

## A Distributed, Multimedia Environmental Information System

Joseph L. Sharnowski, Gerald C. Gannod, and Betty H. C. Cheng\*

Department of Computer Science  
Michigan State University  
East Lansing, Michigan 48824

### Abstract

*Scientific research addressing global change continues to generate large quantities of information for analysis and understanding. However, the volume, distributed nature, and diversity of this information prohibits convenient access by many potential users. This paper describes an object-oriented, distributed system consisting of an integrated collection of software tools that allows a user to query and manipulate distributed, multimedia data sets through a graphical user interface (GUI). The tool, ENFORMS (Environmental Information System), is currently populated with environmental information for use in a regional watershed analysis project.*

### 1 Introduction

In the past, NASA and other agencies have focused on the acquisition of data rather than the integration or the dissemination of data. Many organizations addressing grand challenge problems, such as those defined by earth sciences, require the integration of both physical and human resource databases in an interactive manner. Such a capability allows reasonably informed policy analysts and related staff to query an "Environmental Science Workstation" so as to better understand how human uses impact our natural resource base.

This paper describes an object-oriented, distributed multimedia system consisting of an integrated collection of software tools that allows a user to browse and manipulate distributed, multimedia data sets through a GUI. Once the connection to an archive is established, the user is able to interactively explore it using the features provided by the GUI. Figure 1 shows a high-level view of the organization, where the rectangles represent participating sites.

The tool, ENFORMS, is currently populated with environmental information for use in a regional wa-

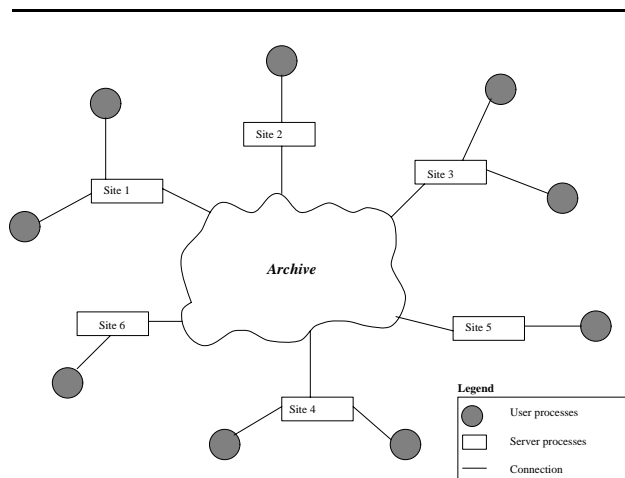


Figure 1: A high-level view of the architecture

tershed analysis project [1, 2]. ENFORMS provides graphics-based, interactive retrieval and manipulation of the distributed, multimedia information. The system archive contains a wide range of information items, including documents, satellite imagery, aerial photographs, color-coded charts, audio files, and animations. *Metadata* for the data (descriptions about the data, including its type, applicable tools, and relevant hierarchy classifications) are used to define the GUI, thus enabling users to construct queries graphically according to the context provided by the metadata. By changing the metadata, the entire interface can be changed to reflect a different perspective without changing the data. When accessing the archive, the distributed nature of the stored items should be transparent to the user with regards to the browser. That is, the query and manipulation activities are the same regardless of whether one or multiple servers is involved. In addition to search and retrieval of distributed data, the system supports data integration activities among the distributed data sets, including analysis with predictive and empirical models and spa-

\*Please address all correspondences to this author.

tial analysis with a geographic information system (GIS), which allows users to overlay geo-referenced maps.

The remainder of the paper is organized as follows. Section 2 describes the basic architecture of the system. The distributed framework is described in Section 3, including the different types of distributed services used in ENFORMS. Section 4 presents several sample user scenarios of the system. Related projects are described in Section 5. Finally, conclusions and future investigations are described in Section 6.

## 2 Architecture Overview

The architecture of the ENFORMS system is based on the understanding that an *archive* can be composed of a number of *items*. Each of the items found in an archive have a number of characteristics that determine the types of operations that can be performed on that item. For instance, an archive may contain a document that can be manipulated by a user via tools that allow editing, viewing, keyword searching, and so on. From a set of basic requirements, we constructed an object-oriented model of the system [3]. The graphical notations of the *Object Modeling Technique* (OMT) [4] are used to represent all models. OMT comprises three distinct models for object-oriented analysis. An *object model* describes the entities of the system and shows their conceptual internal structure and structural interrelationships. A *functional model* describes the functional behavior of the entities. A *dynamic model* specifies the operational relationships between entities. Due to space constraints, this paper presents only the object models.

Figure 2(a) shows the basic object model for the archive. Two classes of objects are shown in this figure: a **Multimedia Archive** and an **Item**, where the roman font denotes classes. Using the OMT notation, the line connecting these two classes asserts the existence of a relationship; the diamond denotes the *aggregation* relationship, where the class touching the diamond is the aggregate. The filled circle at the opposite end of the line denotes “many”, where *many* means zero or more; the absence of a filled circle at the endpoint of a line indicates “one”. Given this notation convention, the diagram can be interpreted as “an object of type **Multimedia Archive** is an aggregate of many objects of type **Item**.”

Rectangular boxes, representing classes, can be partitioned into three layers: the top layer provides the name of the class, the middle layer lists attributes of the class, and the bottom layer enumerates operations associated with the class. The aggregation notation

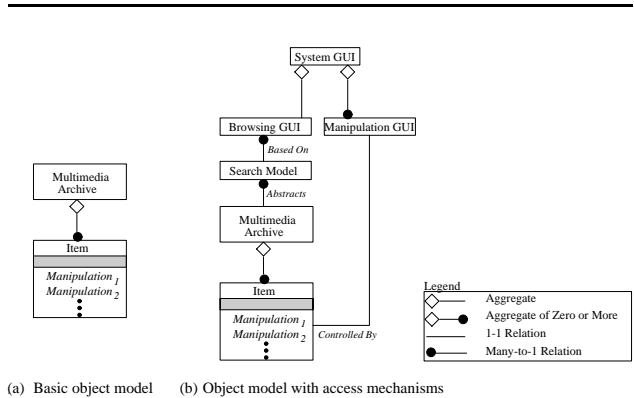


Figure 2: Object models of the archive

also identifies attributes of a class, thus introducing redundancy into the notation. In the case when either the attribute or operation part of the class notation is used, both layers are shown in order to clarify whether an enumeration lists attributes or operations. Note that shading represents unused layers.

In Figure 2(a), the notation for the class **Item** lists several *Manipulation* operations and indicates that **Item** objects are encapsulated units of information that can only be accessed through these services. This model for **Items** adds to the flexibility of the design by allowing nonatomic entities such as a group of related files, a set of maps, or even a software application to be treated as a single **Item**.

Figure 2(b) extends the basic object model by describing entities that provide access to the archive. Atop the **Multimedia Archive** is the **Search Model** that provides a browsing paradigm for the archive. The *Abstracts* relation between the **Search Model** and the **Multimedia Archive** is a *many-to-one* relation, and can be interpreted as “many different **Search Models** can provide abstractions of a single **Multimedia Archive**.”<sup>1</sup>

Figure 2(b) also introduces three interface entities: a **Browsing GUI**, a **Manipulation GUI**, and a **System GUI**. This figure shows that a given **Search Model** can have many **Browsing GUIs** based upon it. **Items** are *Controlled By* multiple **Manipulation GUIs**; the intended interpretation here is that each *Manipulation* operation of an **Item** is allowed to be coupled with its own GUI. Finally, the interface for the entire system, the **System GUI**, is modeled as an aggregation of a single **Browsing GUI** and zero

<sup>1</sup>Note that the interpretation of these general relations are dependent upon the direction that they are read. As a convention, the names of such relations are written to be read left-to-right and top-to-bottom.

or more **Manipulation GUIs**.

Figure 2(b) gives the general models for the archive, while the actual implementation uses specific choices relevant to the search, interface, and archive models, respectively. Figure 3 shows an extension to the general models, where **Multimedia Archive** is modified to include the notion of a **Distributed Multimedia Archive**. The remainder of this section gives details about the **Search Model** and the **Multimedia Archive**.

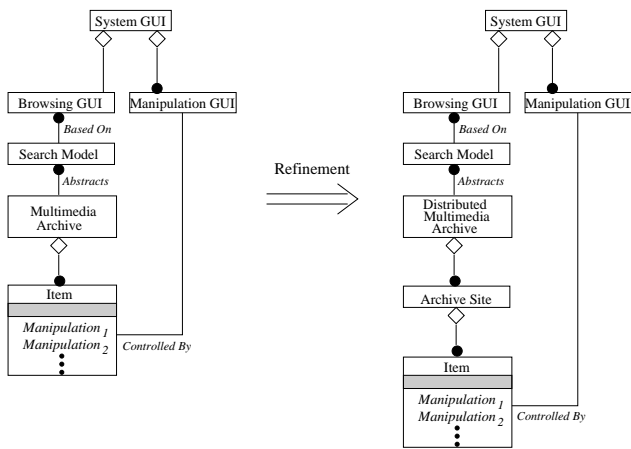


Figure 3: Object model for current implementation

## 2.1 Search Models

In ENFORMS, a **Search Model** defines how items are managed, classified, and retrieved. Examples of different search models include spatial, temporal, and hierarchical. The current implementation of the ENFORMS system uses a **Search Model** called the **IPAR Search Model**. IPAR is a simple, hierarchical classification scheme where the height of the hierarchy is limited to four tiers, namely the *Issue*, *Problem*, *Aspect*, and *Refinement* tiers. An instantiation of the classification scheme consists of a specific hierarchy, such as that shown in Figure 4. The **IPAR Search Model** is a generic architecture that is completely independent of the specific instantiations of the classification scheme. IPAR partitions the classification domain into specific issues of concern. Each issue has an associated set of problems, and any given problem has several aspects. Figure 4 contains a simple example hierarchy for the data of the system.

Item classifications relate the set of items stored in the archive to the interests of the ENFORMS user. When an item is added to the archive, it is described using the IPAR classification scheme in terms of those

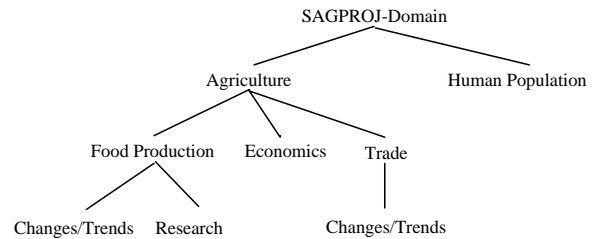


Figure 4: An example IPAR classification hierarchy

topics in the hierarchy to which the item relates. This approach to classification enables the user to locate items by constructing a query that describes the user's interests and requesting the system to find matches.

With respect to the analysis of the overall system, a generic IPAR model provides four basic categories of services to an IPAR-based GUI: elicitation of the current instantiation of the hierarchy, construction of classification-based queries, processing queries with respect to item classifications, and manipulation of query results. Conceptually, the IPAR model is composed of a classification hierarchy and a set of item classifications that are based on the hierarchy, as shown in Figure 5. The **Local Control Broker** provides the mechanism by which items are accessed in ENFORMS. That is, it is the interface into the **Multimedia Archive**.

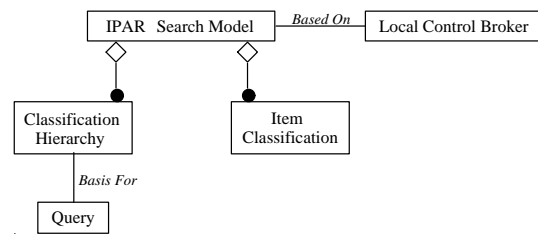


Figure 5: A simple object model for IPAR

## 2.2 Multimedia Archive Analysis

In general, a multimedia archive manages access to items. This relationship is captured by modeling the **Multimedia Archive** class as a collection of **Items**, each of which has its own set of associated access methods (as shown in Figure 2(a)). However, archive items are actually objects that exist external to the archive software, that is, in the domain of the operating system. Given this constraint, it is intuitive to model items abstractly using indirection. Figure 6 shows an object model for **Multimedia Archive**,

where the **Items** are indirectly managed by (*Registered By*) **Item Descriptors**.

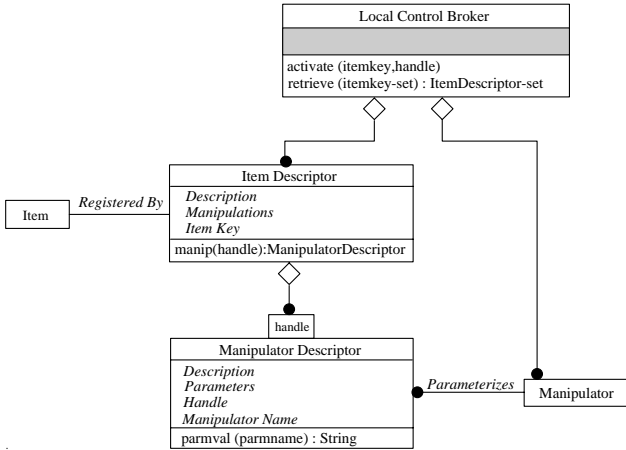


Figure 6: Analytic model of **Multimedia Archive**

For each **Item** in the **Multimedia Archive**, there exists an **Item Descriptor** that contains all relevant information for the **Item**, which may only be examined by allowable manipulators. Manipulations for an aerial photograph item, for example, might be to *describe the photograph* and to *display the photo*, while those of a bibliography might be *search by keyword*, *sort by author*, and so on. In order to perform such manipulations, the appropriate software tools, or **Manipulators**, must be available for use.

The **Local Control Broker** realizes the conceptual aggregation of a **Multimedia Archive** and its collection of items by managing **Items** and **Manipulators**. Access is supported in two ways: through the retrieval of **Item Descriptors** and the activation of **Manipulators**. In our current implementation of ENFORMS, six types of manipulators are supported: a GIS analysis tool, a text file display tool, an image viewing utility, an audio player, an MPEG animation player, and a generic application launching tool for applications that have their own GUI (e.g., data analysis models, pre-defined animations, etc).

### 2.3 Distributed Multimedia Archive

As stated earlier, one of the primary goals of ENFORMS is to provide users with convenient access to large amounts of multimedia information that may be distributed across many sites. As such, we extended the concept of the **Multimedia Archive** from a single machine, standalone entity to a **Distributed Multimedia Archive** as shown in Figure 3. We re-

efined this concept further by extracting the appropriate components of the **Search Model** and **Multimedia Archive** classes (shown in Figure 3). This refinement allows us to derive a new model that supports the access and manipulation of distributed data classes in a *client-server* framework. In this type of framework, *servers* accept requests over a network, perform the relevant services, and return the results. The *client* is a program that requests the services and receives the results.

Figure 7 depicts the results of the refinement of the ENFORMS system into a client-server model. One of the main features of the **Distributed Multimedia Archive** is the introduction of a **Distributed Control Broker** that facilitates the virtual communication between client and server **Search Model** objects. This communication is shown in Figure 7 by the *Talks To* relationship between the client and server **Distributed Control Broker** classes, and the *Talks To* relationship between the client and server **Search Model** classes, which represent real and virtual communication, respectively.

Conceptually, this separation of client and server allows for the implementation of many different kinds of servers, each providing support for a different search model. These search models could include the currently existing **IPAR** model as well as search models that allow for spatial- and temporal-based searches.

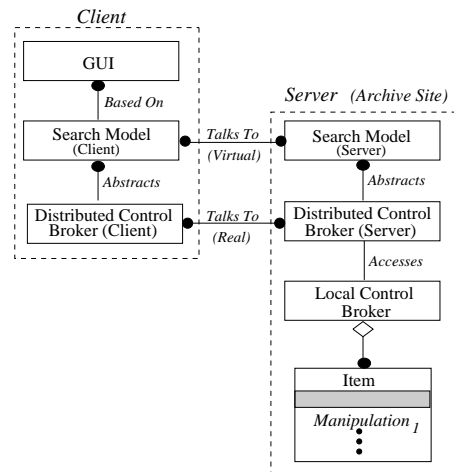


Figure 7: The client-server architecture

## 3 Distributed Operation

ENFORMS is implemented as a distributed system, where the functionality of the system is realized through a collection of communicating programs. This

section discusses the features of the distributed operation of ENFORMS. We first discuss the model that characterizes the distributed architecture of ENFORMS, and we then describe the procedure for performing distributed queries.

### 3.1 The Client-Server Model

In the case of the ENFORMS client, the purpose of the requested services is to support the functionality realized at the GUI level. The requests that may be issued by a client are categorized into four types. First, *Query* requests may be issued for selecting items based on item classifications. Second, *Query-Count* requests may be issued for determining the number of items that match a particular query. Query-count requests are currently used to initialize the ENFORMS GUI for displaying the number of refinements that correspond to each issue-problem-aspect combination in the IPAR search model domain. Third, *Activate* requests may be issued for executing manipulators with selected sets of parameters. Finally, *Server table* requests may be issued for determining the set of servers to which the other three types of requests may be sent. For each of these four cases, the communication between the client and a server is essentially hidden from the user. That is, the underlying details of establishing connections and transmitting messages are handled without user intervention.

Two types of servers are supported by ENFORMS. The first is an *archive server*, which manages the searching and the manipulation of items for an archive site. An archive server directly supports the first three types of client requests described above. In particular, these services perform queries, perform query-counts, and activate manipulators.

The other type of server supported by ENFORMS is referred to as the *name server*. The name server is responsible for maintaining a table of the active archive servers, referred to as the *server table*. The address and port number of the name server is established prior to the activation of ENFORMS, such that when an archive server is activated, the location of the name server is already known. Using this information, the archive server sends a *registration* message to the name server, providing its address and a port number for communicating with clients. The name server then stores that information as an entry in the server table. Likewise, when an archive server is deleted, the corresponding entry in the server table is removed.

The name server is responsible for handling server table requests from clients. Prior to issuing query, query-count, or activate requests to the archive servers, the client must first send a server table re-

quest to the name server to learn the addresses and port numbers of the active archive servers. If the entries in the server table change, then the clients receive an updated version of the table.

Multiple archive servers may be active at one time, where each server provides services for a particular archive site. However, a single instance of a name server answers the server table requests for the entire collection of clients in ENFORMS. An additional characteristic of the distributed operation of ENFORMS is that a client may send requests to any active archive server, and, likewise, an archive server can provide services to any of the active clients. The relationship between the name server, the archive servers, and the clients is illustrated in the architectural overview shown in Figure 8.

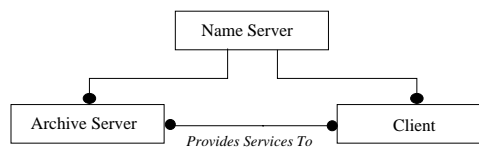


Figure 8: Architectural overview of the client-server model for ENFORMS

The execution of ENFORMS is initiated by activating the name server. The archive servers are then typically activated, followed by any number of clients, although the system does not preclude the possibility of activating the clients before the archive servers. During the normal operation of the system, archive servers and clients may be started or stopped at any time, while the name server remains continuously active.

### 3.2 Distributed Query Processing

One of the key services provided by an archive server is the handling of queries. These queries are formed based on the specifics of the search model being used, for instance, IPAR. From the user's perspective, the query is issued at the GUI level, where ENFORMS promptly responds with the matches to the query. However, the underlying details for processing the query are more involved.

A client processes a query by first issuing a query request to each of the archive servers. Upon receiving the query request, an archive server computes a list of matches for the query based on the information stored at its archive site, and then returns the list of matches to the client. When the client receives the list of matches, the address and port number of the source (i.e., the archive server) is recorded along with each

match. The separate lists from each archive server are collected by the client and then presented to the user at the GUI level. The user may select items from the list of matches, where each item may be examined in greater detail by using the set of manipulators for that item. Since the location of the relevant archive server is recorded with each match, an activate request for a manipulator may be sent to the appropriate archive server, where the archive server then handles the execution of that manipulator.

The ENFORMS system offers an option for sending queries to only a subset of the active archive servers. This option is useful, for example, when the user is only interested in the information located at a particular archive site or a group of sites. The search for query matches will thus be limited to those sites, even if matches are obtainable at other sites.

## 4 User Scenarios

This section illustrates the use of the system through several sample user scenarios. Figure 9 gives the opening screens of ENFORMS. It displays a list of environmental issues available in the system for browsing and a list of the currently active servers.

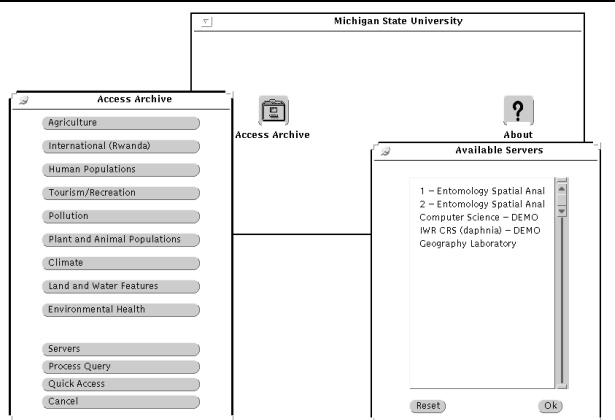


Figure 9: Opening screens of ENFORMS

The user may select a specific issue, which displays the corresponding *issue grid*. Suppose that the user is interested in *pollution* and specifically in locating information related to the problem of *groundwater contamination*. Figure 10(a) shows the issue grid for *Pollution*, where the problems *Surface Water Contamination* and *Groundwater Contamination* are displayed. In order to view all problems in the issue grid, the user must use the horizontal scroll bar at the bottom of the window. Below each problem is a vertical scrolling

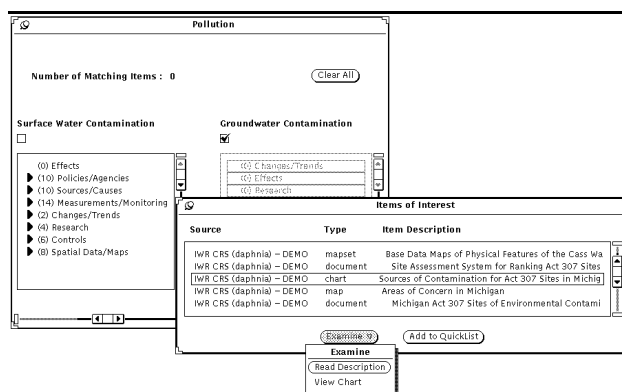


Figure 10(a): Query of *groundwater contamination* problem

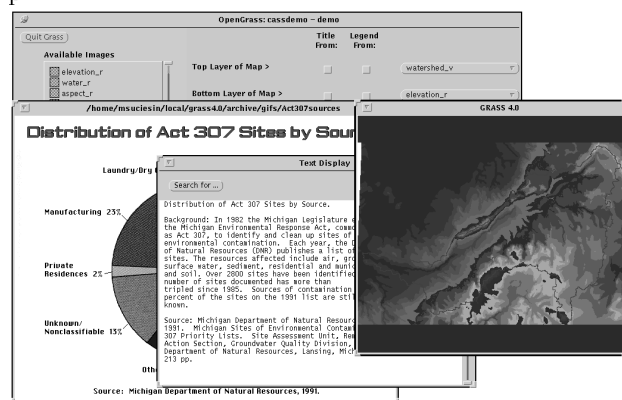


Figure 10(b): Sample types of available objects as viewed by different manipulators

list of subtopics related to that problem. Some of the subtopics, such as *Research* in the *Surface Water Contamination* column of Figure 10(a), have further levels of refinement (indicated by the arrow next to the name). Selection of subtopics with refinements invokes the display of an additional menu. The values in parentheses indicate the number of objects in the archive that match a specific subtopic. This feature is included to give the user a high-level view of the density of information in the archive and also minimizes the number of queries that have no matches. Disjunctive queries are formed by displaying multiple issue grids, where more than one instance of a given issue grid is allowed. In Figure 10(a), the entire column of *Groundwater Contamination* problems has been selected (via the checked box), indicating that the query is to find all relevant items for this problem. The results of the query are listed in the *Items of Interest* window, where among other types of data, a mapset, a document, and a chart are available for

examination. Notice that there are two ways to “examine” the chart object, *Read Description* and *View Chart*. Figure 10(b) displays examples of the different types of available items for the previous query examined via their respective manipulators, including a pie chart describing contamination sources and its textual description. It also shows a digital elevation map overlaid by a watershed boundary map (from the mapset item) displayed via a GIS manipulator, OpenGrass, a simplified graphical interface to GRASS (Geographical Resources Analysis Support System) [5].

A sample scenario for limiting the set of servers that receive a query is shown in Figure 11, where the user is investigating the relationships between crop productivity and climate in order to determine when crops should be planted in order to gain the best yield. In this case, the query was only sent to the last two of the three servers that are listed.

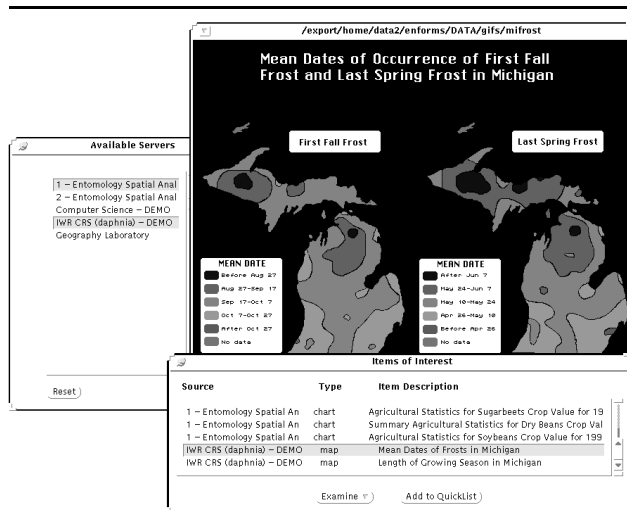


Figure 11: Query to a subset of servers

Figure 12 shows a query constructed by selecting the *Changes/Trends* aspect in the *Nonindigenous Species* problem column on the *Plant and Animal Populations* issue grid. Figure 12 also contains some of the query results that are in the form of two color-coded maps that depict the gypsy moth population in Michigan in 1985 and 1990, respectively. A user might be interested in this information because the gypsy moth is a serious defoliator of oak and aspen trees in the Saginaw Bay watershed region. Using visual data integration, an analyst can clearly determine that there is a need for some type of action to address the gypsy moth problem. Figure 13 illustrates how audio data integration can be used to facilitate the examination of the maps. In this case, an audio file was played that

described the contents of the color-coded map depicting the 1985 gypsy moth population.