

# The Object-Oriented Development of a Distributed Multimedia Environmental Information System\*

Betty H.C. Cheng, Robert H. Bourdeau, and Gerald C. Gannod

Department of Computer Science  
Michigan State University  
A714 Wells Hall  
East Lansing, Michigan 48824  
email: {chengb,bourd,gannod}@cps.msu.edu

## 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 the object-oriented analysis and design of a prototype system consisting of an integrated collection of software tools that allows a user to manipulate disparate 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. In addition, the benefits, with respect to extensibility, maintainability, and integration, from using an object-oriented approach in combination with a multi-layered organization are discussed.*

## 1 Introduction

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, including policy analysts, federal agency staff, and local township planners. This paper describes the object-oriented analysis and design of a prototype system consisting of an integrated collection of software tools that allows a user to manipulate disparate 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, 2]. Using ENFORMS allows decision-makers to

better understand and, perhaps, learn about new environmental issues related to watersheds, which potentially may have a global impact.

ENFORMS provides graphics-based, interactive retrieval and manipulation of multimedia information. The system archive contains a wide range of information items, including documents, satellite imagery, aerial photographs, and color-coded charts. The user accesses the archive by graphically constructing simple queries that reflect specific interests. A user's interest is expressed in the context of terms defined by the interface. Due to the object-oriented development of the system, the functionality of ENFORMS can be tailored to specific classes of users with different perspectives and objectives by externally reconfiguring the GUI. In addition to retrieval, the system supports data integration activities including analysis with a geographical information system (GIS), which allows users to overlay geo-referenced maps.

The remainder of the paper is organized as follows. Section 2 introduces the problem domain that the distributed multimedia archive addresses. Section 3 gives a high-level overview of the requirements for the system and describes the notation used in the paper. The object-oriented analysis of the system is discussed in Section 4. Section 5 gives the basic architecture of ENFORMS, including the object-oriented design based on a simplified version of the analysis results. Section 6 discusses the benefits gained from using an object-oriented approach and presents a sample user scenario and user evaluations of the system. Finally, conclusions and future investigations are described in Section 7.

## 2 Problem Overview

During this decade, NASA will launch many new platforms into earth orbit, including the satellites that will make up the Earth Observing System (EOS). The remotely sensed data obtained from EOS can be used to promote global and national security, extend international cooperation, and improve our ability to understand and manage global environmental,

---

\*This work is in supported in part by National Science Foundation grant CCR-9209873, NASA, USDA, EPA, US Geological Survey, Consortium for International Science and Information Network, and Michigan State University.

economic, and social problems. 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 project has initially focused on *access* to global change information, specifically relevant to earth science and global change issues, and will later address issues of use and understanding of this information. ENFORMS contains a wide variety of items, such as image files, research papers, executable environmental models, and research data sets. Using the graphical user interface, the user is able to examine these items interactively through the archiving software and display images and execute environmental models without requiring an extensive computer background.

### 3 System Requirements

The objective of this project is to provide users with convenient access to large amounts of multimedia information that may be distributed across many sites. The user accesses the archive, via a graphical user interface (GUI). Once the connection is established, the user is able to interactively explore the entire distributed archive by utilizing the features provided by the GUI. Figure 1 shows a high-level view of the organization, where the rectangles represent potential participating sites, and the filled circles represent users.

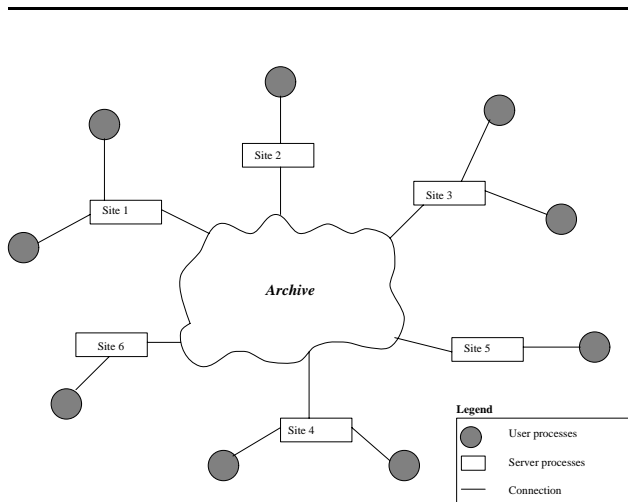


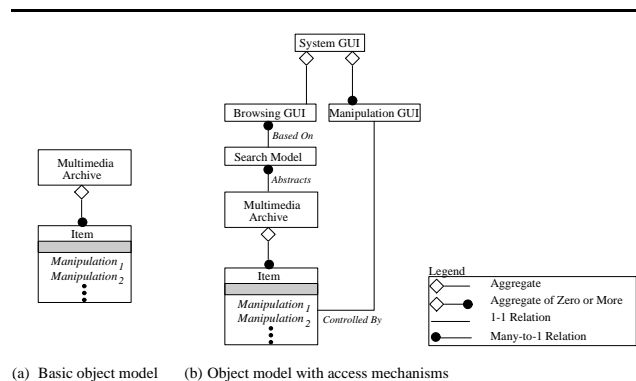
Figure 1: A high-level view of the architecture.

The main requirements of the system are threefold. First, the archive itself is potentially large, and thus some sort of selective browsing mechanism is necessary to help the user navigate through the archive. Although the *user needs team* (consisting

of a representative set of potential users) suggested a specific browsing and querying mechanism, we decided that in order to accommodate new users and new features, it was necessary that the system design be able to support a variety of browsing mechanisms and not be tightly coupled with either the user interface or the user interface technology. Second, information items may be stored in a variety of forms: simple databases and data files, images, documents, GIS maps, animations, parameterized models, and so on; accordingly, the system must be able to determine which software tools are needed to examine each item. Finally, when accessing the archive, the distributed nature of the stored items should be transparent to the user with regards to the browser; this requirement also suggests that the system should be able to operate both as a stand-alone system and as a component of a distributed system. From these basic requirements, we constructed a very high-level object-oriented model of the system.

The graphical notations of the *Object Modeling Technique* (OMT) [3] 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 bold 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**.”



(a) Basic object model (b) Object model with access mechanisms

Figure 2: Object models of the archive.

Square 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 also identifies attributes of a class, thus introducing redundancy into the notation. When either the attribute or operation part of the class notation is used, both layers are shown in order to eliminate confusion (shading represents unused layers).

Also 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` greatly 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 or more Manipulation GUIs.

Figure 2(b) gives the general models for the archive, while the actual implementation made specific choices relevant to the search and the interface models, respectively. The object model for the demonstration prototype is given in Figure 3. The demonstration prototype is simpler than the original archive model in two ways: only a single Search Model is necessary, and only a single Browsing GUI for the Search Model is required. These simplifications correspond to the elimination of the filled circles from the *Based On* and *Abstracts* relations in Figure 2(b), thus reducing a *many-to-one* relationship to a *one-to-one* relationship. The subsequent analysis, design, and implementation of the prototype is greatly simplified.

The demonstration prototype uses 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 (hence the acronym). An instantiation of the classification scheme consists of

<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.

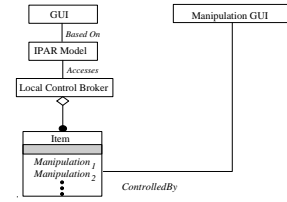


Figure 3: Object model for demonstration prototype.

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 domain into specific issues of concern. Each issue has an associated set of problems, and any given problem has several aspects.

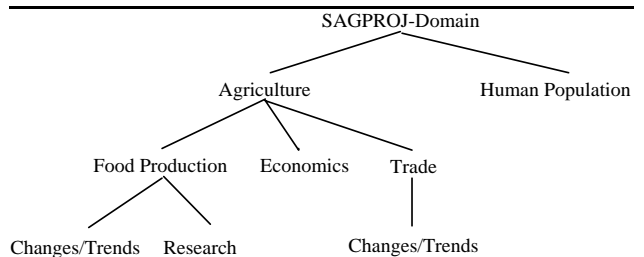


Figure 4: An example IPAR classification hierarchy.

Item classifications relate the set of items stored in the archive to the interests of the system user. When an item is added to the archive, it is described to IPAR in terms of those topics in the hierarchy to which it relates. The user is then able to locate items by constructing a query that describes the user’s interests and requesting the system to find matches.

## 4 Analysis and Decomposition

This section presents an analysis of the system requirements with respect to the object model shown in Figure 3, and a more detailed object model of the proposed system is derived.

### 4.1 Graphical User Interface

Rapid prototyping provides a means of communication between the development team and the users when defining system requirements [4]. Much of the analysis of the GUI layer was performed in the context of rapid prototyping, and as such, was helpful in communicating the needs and requirements of users. Example screens displayed by the system are shown in Section 6. In terms of the analysis of the other system components, the GUI does not fall within the scope of this paper, instead, refer to [5].

### 4.2 Analysis of the IPAR Search Model

With respect to the analysis of the overall system, a generic IPAR model must provide four basic categories of services to an IPAR-based GUI: elicitation of the current instantiation of the hierarchy, construction of classification-based queries from the hierarchy,

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. This object model can be further decomposed, beginning with the classification scheme.

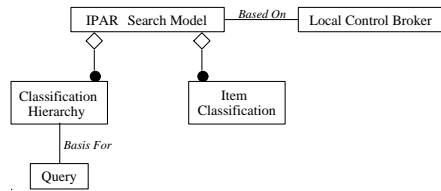


Figure 5: A simple object model for IPAR.

A classifier is a 4-tuple that must be of the form:  $(i, p, a, r)$ ,  $(i, p, a, *)$ ,  $(i, p, *, *)$ , or  $(i, *, *, *)$ , where  $i, p, a$ , and  $r$  are specific names for issues, problems, aspects, and refinements, respectively. An IPAR hierarchy is determined by a finite set of classifiers; for example, the classifiers that characterize the hierarchy shown in Figure 4 are:

- (Agriculture, Food Production, Changes/Trends)*
- (Agriculture, Food Production, Research)*
- (Agriculture, Economics)*
- (Agriculture, Trade, Changes/Trends)*
- (Human Population)*

Thus, the Classification Hierarchy and Item Classification components shown in Figure 5 can be refined to the object model shown in Figure 6. Dashed and dotted regions enclose the components that correspond to the Classification Hierarchy and the Item Classification components, respectively, shown in Figure 5. Specifically, the IPAR Classification Hierarchy is modeled as a collection of classifiers. The IPAR Search model contains a set of Item Classifications, each of which is a collection of Classifiers and an ItemKey. An ItemKey is used in this analysis to emphasize that a Search Model does not “own” Items (Items are components of the Multimedia Archive entity); instead, only references to Items are allowed. Therefore, the Classification Hierarchy defines the structure of the classification scheme, and the set of Item Classifications defines the logical mapping from the classification scheme to Items.

The last component of the IPAR Search Model of interest is the Query class. A query is an expression that is used to select items from the archive contents based on item classifications. Abstractly, a query is modeled as a propositional formula, where classifiers are treated as the simplest type of query (atoms). In Figure 6, the Query class is modeled using a recursive structure; the triangle symbol denotes a subclass-superclass relationship where the triangle “points” to the superclass. Three types of queries can be formed: conjunctions, disjunctions, and individual classifiers, where conjunctions and disjunctions are (recursively) collections of queries. The IPAR Search Model class, as depicted in Figure 6, defines only

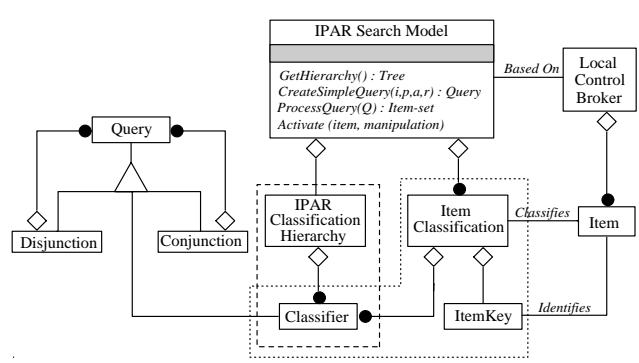


Figure 6: Details of the IPAR Search Model.

one query construction service, *CreateSimpleQuery*. Instead of modeling query composition operations, such as conjunction and disjunction, as services of the IPAR Search Model, two Query class operations are introduced:  $\wedge$  (conjunction) and  $\vee$  (disjunction). Conceptually, an expression of the form  $Q_1 \wedge Q_2 \vee Q_3$  represents a Query, where  $Q_1, Q_2, Q_3$  are objects of type Query.

### 4.3 Multimedia Archive Analysis

In this analysis, a multimedia archive manages access to items. This notion 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 7 shows an object model for Multimedia Archive, where the Items are indirectly managed by (*Registered By*) Item Descriptors.

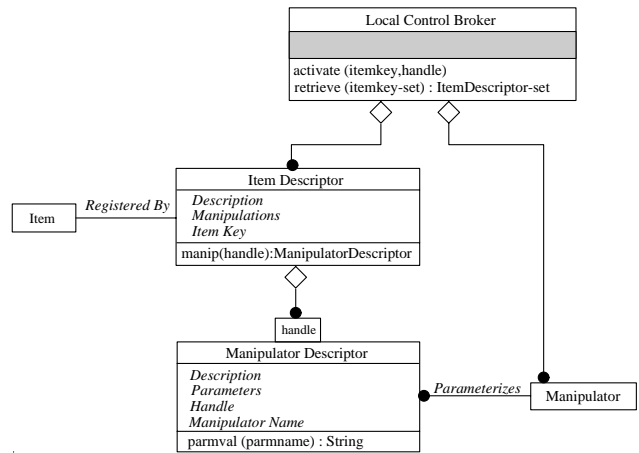


Figure 7: Analytic model of Multimedia Archive.

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

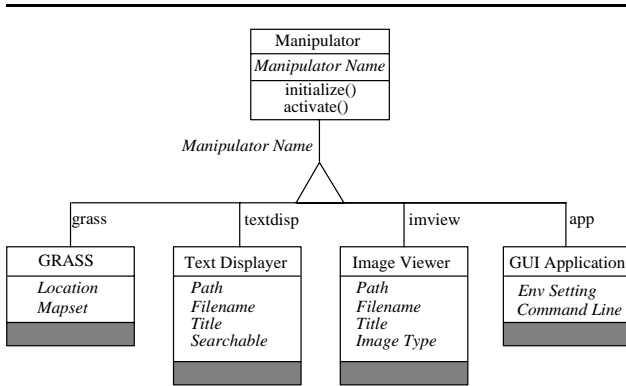


Figure 8: Analytic object model of Manipulators.

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.

For the demonstration prototype, only four types of manipulators were identified: a GIS analysis tool, a text file display tool, an image viewing utility, and a generic application launching tool for applications that have their own GUI (environmental models, pre-defined animations, etc).

By parameterizing a manipulator, it may be reused for different items. For example, an image display manipulator need only know the name, location, and format of the image file to display it. These attributes are known as the *parameters* of the Manipulator. Figure 8 gives an object model for manipulators.

All Manipulators must have a unique name and two specific services, initialization and activation. Initialization is only performed before the first activation, where activation is synonymous with invocation. As shown in Figure 8, binding the names of the Manipulator parameters to specific values activates the Manipulator with the corresponding behavior.

As shown in Figure 7, Item Descriptors are modeled as an aggregate of Manipulator Descriptors, each of which is uniquely identified by its *handle*. For a given Item, each Manipulation is specified by a Manipulator Descriptor. The attributes of a Manipulator Descriptor include a fixed manipulator name and a set of Parameters. A Parameter consists of two attributes: a name and a value. In order to execute a Manipulation for a given Item, the Multimedia Archive extracts the appropriate Manipulator Descriptor from the Item and supplies it to the named Manipulator.

## 5 Design-Level Object Modeling

The analysis phase focuses attention on capturing concepts rather than producing an easily implementable system. The purpose of the design phase is to produce a detailed description of a software architecture that can be used to guide the implementation. Therefore, the concepts identified in

the analytic model must be captured in the design of the corresponding object model, but the actual organization of the design need not be the same as that of the analytic model. This section describes the development of a software architecture based on the analytic object model.

The distributed nature of the archive is the first issue addressed. As described in Section 3, the execution of the communication protocols should not involve the user. This requirement is fully realized in the analytic model by not referencing the physical location of information. In the design phase, however, the analytic model must be decomposed to accommodate a network-based implementation. Figure 9(a) gives a high-level, client-server decomposition of the analytic model shown in Figure 2(b). Query processing is handled by the Search Model entities; accordingly, the distributed query processing must be handled by the client and server Search Models. The Multimedia Archive entity manages access to the items. Therefore, access to items in a distributed environment must be managed by a Multimedia Archive server and a corresponding client. Figure 9 indicates that any given server can interact with many clients, and any given client can access many servers. For the demonstration system, we specifically used the IPAR Search Model for the searching mechanism. A detailed analysis of the object model shown in Figure 9 is documented in [6].

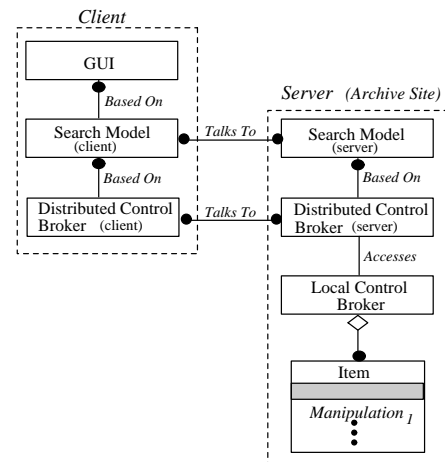


Figure 9: The client-server architecture

For brevity purposes, we focus the remaining design discussions on the browsing, searching, and manipulation of items on one of the archive sites. It is assumed that each of the archive sites has the same design, and one specific site has additional responsibilities involving a name server to maintain the participating archive sites. Details regarding the networking issues of the distributed archive may be found in [6].

### 5.1 IPAR Search Model Architecture

Figure 10 gives a high-level object model for the design of the IPAR Search Model. This design

adheres to the analytic IPAR model given in Figure 6, while incorporating as much reuse of existing classes as possible. Two major factors account for the difference between the analytic and design models: the design of collections and the delegation of services. Collections, such as those used for the IPAR Classification Hierarchy shown in Figure 6, must have a structure that allows storage and retrieval operations, and, for implementation purposes, these details must be provided. Services, such as *GetHierarchy* must be clearly implementable; since the classification hierarchy is represented by the IPAR Classification Hierarchy entity, it is appropriate to delegate the implementation of the *GetHierarchy* service to this class.

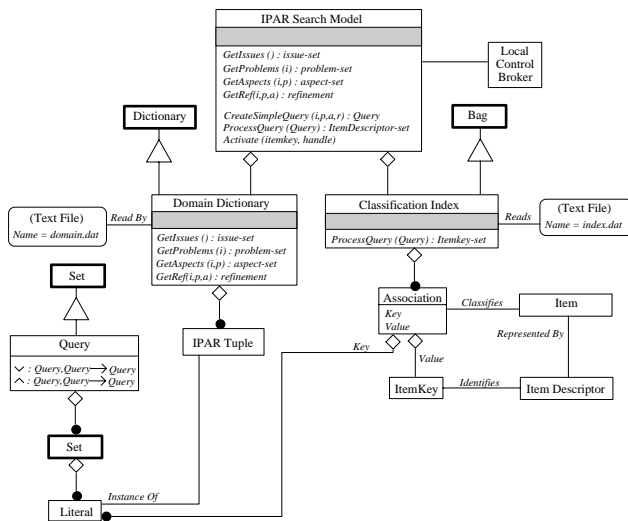


Figure 10: Architecture of the IPAR Search Model.

The design of the IPAR Search Model architecture makes use of three different types of container classes from an existing container class library: Set, Bag, and Dictionary. A Bag is an arbitrary collection of objects with possible duplicates, a Set is a Bag with no duplicates, and a Dictionary is an indexed collection of objects. By reusing these classes, we simplify both the design and the implementation of many collections. For example, Set is used in the analysis of a Query object, which is modeled as a propositional formula without negation, where classifiers are treated as atoms. Any propositional formula can be written in disjunctive normal form (DNF), or sum of products, and a DNF expression can be represented as a set of conjuncts, where each conjunct is represented as a set of atoms. Thus the recursive structure that is used to represent a Query in the analytic model can be designed as a Set of Sets of Literals; Literals approximately correspond to the notion of a Classifier in the analytic model.

As another example of class reuse, the Domain Dictionary shown in Figure 10 corresponds to the

analytic notion of the Classification Hierarchy displayed in Figure 6. The Domain Dictionary stores the classifiers (IPAR Tuples) and extends the specification of the class Dictionary to facilitate file storage and retrieval of the hierarchy. In examining the *GetHierarchy* service from Figure 6, it is clear that the delivery of the entire hierarchy is unnecessary and costly. Accordingly, the *GetHierarchy* service is decomposed into four primitive operations that allow the hierarchy to be extracted through different types of traversals. These four primitive services, *GetIssue*, *GetProblem*, *GetAspect*, and *GetRef* can be used to perform a breadth-first search, depth-first search, and many other types of traversals. The implementation of these primitives is naturally delegated to the Domain Dictionary entity that stores the hierarchy.

## 5.2 Multimedia Archive Architecture

The design of the Multimedia Archive (shown in Figure 11) is more intuitive than that of the IPAR Search Model. The analytic model of the Multimedia Archive depicts this class as a collection of Item Descriptors. Again, it is preferable to derive the storage structure for this collection from our container class library.

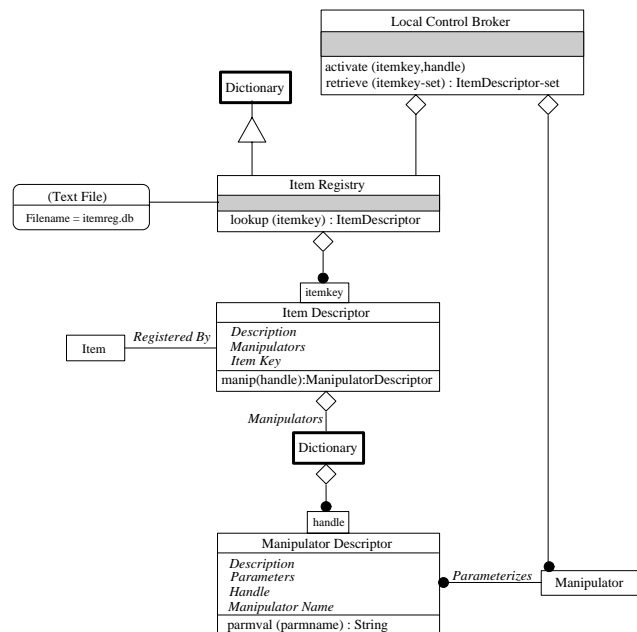


Figure 11: Architecture of the Multimedia Archive.

The analytic model for Multimedia Archive indicates that an Item can be accessed by specifying its key, thus suggesting that the collection should be indexed and, accordingly, be derived from the class Dictionary. The Item Registry class, shown in Figure 11, is the resulting design of the collection of Item Descriptors. This collection is indexed by the ItemKey as shown by the qualifier *itemkey* on the aggregation connection. The Item Registry is also related to a specific text file;

this file is the Item Descriptor database that is used to initialize the registry.

As shown in Figure 7, the class Item Descriptor is modeled as an aggregate of Manipulator Descriptors, indexed by a *handle*. This collection is also designed as a Dictionary as shown in Figure 11.

## 6 System Evaluation

This section illustrates the use of the system through sample user scenarios. Feedback from a group of users of the system is discussed, and a discussion of the benefits of using an object-oriented approach from the development team's perspective is presented. Finally, a brief comparison is made between ENFORMS and other similar systems.

### 6.1 User Scenario

Figure 12 contains a list of environmental issues currently available in the system for browsing; this list is determined at runtime when the system is initially invoked using the IPAR classification scheme.

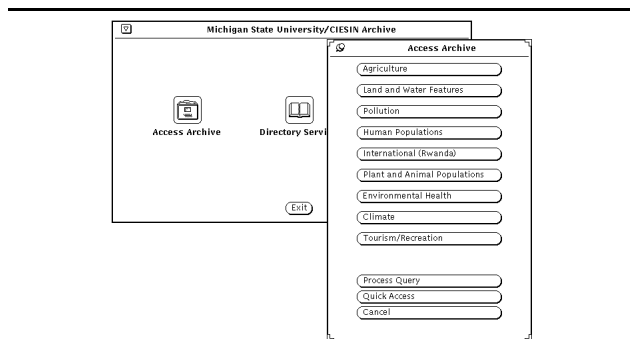


Figure 12: List of issues populating the system

The user may select a specific issue, which will bring up the *problem grid* associated to that issue. Suppose that the user is interested in *pollution* and specifically in determining what information is relevant to *groundwater contamination*. Figure 13 contains the problem grid for *pollution*. In each grid, problems are displayed on a scrolling panel that contains a scrollable list of aspects for each specific problem. By selecting the aspects within the problem lists, the user narrows the scope of the query. Checking the box at the top of a column selects all the problems in the archive. Notice that some of the aspects have caret symbols to the left of the entries, indicating that there is another level (refinement) of detail to the aspect. The values in parentheses indicate the number of items in the archive for that specific aspect. This feature is included in order to minimize the possibility of forming queries that have no applicable items. Disjunctive queries are formed by displaying multiple problem grids, where more than one instance of a problem grid for a given issue is allowed.

In Figure 13, the entire column of *groundwater contamination* problems have 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"

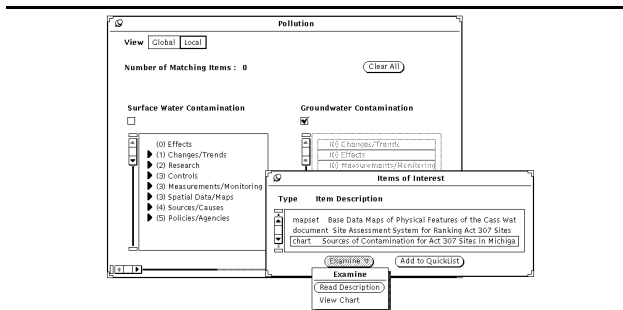


Figure 13: Problem grid and query results for *groundwater contamination*

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, *Read Description* and *View Chart*. Figure 14 displays examples of the different types of available items examined via respective manipulators, including a chart describing contamination sources and its textual description. It also shows a digital elevation map superimposed with watershed boundaries (from the mapset item) displayed via the GIS manipulator, OpenGrass, a simplified graphical interface that we developed to provide access to GRASS (Geographical Resources Analysis Support System) [7].

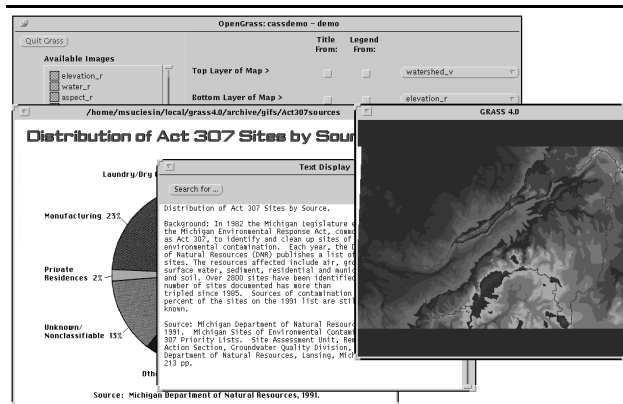


Figure 14: Sample types of available items and their respective manipulators

### 6.2 Preliminary Feedback from Users

The environmental science researcher, who may also act a consultant to policy makers, is the target user for the initial prototype. Such a user is assumed to be interested in accessing the vast amounts of heterogeneous environmental information available through the system. This user is *not* a computer expert, thus necessitating a user-friendly interface to all the tools needed to manipulate each item in the archive.

The system has been evaluated by users (e.g. policy analysts, township officials, federal agency staff, environmental science researchers from international

organizations) at workshops that specifically focused on environmental science and global change[1, 8]. Feedback from users has been used to refine the interface and add additional functionality. More specifically, the overall generic nature of the tool has yielded benefits outside the physical act of implementing the design. The configurability of the tool based on the IPAR classification scheme has allowed for the total reclassification of archive items between demonstrations, thus providing users with a completely different view of the watershed data without the need for changing the software implementation. The implementation also remained intact when new data sets and their corresponding classifiers were added to the archive.

### 6.3 Related Projects

The *International Institute for Advanced Systems Applications* (IIASA) develops environmental decision support systems [9]. IIASA systems are developed to be application specific. That is, an extensive analysis is performed by the IIASA team with a given client to determine specific details about a given problem. After the initial expert analysis is completed, tools are identified and models are constructed to capture the types of analysis that the experts would like to perform. While sophisticated, these systems are only usable for the specific problems for which they were developed and are static by nature. In contrast, ENFORMS can be completely reconfigured and repopulated by the user/administrator without any changes to the software itself. ENFORMS provides a dynamic archive that serves as a repository for information.

ENFORMS is perhaps more similar to the widely available information servers such as Gopher and WAIS. The main differences between ENFORMS and these systems are that ENFORMS presents information in an object-oriented fashion, is configurable to different search models, such as those that are hierarchical, temporal, and spatial, and is instantiated with data and interface components at run-time. Also, neither Gopher nor WAIS provide interfaces to GIS tools or models that can be instantiated with data dynamically.

## 7 Summary and Future Work

The tool, ENFORMS, provides access to data in two stages: the system is queried by the user for relevant data items, and located data items can be manipulated by a variety of integrated tools. Query support is achieved through an abstract representation of the data items stored in the archive. Analysis, design, and implementation using object-oriented principles has proven to be beneficial in the development of ENFORMS, resulting in a generic and highly configurable system. Much of the success in developing this system can be attributed to the high degree of software reuse achieved through the use of existing classes and inheritance of attributes and methods of base classes contained in the class library.

Currently the system is being extended to handle the integration of data from the distributed systems

into one environmental model. Also, we are investigating the parallelization of various applications available within ENFORMS exploiting the computing capabilities offered by clusters of workstations, particularly those connected by high-speed networks.

## References

- [1] R. H. Bourdeau, B. Pijanowski, and B. H. C. Cheng, "A decision support system for regional environmental analysis," in *Proc. of 25th International Symposium on Remote Sensing and Global Environmental Change: Tools for Sustainable Development (Vol. II)*, (Graz, Austria), pp. 223-233, 1993.
- [2] R. H. Bourdeau, B. H. Cheng, and B. Pijanowski, "A regional information system for environmental data analysis," *Journal of Photogrammetric Engineering & Remote Sensing* (accepted to appear), 1994.
- [3] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Sorenson, *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
- [4] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw Hill, 3rd ed., 1992.
- [5] R. Bourdeau and B. H. Cheng, "Distributed environmental multimedia archive: Software requirements specification document," Technical Report MSU/CIESIN WDP-CPS-1, Michigan State University, East Lansing, MI 48824, July 1992.
- [6] R. H. Bourdeau and B. H. Cheng, "Contextual EIS: Analysis and preliminary design of protocols for model layer and distributed access control layer," Technical Report MSU/CIESIN WDP-CPS-6, Michigan State University, East Lansing, MI 48824, January 1993.
- [7] J. Westervelt, *Geographical Resources Analysis Support System (GRASS): Introduction and Programming Manual*. U.S. Army CERL, GRASS Information Center, U.S. Army CERL, Champaign, Illinois 61826, July 1991.
- [8] B. Pijanowski, C. He, and B. H. Cheng, "Integration of human and natural science data for planning and management within a regional framework: The Saginaw Bay watershed project," in *Proceedings of the Building A Global Environmental Change Information Cooperative: First CIESIN Users Workshop*, November 1992.
- [9] K. Fedra, "Interactive environmental software: Integration, simulation and visualization," Technical Report RR-90-10, International Institute for Applied Systems Analysis, Advanced Computer Applications, Laxenburg, Austria, December 1990.