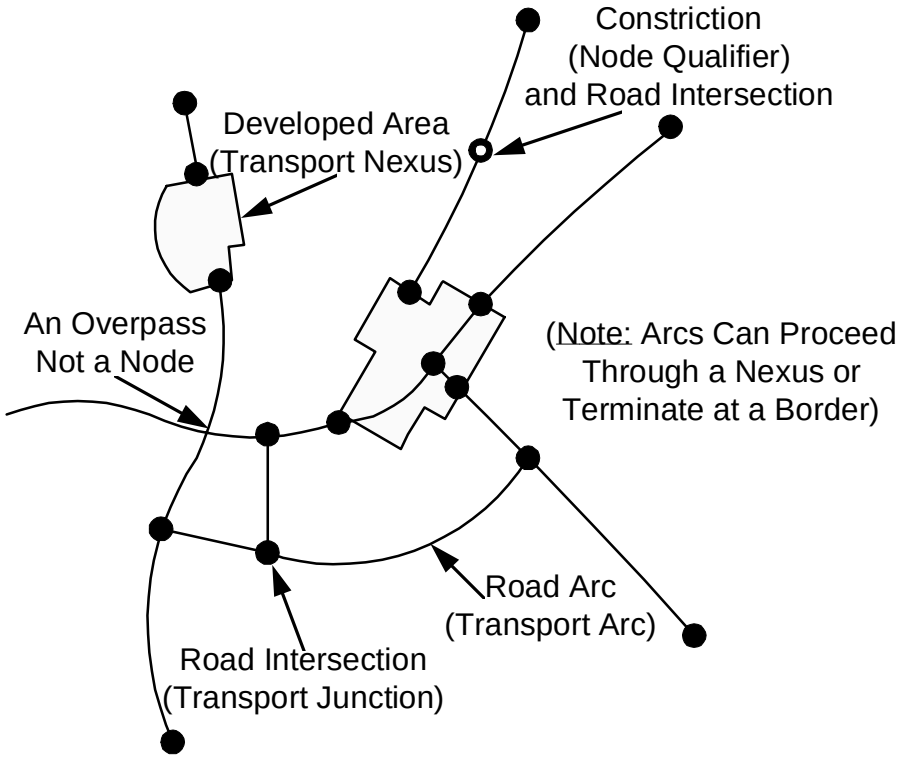


Figure B-2. Schematic Representation of a Road Net That is Constructed Out of DEEM Objects.

Transport Nexus: This class is essentially an "area link". It represents transport link elements which cover an *area*, and can diffusely provide transport paths between any number of nodes embedded in the area or attached at the border of the area. Subclass examples include:

- Developed Area - an area assumed to have a pervasive network of roads which is not resolved as separate links in the DEEM data. This class is exceptionally useful in allowing construction of usable, fully connected road networks from the types of data normally provided by a Geographical Information System (GIS), in which urban areas are frequently represented as undifferentiated polygons and major roads end at their borders.
- Rail Yard - an area assumed to carry an unresolved sub-network of trackage.
- Airfield - an area assumed to carry an unresolved sub-network of taxiways and runways.
- Stream and Basin - areas covered by open water, connected to other such areas at one-dimensional Hydro Boundary nodes (see *Transport Boundary*, below).

A significant feature of the Transport Nexus object class is its ability to optionally carry a sub-network at a higher level of detail. The more detailed sub-network must possess the same topological connectivity to other elements of the main network as does the Transport Nexus. For example, consider a Rail Yard (Transport Nexus) through which a single

main rail line passes. Any sub-network of railroad tracks within the Rail Yard boundary which provides connections to the main line at the same two points at which the main line crosses the yard boundary is a valid sub-network. It is not difficult to envision other potential uses for the sub-network capability. For example, a Stream object representing a section of a braided stream which is normally treated simply as a single channel might reveal detailed information on its network of actual channels if viewed in the context of a detailed engineering analysis of a river crossing.

Transport Space: This is an extension of the Transport Nexus concept from two to three dimensions. A Transport Space is a 3D volume within which transport from place to place may occur. Examples of potentially useful subclasses of Transport Space (not yet implemented) include:

- Aquifer - a 3D transport of ground water within a bounded permeable region.
- Air Traffic Control Region - a 3D airspace within which rules for air movement are established. It could also be used for 3D air corridors, air battlespace regions, etc.

Transport Node: This is a superclass for those elements of a transport network which connect transport links. Classically, nodes are thought of as zero - dimensional objects (or points). In DEEM however, they can take on zero, one, or two dimensions, depending on the type of network and the type of node represented.

Transport Junction: This is a "normal" zero-dimensional node, connecting any number of links, at least one of which must be a subclass of Transport Arc. Subclass examples include: Road Intersection, Rail Switch, and Air Route Junction.

Transport Boundary: This class represents a one-dimensional (line segment) boundary node connecting two-dimensional Transport Nexus objects. The most common subclass is a Hydro Boundary which is the node joining contiguous Stream or Basin water bodies. Other subclasses include Road Nexus Boundary (line joining two contiguous Developed Area objects, for example), and Railyard Boundary (*e.g.*, line joining two contiguous but distinct Rail Yards).

Transport Interface: This class is the complement to the Transport Space object. It represents the two-dimensional area of contact between two contiguous Transport Space volumes.

Two other real-world object classes not derived form the Entity class are worthy of mention. The following classes, which are not shown in Figure B-1, also support the Transport Net concept which has been presented above.

Arc Pathway: This is a superclass which handles all Entities which assist a Transport Arc in completing its connections between its two endpoints. Subclasses include:

- Bridge (also derived from Structure).
- Tunnel (also derived from Structure).
- Ford (on-route).
- Ferry (may contain a list of Vessel objects).

Node Qualifier: This is a superclass for any additional information, including identification of additional objects, which may be associated with a Transport Node. Thus far, the only subclasses are those covering specific features carried in DMA Interim Terrain Data transportation files:

- Drop Gate Road - notes the existence of a gate across a road link.
- Drop Gate Rail - notes the existence of a gate across a rail link.
- Constriction - notes the existence of a constriction point along a road right of way.

B.2.2 DEEM Data Ingestion Pathways

DEEM simulations require data to describe the Frame, the area of interest for the simulation, and the weather conditions. There are a number of standard sources for these data, but seeing that DEEM is being used in simulations for different places around the world where the data quantity and quality can be highly variable, different of data ingestion pathways have been and are continuing to be developed in order to provide maximum flexibility in meeting the data needs of a simulation. In addition, we are examining how to provide the user with qualitative information on the "quality" of the data sources. Figure 6 gives a conceptual representation of this issue.

In one DEEM application, we are determining the soil moisture strength in order to make estimations of travel times for different types of vehicles of different types of terrain and under varying environmental conditions. In order to make the required calculations, data on the soil moisture conditions (among other parameters) must be used. Depending on the location under study, soil moisture data may or may not be available. However, DEEM must have these data in order to make the calculations, so DEEM has been given a set of rules that outline a series of options if the best kind of data, field data, are not available. As shown in Figure B-3, the quality or confidence in the data decreases as one goes down the list of options. At this point, it is not envisioned that DEEM will attempt to give a numerical measure of data quality, but will instead merely report back to the user where the data came from and let him or her draw their own conclusions.

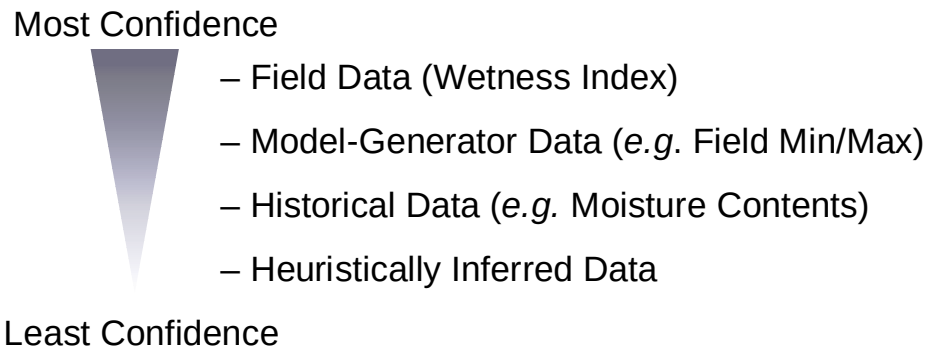


Figure B-3. A Conceptual Representation of Data Quality Issues That Are Being Considered With DEEM

In terms of the data used to define the Frame, the native form for the spatial locus of DEEM objects is vector-based. Arbitrary raster-based representations are created, as needed, via DEEM Projection objects.

All DEEM Entities and Aspects of all types (thus everything that can populate a Frame) will be able to generate their own edit screens, by automatically parsing each object class' header (definition) file. Each Entity will, when selected from a map, realistic image, or table, present itself for inspection and editing.

Essentially any type of DEEM object can read its defining parameters and locus from an external ASCII file on request. This feature has made it possible to build some DEEM Frames for simulation largely from scratch, when we had no access to digital GIS information.

Table B-1 summarizes the data ingestion pathways that are available with DEEM for the DEEM Frame and weather. The table lists those ingestion pathways that are currently implemented, under development, or proposed.

B.2.2.1 Topographic Data

The DIAS internal terrain representation is the Terrain Surface object for a Frame -- a variable-resolution grid of recursively subdividable cells. Several types of DEEM 'Bump' object (*e.g.*, embankments, craters, canals, quarries) have the attribute of being able to automatically modify both terrain elevations and local terrain grid resolution. (Finer resolution is generally required to properly represent manmade earthworks than is carried in the digital terrain.) DMA DTED and USGS Digital Elevation Map (DEM) can be read directly into a DIAS Terrain Surface.

Table B-1. Data Sources for the DEEM Frame and Weather

Data Type	**Available or Planned Data Sources**
Topographic	DMA Digital Terrain Elevation Data, USGS Digital Elevation Map
Surface Cover	DMA Interim Terrain Data* (ITD), USGS Digital Line Graph (DLG), Remote Sensing Products†
Transportation	ITD, DLG, Dept. of Commerce Topographically Integrated Geographic Encoding and Reference (TIGER) Data, National Highway Database
Surface Drainage	ITD, DLG, Remote Sensing Products†
Soils	ITD, World Soils Database‡, USDA National Resources Inventory Primary Sampling Unit‡, USDA National Soil Geographic Database‡, USDA Soil Survey Geographic Database‡, USDA State Soil Geographic Database‡
Weather	Synoptic, Airways, and METAR Surface Observations†, Radiosonde Reports, Gridded Forecast Products

*ITD are Now Being Produced in a Vector Product Format
†To be Implemented
‡Proposed Implementation

B.2.2.2 Surface Cover Data

The DMA ITD "Vegetation" Thematic File can be read to create and load appropriate surface-covering objects, such as Forest, Plantation, Grassland, etc. The same capability exists for USGS Digital Line Graph (DLG) data. DLG's internal schema is substantially different from ITD, for all data types. (Note that DLG is the preferred data interchange format for the Corps of Engineers' GRASS system.)

The same data ingestion capability exists for arbitrary raster files of surface classification index. DEEM surface cover objects of appropriate types are created by mapping the surface classification index to a DEEM object class. The raster data are converted to vector polygon loci for the newly created DEEM objects. This ingestion pathway is intended to support input of surface cover maps resulting from ANL's neural network-based workbench for interpretation of multispectral imagery results initially, but will work equally well with essentially any raster inputs. Also, the capability will not be limited to surface cover objects only; it can be used for soils, microterrain, etc.

B.2.2.3 Transportation Data

The DMA ITD "Transportation" Thematic File can be used to create and load road, rail, and air network objects. (Note that rail, road, and air networks, as well as drainage, navigation, and utility networks, are derived from a transportation network superclass.) The same capability is available for USGS DLG data.

B.2.2.4 Surface Drainage Data

The DMA ITD "Surface Drainage" Thematic File can be used to create and load both drainage network and water cover objects in DEEM. The same DEEM capability exists for USGS DLG data.

B.2.2.5 Soils Data

The DMA ITD "Surface Materials" Thematic File can be used to create and load soil cover objects in DEEM. Creation of data ingestion procedures for other sources of soils data is planned. Other soils data sources under consideration include:

- A global soils database based on United Nations Food and Agriculture and US Department of Agriculture soils classification schemes is under development and available in GRASS format. The resolution is somewhat coarse, $1° \times 1°$, but may have applicability in some simulations.
- USDA National Resources Inventory (NRI) Primary Sampling Unit (PSU) Geographic Database. This database contains soil characteristics, land cover and use, erosion, land treatment, etc., for non-Federal U.S. land. It is available for 1982 and 1987 and is updated every 5 years. The data are in DLG format.
- USDA National Soil Geographic Database (NATSGO). This database contains soil spatial and attribute data. Using this database requires access to Soil Interpretations Record relational database. Coverage is for the U.S. and Caribbean. The format of the data is DLG.

- USDA Soil Survey Geographic Database (SSURGO). This database is similar to NATSGO, but has finer resolution although spottier coverage.
- USDA State Soil Geographic Database (STATSGO). This database is coarser than SSURGO, but has better geographic coverage. The format of the data DLG.

B.2.2.6 Microterrain Data

The DMA ITD "Surface Configuration" data (direction-independent maximum slope in irregular polygonal regions) can be read into DEEM Microterrain objects. The DEEM Microterrain objects will be designed to accept much more information than just maximum slope, for other applications besides DOD trafficability analyses. Examples might include roughness statistics from LIDAR, fractal dimensions of the surface, etc.

B.2.2.7 TIGER Data

DEEM has an automated ingestion pathway for Topographically Integrated Geographic Encoding and Referencing (TIGER) System data from the U.S. Dept. of Commerce, Bureau of the Census. TIGER data on installations (airports, hospitals, etc.), population, political boundaries, shorelines, census tract street locations, and other geographic information can be brought into DEEM objects for areas of interest in the U.S.

B.2.2.8 Weather Data

The types of weather data that can be required for a simulation include conventional surface and upper observations, weather forecast products, daily summaries, and climatologies. The ingestion of these data will take into account the differences between alternate data sources, such as synoptic, Airways, and METAR data for surface observations.