999

N90-22315

Using Expert Systems to Implement a Semantic Data Model of a Large Mass Storage System

Larry H. Roelofs
National Space Science Data Center
Computer Technology Associates
Rockville, Maryland 20852

William J. Campbell
National Space Science Data Center
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

The successful development of large volume data storage systems will depend not only on the ability of the designers to store data, but on the ability to manage such data once it is in the system. Our hypothesis is that mass storage data management can only be implemented successfully based on highly 'intelligent' meta data management services. Such services would allow database administrators and users to manipulate, update, and access data, and related information and knowledge in a logical manner and yet is powerful enough to support the performance needs of a large mass store system. Historically, there have been attempts at building data management services for very large volume data systems, however, when the amount of data being managed got large the meta database itself failed as a consequence of its own size and complexity.

There now exists a proposed mass store system standard proposed by the IEEE, that addresses many of the issues related to the storage of large volumes of data, however, the model does not consider a major technical issue, namely the high level management of stored data. However, if the model were expanded to include the semantics and pragmatics of the data domain using a Semantic Data Model (SDM) concept the result would be data that is expressiveness of the Intelligent Information Fusion (IIF) concept, the result would be data that is organized and classified in context to its use and purpose. The implementation of a SDM requires the application of AI and related computer science technologies such as object oriented representation, property inheritance and rule based decision making. Presently there does not exist unique software for developing SDM's that address the complex representation of data meta data and related information and knowledge.

This paper presents the results of a demonstration prototype SDM implemented using the expert system development tool NEXPERT OBJECT. In the prototype, a simple instance of a SDM was created to support a hypothetical application for the Earth Observing System, Data Information System (EOSDIS). The massive amounts of data that EOSDIS will manage requires the definition and design of a powerful information management system in order to support even the most basic needs of the project. The application domain is characterized by a semantic like network that represents the data content and the relationships between the data based on user views and more generalized domain architectural view of the information world. The data in the domain are represented by objects that define classes, types and instances of the data. In addition, data properties are selectively inherited between parent and daughter relationships in the domain. Based on the SDM a simple information system design is developed from the low level data storage media, through record management and meta data management to the user interface.

Background

In the past decade, operations and research projects that support a major portion of NASA's overall mission have experienced a dramatic increase in the volume of generated data and resultant information that is unparalleled in the history of the agency. This information glut is growing nonlinearly due to the increasing number and quality (higher resolution) of sensor systems and is expected to continue to accelerate in this fashion for the foreseeable future. The effect of such large volumes of data is that without a significant improvement in information technologies there is no assured way that desired data can be managed, identified and accessed.

If nothing is done to reverse this process, large data archives will evolve that may contain very valuable data for which there is no easy way to find desired data. The reason being is that the search times to find desired files will be prohibitive as selection will require human interpretation and user queries will not map logical into the data being stored. For example, in the Earth Observing System (EOS), which is a major new NASA project supporting the "Mission to Planet Earth" program, data volumes produced could be as high as four Terabytes per day, with much of the data being spatial in nature. [1] If such volumes are accumulated over the life of the mission (planned for 15 years) the total data volume may be as high as 30 Petabytes (30,000 trillion bytes) when backup and data reprocessing are included.

In addition to the amount and kinds of data, the number of professionals in the application disciplines is not expected to increase significantly enough to resolve this data/information selection problem. Thus the dilemma arises that the amount and complexity of data and information system will exceeded the ability of scientist and engineers to understand and take advantage of them. [2] Based on the scope, expected growth and dominance of the data volume problem, it is anticipated that the future ability of NASA to say apace with the data it collects from space and Earth science programs will be significantly affected by its ability to manage and use such data in an efficient and timely fashion.

The Limitation of Information Systems for Mass Storage

Present database management systems cannot support the low level performance needs of a mass store system, and at the same time provide the capability to the user to identify, select and access desired data quickly and easily. The result is that the design of an information system for a large mass storage system must be separated both operationally and functionally into two elements: (1) Low Level Record Management and (2) High Level Meta Data Management and User Interface (Figure 1). The low level record management component must be biased toward performance and must focus on storing and managing file storage records at the device or hardware level. The high level meta data management system must interface to both the low level record management and user interface components and provide a robust data management and access to the files in the mass store system. In addition, such a system should allows the user to access the data without understanding the data domain architecture, data content or data query language.

The rationale for separating the low level record management from the rest of the database system is that the mass store system performance and the large number of individual records limit the kind of

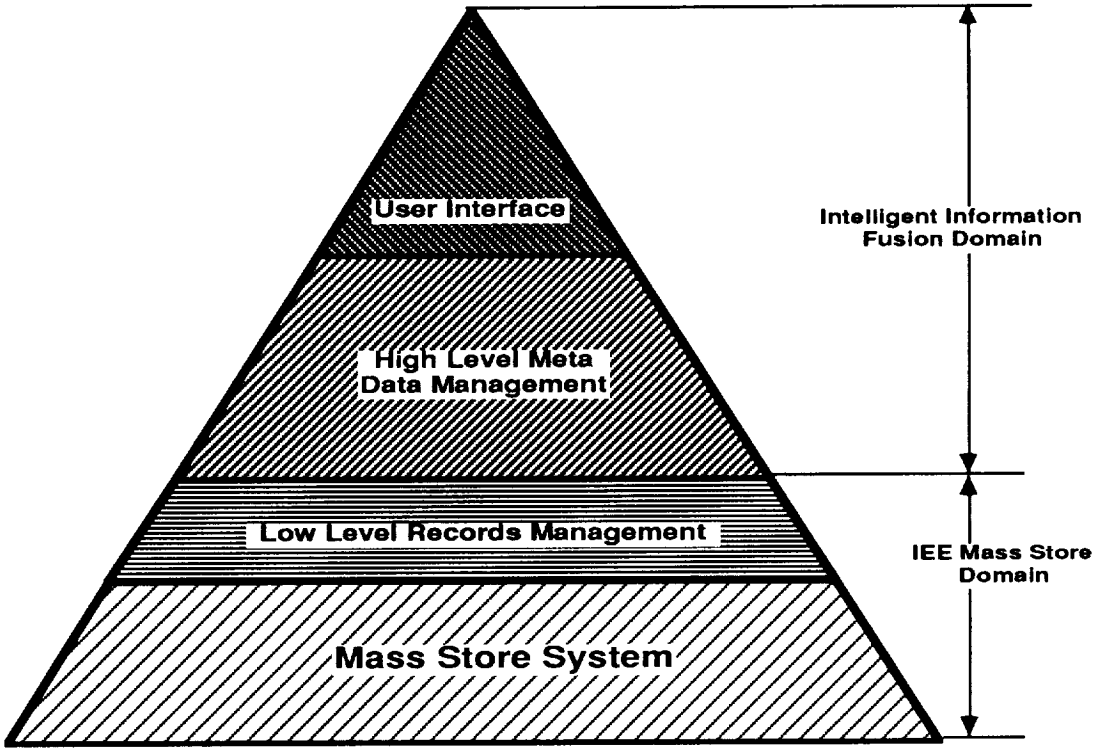

Figure 1
Mass Store Information Management Functional Architecture

data management system that can be implemented. Such a system must be simple in order match the access and I/O performance of the storage hardware and at the same time be easy to update and manage. Since the low level meta data system is not intended to support user access, it will be necessary to implement a separate high level meta data system that can interact with the low level meta data system and at the same time interact with and support the user interface. Using present database technologies the system would most likely use a relational database system, even though it would impose limitations on the capabilities of the mass store system in terms of data access and ingest [3]. There three reasons for this selection: the technology has significantly grown in performance and capability over the past ten years, such that it now provides significant benefits over databases that are based on the other two data models (network [4] and hierarchical [5]); it is a fairly mature technology that is well understood, easy to implement and easy to maintain; there are a large number of commercially available software systems that run on most of the popular computers available today.

However, once the decision is made to implement the high level meta data management system using a relational database there are significant representation and data manipulation issues inherent in relational technology, namely:

1. Database architecture based on a flat file structure
2. Database abstraction limited to aggregation
3. No way to capture and manage meta knowledge (procedural and control information)
4. No way to support the representation of time

5. No way to deal with class-type and derived data where it is required to support the generalization of the class
6. Difficult query language based on relational calculus that is not easy to use or understand
7. The database requires normalization in order to function efficiently.

The first issue can be explained by considering that in a relational database management model, the tuples in a relation correspond to the records in a file and the tables are simply flat lists that can be manipulated only by table joining. [6,7] The second, third, fourth and fifth issues are a consequence of the mathematics that the database model is based on. Essentially, the only structure that a relational database supports is aggregation (one to one, one to many and many to many), not class, type or time. [8] The sixth issue is also a consequence of relational calculus which requires very precise syntax and structure in order to form a mathematically correct query. Such a query language usually requires a rather sophisticated user to form even the most simple requests. [3]

The last issue is the result of the need to optimize the database design in order to maximum performance and minimize maintenance. [9] However, normalization usually results in most of the semantics and pragmatics of the data domain being removed. The removal process is the consequence of redundant, but logically related, attributes being deleted from common relations (tables), such that an attribute exists only once in the database. This removal process logically fragments the database in a manner that makes the database difficult to understand, to the casual user, without a design map (e.g. E/R diagram). If this is combined with the lack of expressiveness of the Database Manipulation Language (DML), it is no wonder that few large database systems are understandable by the users' except through some simplistic interface that more often than not severely limits the users' access to the data.

Given the above, the only recourse is to put all the semantics, pragmatics, procedural and operational knowledge into a user interface which must be totally customized to accommodate each application or user view. Since the interface is usually implemented after the database has been designed and populated, its architecture tends not to be related to the database architecture in any way.

A Proposed Solution

Considering the data volume problem, it would appear that large mass store systems will be needed to store the flood of data that will be collected over the next fifteen to twenty years. However, traditional database technologies will not be able to support the high level meta data management needs of such a system, and at the same time have the performance capability necessary to handle low level file management. Consequently, alternative information management strategies that implement the management and access of data, meta data and supporting information and knowledge in a coherently structured manner must be employed. Such a structured approach to data and information management we call Intelligent Information Fusion (IIF). The IIF concept is based on a layered architecture that supports the semantics, pragmatics and syntax of the data domain from the system, data and user points of view so that data can be input, managed and accessed in the most efficient and appropriate manner. A diagram representing the overall IIF concept, in the context of an EOS archive, is shown in Figure 2.

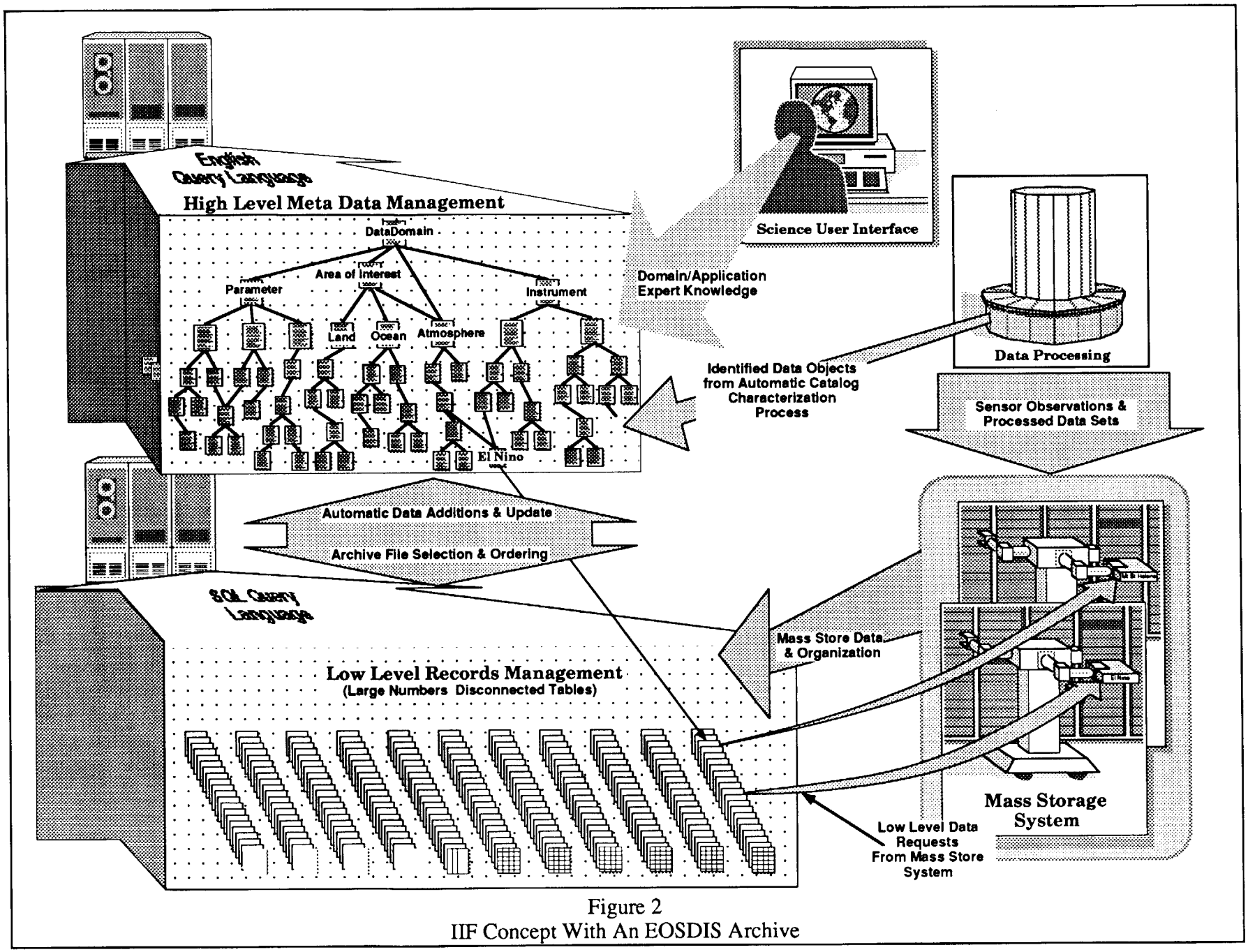

Figure 2
IIF Concept With An EOSDIS Archive

The IIF concept consists of three essential elements that are absolutely necessary to the successful implementation and operation of a large mass store system, these are:

1. *Semantic and Knowledge Based Representation*, which we call information synthesis is the process that captures the essence of the data enterprise (i.e. database domain) at all three levels of data representation (semantic, pragmatic, context, and syntax) for the generalized data domain. It addresses functional and operational views, from the highest level class structure, to intermediate meta data, to the lowest level data granule or file.

2. *View and Application Representation* supports multiple discipline, user, and application goals, based on the understanding and needs of the users.

3. *Automated Data Cataloging and Characterization* supports the addition and updating of data to the mass storage system while automatically updating the supporting meta data to reflect changes and additional data. [10]

The first element is the most important for the implementation of a large mass store system since it provides the organizational structure of the storage system, and its supporting record management system. In addition, it also provides input and direction to the design of the higher level meta data and user interfacing that supports the data access and ingest operations.

Information Synthesis and Semantic Data Models

Information synthesis, the context of the massive data storage problem, means the structuring and fusion of data, information, knowledge, and meta knowledge into a coherent structure that can logically be used for a particular purpose. The information synthesis concept is based on the semantic data model, enhanced with a knowledge base that can provide the control and operational strategies needed to support a complex information domain. The semantic data model was proposed in the late seventies by McLeod and King, [11] and refined by Potter, and Trueblood. [12] The model draws on research from the fields of Database Management (DBM) and Artificial Intelligence (AI) in performing data domain or enterprise modeling. The model is intended to address and overcome the problems of domain representation limitations that exist in the three traditional database models [13]; the hierarchical , network, and relational. All three of these models are basically record oriented with very complex data manipulation languages. [9] Their development was motivated and influenced by primitive file system implementation concerns that limit the size and volume of such systems. [14]

The development of models that provide more user oriented modeling flexibility, without being constrained by an implementation structure, was the primary goal of early semantic data model researchers [15, 16]. The resulting models provided a "natural" mechanism (e.g. similar to the way a user or system designer views an application) for specifying the design of a database which more accurately represents the data and relationships among the data than the traditional models. [12]

The IIF concept allows database designers to represent the objects of interest in the proper context and their relationships in a manner that more closely resembles the view that the user has of these

objects and relationships. [8] Consequently, designers are freed from the necessity of having to model a variety of different relationships with only one modeling construct, which typically results in the loss of their exact meaning. The end result of using limited constructs is that the application design falls short of modeling the user's actual situation or needs.

One of the most important features of the IIF concept is that it provides powerful abstraction constructs, such as generalization and aggregation, that are not found in the traditional data models. [17] Generalization allows the designer to group similar objects together in order to concentrate on the more general group object. Aggregation allows the designer to model an abstraction object based on the properties or attributes on the object. [8] Other constructs that deal with time and space have also been associated with semantic data modeling. [16]

In addition to generalization and aggregation, semantic data models are characterized by the notion of "derived or virtual data." [18, 19] Essentially, derived data can be thought of as data values which are not actually stored in the database but are produced or derived when needed from existing data and relationships. This concept is very important for mass store systems since it can potentially reduce the stored data volume significantly, because the storing of explicit values is unnecessary in as much as the relationships and means exist for deriving this data.

Semantic data models may be classified according to one of three general categories: relational, functional and semantic network. Models in the relational category are basically extensions to the relational database model. The functional models have an equally strong mathematical influence. In the functional model, the function is the primary notion used to represent and manipulate objects and relationships. The third model category is the semantic network model which has a close relationship to the semantic network knowledge representation formalism found in the AI field. However, unlike the early network models it was necessary to enhance the model with meta knowledge about the domain that describes how the data is manipulated and acted on, as proposed by Potter and Kerschberg. [12,20]

An Intelligent Information and Fusion Prototype

To prove the usefulness of IIF concept to the mass storage information system design, a demonstration prototype effort was undertaken using existing technologies. A cursory survey of the market place was conducted to identify any existing semantic database system tools. The result was that there are few commercially available systems, none which provide the robustness necessary for supporting large mass storage operations. Therefore, the only alternatives were either to develop new customized modeling tools or to use existing related technologies to implement such a tool. The second alternative was selected because previous experience has shown that there is a higher probability of success if several existing technologies can be combined to create a new single integrated tool.

An initial prototype system was conceived, based on a Macintosh II computer, using the expert system development tool, NEXPERT, built by Neuron Data Inc. NEXPERT provided many features that were necessary for semantic data modeling including:

- Frame based representation for creating data objects, data classes super classes that represent the content of the data domain
- Inference engine that supports rule strategies that employ pattern matching and forward and backward chaining for capturing the meta knowledge and procedures that support specific operational agendas
- Object representation that supports, properties, meta slots and multiple inheritance in both directions for supporting the passing of data properties and values between data objects and classes.
- Graphical interfacing for displaying, in a easy to understand manner, the complex network structure that represents the data domain. [21]

In addition to the development tool, it was also necessary to find or create a hypothetical database for modeling. The second option was selected since this was only a demonstration prototype effort. However, in order to understand how scientific databases are designed, two operational databases were reviewed, the Pilot Land Data System (PLDS) and the NASA Climate Data System (NCDS). Both systems store their sensor data off line and use the ORACLE DBMS for management of the meta data which is structured into two levels of abstraction. Namely, a Data Catalog that provides a general over view of stored data sets and a Data Inventory which provides detailed information about each stored data set. . In addition to the Data Catalog and Data Inventory meta database structure, both have an interface based on the Transactional Application Executive (TAE) software, with the NCDS enhanced to include some data visualization.

Besides understanding how existing data systems manage their data and meta data, a atmospheric scientist was interviewed to determine how she perceived the overall EOS data domain, as well as her specific areas of interest within the domain. The result of this effort was the definition of a simple structure for studying cloud cover, as well as the identification of three critical design goals that the model should be able to support. These goals were:

1. Where are all occurrences of a specific data object (e.g. episodic event, El Nino)
2. What data objects have been found for a specific observation.
3. What data objects have been found to occur, or change, over time for a specific location.

Semantic Model Design

Based on a review of the two data systems studied, the atmospheric scientist interview and a review of the EOS program, a hypothetical EOS data domain was formulated and is presented in Figure 3, Semantic Model Top Level View. The domain consisted of the three major data areas (Instrument, Interest Area, and General Parameter) that could be used to organize and find data as well as a more detailed structure to support a specific area of interest (e.g. atmosphere and cloud cover). The specific area of interest selected for the prototype was atmosphere, with the sub area clouds.

In addition to an overall data domain structure there were six key design considerations identified that needed to be accommodated in the data modeling process. These considerations were:

1. All low level data elements or granules had to inherit the property slots, and where appropriate, the values associated with the slots from its parent class object.

2. All first level data objects that related to a specific sensor observation file in the low level records management system must have properties associated with the object that represent the important features observed in the record, as well as all ephemeris and supporting ancillary data.

3. Rules are required to guide the ingesting and summarizing of data set header data (name, data, time, location, etc.) from the low level data objects to the object's class frame. [22]

4. All low level data objects and their associated sensor observation file have to be accessible by all logical paths.

5. Must be able to interact with a relational database system to allow it access and read low level data in the records management layer.

6. Must be able to deal with both the generalized data domain organization as well as supporting unique applications that use the data for some specific goal or purpose.

A data model based on the data domain concepts was formulated. The model included the three major information components that make up a data domain: the operational or database view, the application or semantic view and meta knowledge and control. In addition, the operational view was driven by three design considerations: the creation of a skeleton structure upon which the rest of the model could be implemented, the representation of the database domain that it supported the management, access and ingest of the actual data objects (individual sensor observations), and a strategy to support the access of data by instrument.

Construction of the Prototype

STEP 1 - Formulation and Characterization of the Operational View

The first step in the operational view representation was to identify the major class objects in the data enterprise and determine how they were related. Once this information was determined, it was input into NEXPERT by characterizing each class object with a frame that defined its daughter classes, as well as any properties that needed to be associated with the frame. In this step the top level database object was defined as, *Enterprise*, with primary super class objects (or domains): *Instrument*, *Area of Interest*, and General *Parameter*. In some since these three super class objects represent more than just the high level abstraction of its daughter elements because they must also include meta and procedural knowledge. The rationale for selecting the above super classes were:

- **Instrument** - The low level data needs to be stored and managed by instrument so an overall super class object must be created that deals with the topic area of instruments.

- **Area of Interest** - The user can select data by area of interest, where the focus is on some property within the area, sub area or component (e.g. cloud type at some time and/or location). This super class object is part of the application view which is the logical area where

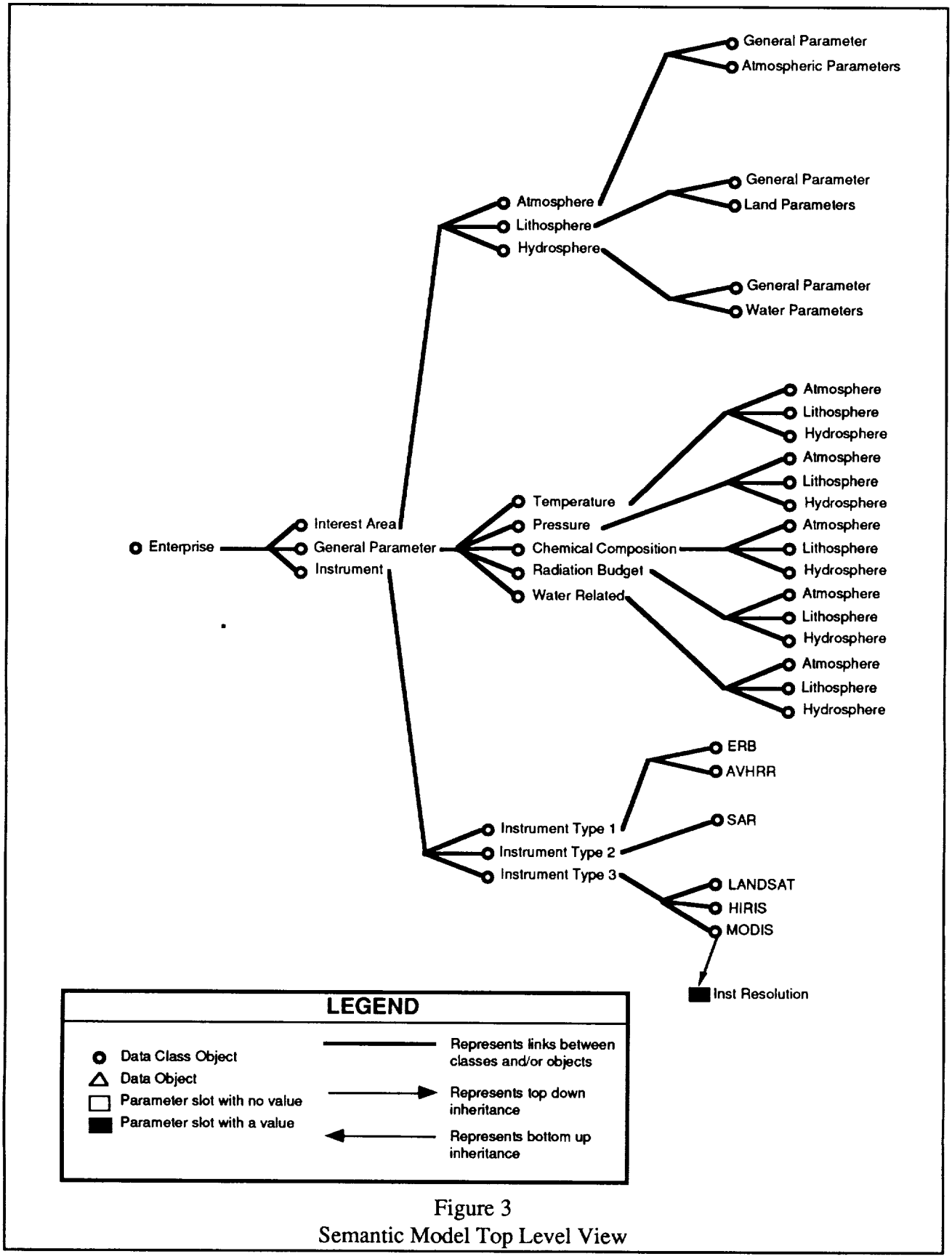

Figure 3
Semantic Model Top Level View

discipline, feature objects (data objects found with in a sensor observation) and supporting scientific information should be located.

- **Parameter** - Data can be selected by parameter across multiple disciplines or areas of interest (e.g. temperature for some location for land, atmosphere and water). This super class object is an alternative path that users require when the focus is more parameter and less single interest area oriented.

After the first super class objects were selected, each was expanded into several lower level daughter classes using generalized information gained from discussing the problem with scientists involved in atmospheric science and scientific database system design. A diagram of the result of this step is presented in Figure 3, Semantic Model Top Level View. Of particular importance is how the domain can only be represented by a network. This seemed rather obvious to a user until one tries to use the graphics tools of NEXPERT and finds that what is presented is a tree that is session dependent. However, upon closer inspection the super class object *General Parameter* was also found to be a subclass object of each of the subclasses in the *Area of Interest* super class object. Thus *General Parameter* serves not only as a high level super class object, but it is also a daughter object for each of the sphere objects (*Atmosphere, Lithosphere, Hydrosphere*) in the *Area of Interest* super class object .

Under the super class object *General Parameters,* representative physical parameters were identified and selected that appeared to cut across discipline and unique science boundaries. This was considered important because of the need to study a single parameter for more than one discipline. Although it is possible to look by *Interest Area*, by *sphere* and by *General Parameter,* this query would produce a large amount of non-relevant data since there are many more antecedents involved in defining what data or information is desired.

Under the super class object *Instruments* three classes were created that are used to partition this part of the domain into more manageable groups. This structure approximates what needs to be done in order to efficiently perform a query based on instrument type.

STEP 2 Formulation and Representation of the Instrument Super Class Object

The second step in the modeling process was the complete representation of the *instrument* super class. This is of critical importance since this is where all of the data sets and related sensor observations are located and subsequently managed. A singularly important consideration was that the model had to accommodate the automatic input of meta data and identified feature data into the database as new data objects were added to the mass store system. The impact of this requirement was profound, as it required a property inheritance strategy to function both from the top down as well as the bottom up.

NEXPERT supports a robust property inheritance capability by providing either top down, bottom up or both. Basically properties are included into each frame's property list by name. After the name is entered the system asks for the property to be defined by type as either a strings (variable length), integer, floating point, date, time, or boolean value. After the slot is defined, an inheritance

strategy is selected and a property value is put into the slot. If the parameter is to be inferred, then the slot will defined as empty and the value will be determined based on some set of rules. (It should be noted that at the beginning of the model development session the breadth first strategy was selected as the default.)

Property slots were needed for the data model for three purposes. First, they were required to allow the passing of relevant class type information to the low level objects that were associated each unique data records in the mass store system (information such as sensor resolution, sensor type, etc.). Second, they were required to store summary information at the class level based on the inference of all the related parameters of the daughter objects. Third, they were required for storing 'identified features', detected and cataloged as part of the Automatic Data Cataloging and Characterization process. [10]

The property slots without values, were inferred by rules for the following cases:

- To determine meta data for a data object from the header record of the data file stored in the mass store system. Meta data consisted of ID number, date, time, observation location, etc.

- To determine summary information properties that would be required for some parent object. Information inferred included: data range, time range, number of data records, etc.

- To store identified features found in the data file (sensor observation) from the results of the Automatic Data Cataloging and Characterization process. Identified features include such things as clouds, volcanos, thunderstorms, lakes, forests, etc.

Once the *instrument* super class domain representation was complete (shown in Figure 4), sets of rules were coded to support the summarization of the inheritable properties from all the daughter data objects into property slots at the class level. For example, rules were created that read the "date value" from each data object's "date" property. These values were then stored in a list, and sorted in ascending order. When this was complete the top value would be the earliest date and the bottom the latest date. When the two values were combined they provided the date range that was then placed in the class's date range slot. Using a similar strategy all of the class's other summary property slots were populated.

STEP 3 Formulation and Representation of the Application View

The application view consists of the discipline domain which is characterized by areas of scientific interest, and science object domain which contains all classes and objects that are studied and measured as part of a scientific endeavor. There is a close logical link between any discipline domain and its related objects of interest. However, they are separate in that a discipline domain uses observations from instruments to study phenomena that are the objects of interest. The discipline observations are actually the same as instrument sensor observations with some value added processing such as calibration correction and registration.

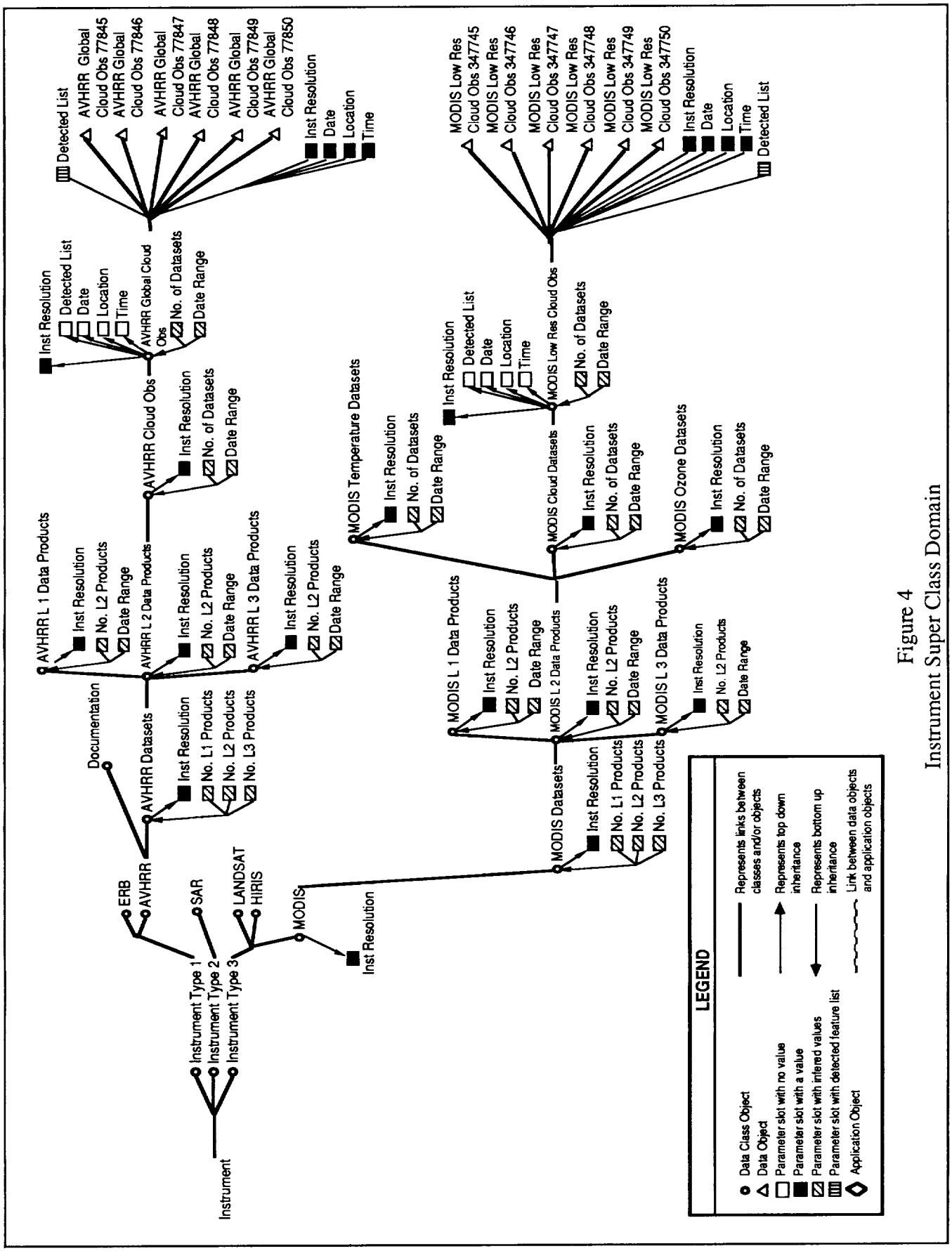

Figure 4
Instrument Super Class Domain

The modeling of the first domain resulted in the representation of the *Area of Interest* super class and the *Parameter* super class discussed in Step 1. The representation of both these super classes is incomplete, with only sufficient detail to demonstrate the semantic modeling concept and how one accesses data by either interest area or parameter.

The actual structure for each of the super classes was based on interviews with a atmospheric scientist supporting the NASA Climate Data System, and was organized to focus on selecting desired sensor observations that were ingested in the instrument super class domain. Figure 5 provide a diagram of this domain.

The representation in atmospheric science focused specifically on clouds and cloud related phenomena. Access to data using the two class objects was supported by links between objects that span between the operational view and the application views. These links are shown in Figure 6, Semantic Model Overall Architecture. The links provide not only the pointer to where a particular data object class can be found in the instrument super class domain, but it also provide a path for meta data to move so that objects that relate to phenomena of interest can be updated with parameter data from the instrument class objects. For example, summary data from the class objects *AVHRR cloud observations* and *MODIS cloud data sets* were linked to the object *Formations* . [1] Within these objects, meta information was further summarized using rules such as would be required to determine the date range for all observations regarding cloud formations along with information as to a list of sensors that observed identified formations. Using the information stored in the summary data property slots, it would be possible to present to a database user information that would assist in selecting desired observations prior to actually browsing through actual data sets. Information stored in a summary property slots might include what instruments are available, the time duration of a sensing activity and the number of observations made.

The modeling of the second domain is based on creating a logical structure of related data classes and objects that can be associated to a specific area of interest, discipline or parameter. The rationale for building this part of the model was motivated by having to support some sort of Automated Data Cataloging and Characterization operation. Basically such an operation would scan a sensor observation for any unique feature and then note this feature in a identified list that is related to the observation itself. Because automated data ingest is still in its formative stages, several assumptions had to be made as to how it will function. The assumptions are:

1. All detected features/objects are placed in a property slot called *detected list* that is associated with the sensor observation object in the instrument super class domain.

2. Detected features are coded so they can be sorted by class/group in some sort of logical structure.

3. All detected features/objects retain parameters that point back to the sensor observation where the feature/object was found.

Given the above conditions, features and objects move from the detected list parameter slot to the science object domain where sets of rules are used to sort the objects into the various grouping where they will be stored and then summarized at higher levels. This part of the semantic data

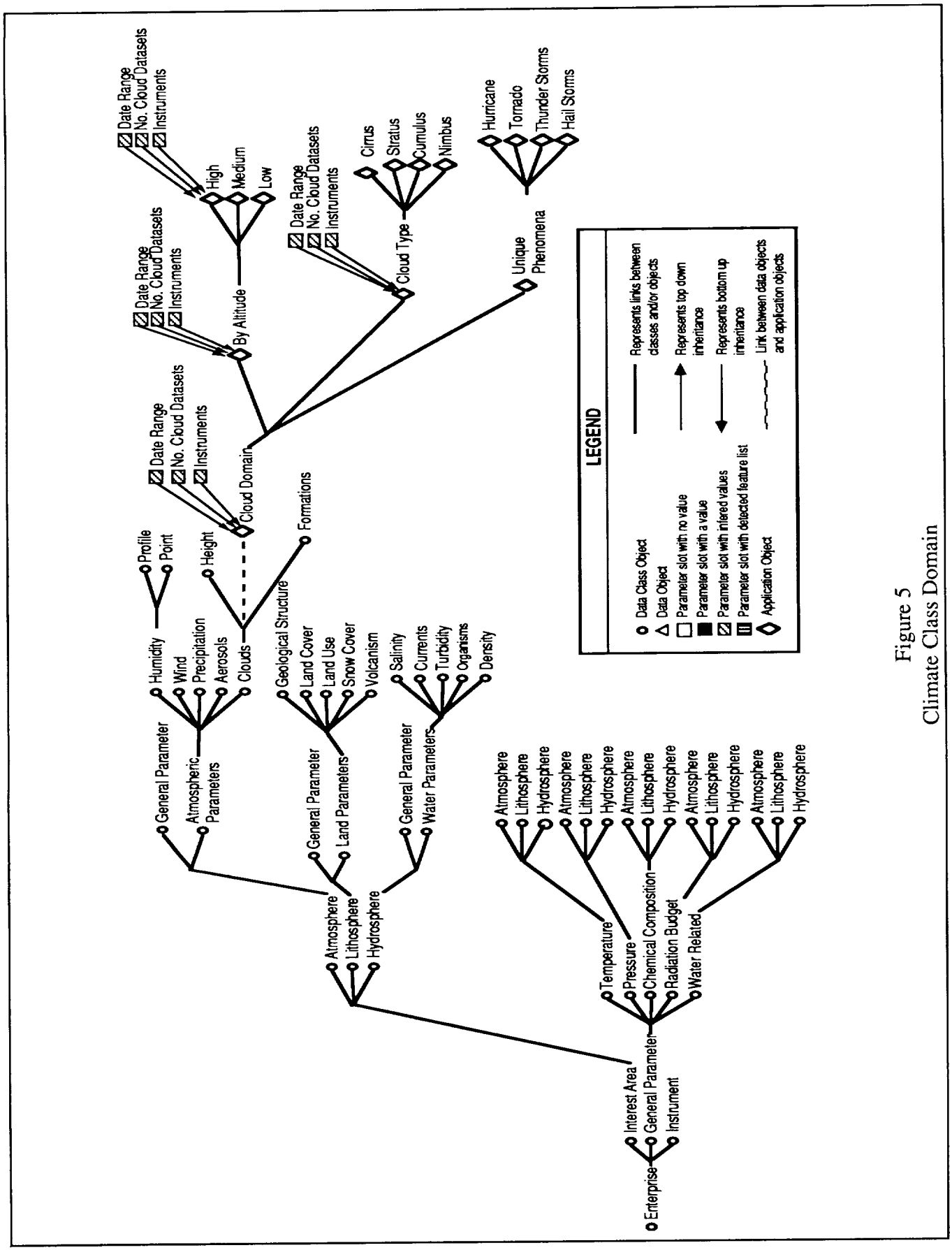

Figure 5
Climate Class Domain

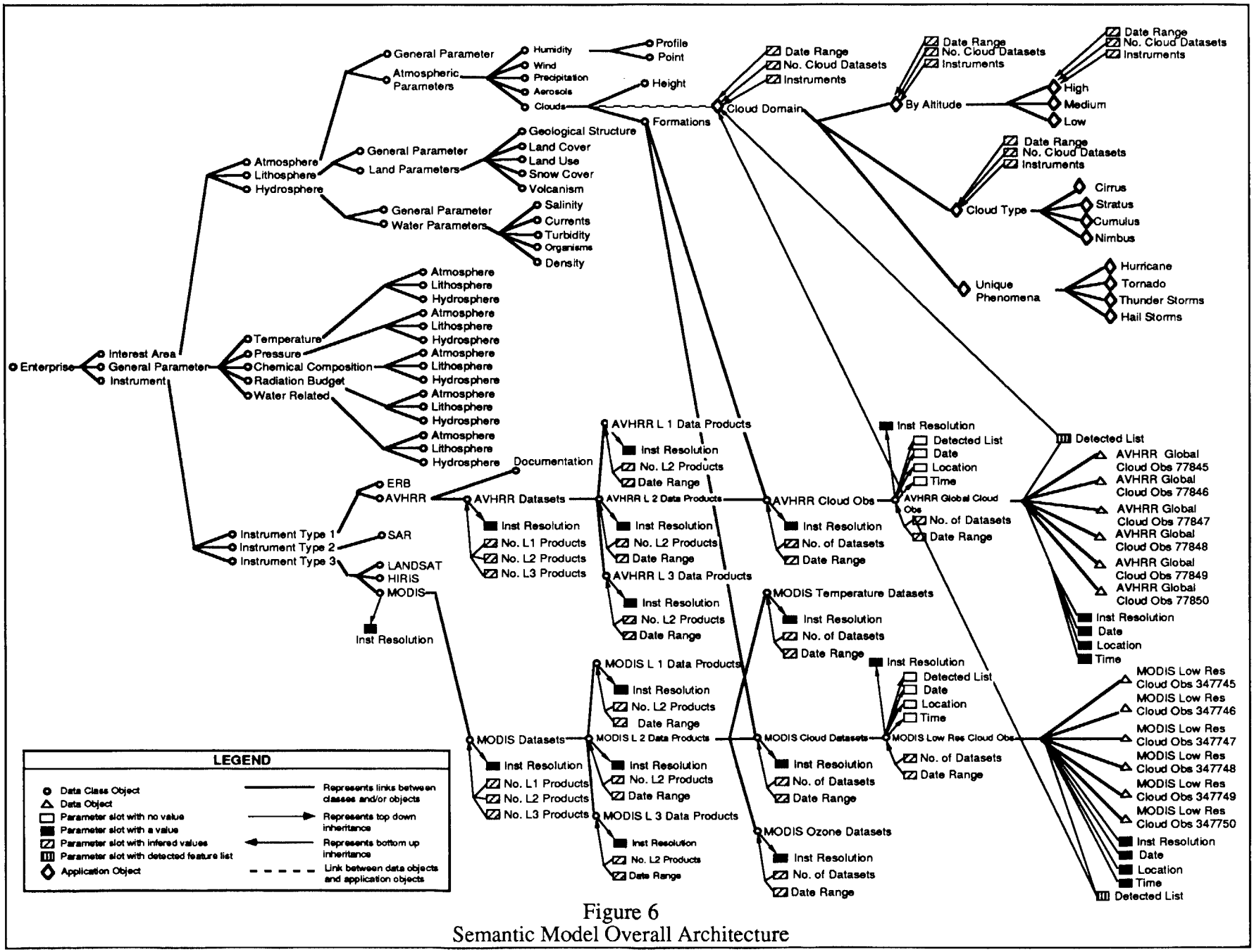

Figure 6
Semantic Model Overall Architecture

model was not prototyped in NEXPERT because of the uncertainties involved in how the object will be ingested and classified. Once these two issues have been resolved then the model can be updated to accommodate the feature/object ingest function.

Conclusions

Once the model prototype was completed, dummy data set objects, and associated detected lists were put into the model and evaluated. The evaluation found that semantic modeling significantly aided in understanding the data domain; something that needs to be done prior to building any database system. Of special note is the fact that semantic data modeling, when coupled with supporting meta knowledge provides a powerful tool for defining the logical architecture, and resultant information system design, for a large mass storage system. In addition, the semantic data modeling design approach provides a method for laying out a information system design and development effort and then a way of measuring the effort's success against some logical standards. Given this, we feel that any future data system design of any size and complexity should consider this approach since it probably is the only way one can be certain that the resultant design is logical sound and supports not only the needs of the database administrator but also the needs of the user.

The design and development of any useful semantic data model will require a great deal of effort that is directly dependent on how complex of the operational and application views of the data domain. It cannot be over emphasized how much effort (much of which probably will be of a high quality like a knowledge engineer) such a modeling activity may require especially for a large mass store system.

Finally, it appears that it is possible to implement a semantic data model with an associated knowledge base using an object oriented expert system development tool like NEXPERT. However, one should be cautioned that familiarity with the tool is quite important as the complexity of the model will dependent on how well the tool is able to represent the domain.

REFERENCES

[1] Earth Observing System, 1989 Reference Handbook, National Aeronautics and Space Administration, Goddard Space Flight Center

[2] W.J. Campbell, L. H. Roelofs, N.M. Short Jr., *The Development of an Intelligent User Interface for NASA's Scientific Databases*, Telematic and Informatics Vol. 3 No. 3 pp 177-190, 1986

[3] E.F. Codd, *A relational model for large shared data banks,* CACM, Vol. 13 No.6 377-387 1970.

[4] D. Tsichritzis and F. Lochovsky, *Hierarchical database management: a survey,* ACM Computer Surveys, Vol. 8 No. 1, 105-124, March 1976.

[5] R.W. Taylor and R.L. Frank, *CODASYL database management systems,* ACM Computing Surveys, Vol. 8 No. 1, 67-104, March 1976.

[6] M. Hammer, and D. Mcleod, *On the Architecture of Database Management Systems*, Infotech State-of-the-Art Report on Data Design, 1979

[7] W. Kent, *Limitations of Record-Based Information Models*, ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979 pp 107-131.

[8] J. Peckham and F. Marynaski, *Semantic Data Models*, ACM Computer Survey, Vol. 20, No. 3, September 1988, pp 153-221.

[9] E.F. Codd, *Further Normalization of the Database Relational Model,* Database Systems, ed. R Rustin. Englewood Cliffs, N.J., Prentice-Hall Inc., 1971.

[10] W. J. Campbell, L. H. Roelofs, M. Goldberg, *Automated Cataloging and Characterization of Space-Derived Data*, Telematics and Informatics, Vol. 5, No. 3, pp 279-288, 1988.

[11] McLeod, D., *A Semantic Database Model and Its Associated Structured User Interface*, Technical Report, MIT Laboratory for Computer Science, Cambridge, Mass., 1978

[12] W.D. Potter, R.P. Trueblood, C.M. Eastman, *Hyper-semantic data modeling*, Data & Knowledge Engineering, North Holland, Volume 4, Number 1, July 1989.

[13] D. Tsichritzis and F. Lochovsky, *Data Models*, Prentice Hall, Englewood Cliffs, NJ, 1982

[14] W. Kent, *Limitations of record based information models,* ACM Trans. on Database Systems, Vol. 4 No. 1, 107-131, March 1979.

[15] J.R. Abrial, *Data Semantics in Data Base Management,* J.W. Klimbie and K.L Koffeman Eds, 1-59, North Holland, Amsterdam, 1974.

[16] R. Hull and R King, *Semantic database modeling: survey, applications and research issues,* ACM Computing Surveys, Vol. 12 No. 1, March 1987.

[17] J.M. Smith and D.C.P. Smith, *Database abstraction: aggregation and generalization,* ACM Trans. on Database Systems, Vol. 2 No. 2, 105-133, June 1977.

[18] N.M. Short Jr, W.J. Campbell, L.H. Roelofs, and Scott L. Wattawa, *The Crustal Dynamics Intelligent User Interface Anthology*," NASA TM 100693, October, 1987.

[19] W.D. Potter, R.P. Trueblood and E.M. Eastman, *Hyper-semantic data modeling*, Data & Knowledge Engineering, Vol. 4, No. 1, July 1989, pp 69-90

[20] W.D. Potter and L. Kershberg, *A unified approach to modeling knowledge and data*, Proc. IFIP TC2, conf. Proceeding on Knowledge and Data, (DS-2), Algarve, Portugal, November 1986.

[21] Neuron Data Inc., NEXPERT *Object Fundamentals*, 444 High Street, Palo Alto, CA 94301