

Short Running Title: Object-Oriented Distributed Multimedia System
Contact Author : Betty H.C. Cheng
Address : Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
3115 Engineering Building
East Lansing, MI 48824 USA
Phone: (517) 355-8344
FAX: (517) 432-1061
email: chengb@cse.msu.edu

In the Annals of Software Engineering (Special Volume on Multimedia Software Engineering), Baltzer Science Publishers, Volume 12, December 2001, pp. 95-118.

Developing and Maintaining an Object-Oriented Distributed Multimedia Information System

Betty H.C. Cheng
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
chengb@cse.msu.edu

Gerald C. Gannod
Department of Computer Science and Engineering
Arizona State University
PO Box 875406
Tempe, Arizona 85287

May 31, 2001

Abstract

As object-oriented analysis and design techniques mature, the promises of object-orientation become more realizable. The advent of multimedia hardware and software technology give rise to a need for generalized multimedia data access methods. This paper describes the object-oriented development and subsequent maintenance of ENFORMS, an object-oriented distributed multimedia information system. We discuss the advantages and disadvantages of using object-oriented analysis and design techniques to develop a general client-server architecture for browsing, retrieving, and analyzing distributed multimedia data. We also describe lessons learned from this project, including the impact of a rigorous application of object-oriented analysis and design techniques.

1 INTRODUCTION

The widespread use of web browsers, Internet tools, and multimedia-equipped computers has created an enormous demand for quick and easy access to a wide variety of information. The need for electronic information is present in almost every application area and domain. From industrial organizations to the K-12 arenas, all share an appreciation and thirst for all that is offered by the “information superhighway.”

Correspondingly, an abundant supply of information exists to be perused by users. Many areas, such as scientific research addressing global change, continue 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. The users are forced to adapt to new browsing and presentation mediums with every new data set or information domain. In contrast to general information systems, such as the world wide web, decision support systems (DSS) must provide guided access to specific collections of information and support the appropriate analysis and data integration utilities. While a DSS must support sophisticated analysis utilities for large data sets, the DSS must also be flexible enough to accommodate changes in data sets, analysis tools, and even the browsing mechanism.

Object-orientation is particularly amenable to such demands placed on large scale information and decision support systems is object-orientation. Given their heavy emphasis on entity-relationships, object-oriented analysis and design techniques are particularly suitable for modeling information-intensive systems. Object-orientation also promotes reuse and facilitates maintenance. Object-oriented analysis and design enables the developer to model systems in terms of specific domains, rather than be constrained by the constructs of a specific programming language. Furthermore, because multimedia information systems have the additional property of supporting the integration of different types of data access and manipulation capabilities, object-orientation enables the developer to simplify the modeling, and thereby the implementation of the multimedia capabilities.

This paper describes how object-oriented analysis and design techniques have been applied to the development of ENFORMS [Bourdeau *et al.* 1993; Bourdeau *et al.* 1996; Cheng *et al.* 1994; Sharnowski *et al.* 1995], a distributed multimedia information and decision support system. ENFORMS comprises an integrated collection of software tools that allows a user to browse and manipulate disparate data sets through a graphical user interface (GUI). Figure 1 shows a high-level view of the organization, where the rectangles represent potential participating sites, and the filled circles represent users.

The object-oriented design of ENFORMS makes it extensible, flexible, and maintainable. The ENFORMS system is being used for three separate projects, each with its own set of data and tools. The same architecture has been used for all three projects. Using ENFORMS, decision-makers (who do not need to be computer experts) are better able to understand new environmental issues relevant to particular geographic regions.

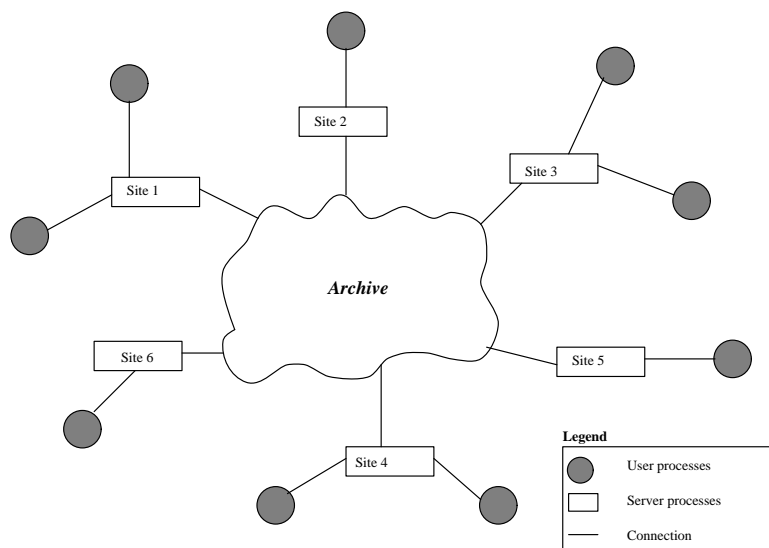


Figure 1: A high-level view of the ENFORMS structure.

The remainder of this paper is organized as follows. Section 2 gives background information in the area of object-oriented development techniques. Section 3 describes the object-oriented framework of ENFORMS and its evolution to satisfy both functional and nonfunctional requirements for different projects. Descriptions of specific ENFORMS applications are briefly overviewed in Section 4. Section 5 discusses lessons learned from this project over a number of years. Finally, concluding remarks and future directions for object-oriented frameworks for multimedia systems are discussed in Section 6.

2 OBJECT-ORIENTED ANALYSIS AND DESIGN

Object-oriented development techniques encompass analysis, design, and implementation. As with traditional development approaches [Yourdon and Constantine 1978], analysis identifies the functional and non-functional requirements of the system, design produces a family of solutions meeting those requirements, and implementation produces a specific solution obtained by refinement of the design. Graphical modeling techniques, such as data flow diagrams, state transition diagrams, and structure charts, facilitate the analysis and design stages.

As systems continue to grow in size and complexity, alternative development techniques are gaining increasing attention. One such alternative is *object-orientation* [Rumbaugh *et al.* 1991; Coad and Yourdon 1990; Wirfs-Brock *et al.* 1990]. The most significant difference between object-oriented development (OOD) and structured analysis and structured design (SA/SD) is the point of focus of the respective development techniques. In SA/SD, the focal point is the procedure or function. The analysis phase centers around

high-level descriptions of the functionality of the system. During the design phase, the refinement and decomposition of the high-level descriptions of functions yield more detailed descriptions of functions and procedures that incorporate implementation details. Finally, during the implementation phase, functions and procedures identified during design are decomposed into more specific functions. In contrast, OOD focuses on objects of the system and the relationships between the objects, where objects refer to real-world entities (e.g., servers, protocols) as defined by the developer. For each object, a set of attributes and specific operations (methods) for manipulating or accessing the object are defined. An object can only be manipulated by these operations. The implementation of these objects and their operations are “hidden” from users of the objects, thus facilitating the development and maintenance processes. Furthermore, concepts such as inheritance promote reuse in object-oriented systems. During the analysis phase, the developer identifies high-level objects and classes (collections of objects that share common properties) of the system, including the relationships (associations) between the objects. The design phase adds attributes and operations to the classes, refines classes, and introduces generalization relationships (i.e., subclasses). Finally, the developer identifies concrete data structures to implement the classes, implements the corresponding algorithms for the individual methods of the classes, and implements the associations between classes (e.g., subclassing, aggregation, binary associations).

Systems that involve significant volumes of data with well-defined operations are particularly suitable for object-orientation [Rumbaugh *et al.* 1991; Coad and Yourdon 1990; Wirfs-Brock *et al.* 1990]. Examples include database applications and information systems, where information-intensive systems, focus particularly on the data and its structure.

Object-Oriented Modeling Techniques. Currently, several approaches for performing object-oriented analysis and design exist [Rumbaugh *et al.* 1991; Coad and Yourdon 1990; Wirfs-Brock *et al.* 1990]. Most of these approaches involve some type of graphical modeling that facilitates communication among developers and between a “customer” and the developer by offering a visual (simplified) representation of the system prior to its implementation. The *Object Modeling Technique* (OMT) [Rumbaugh *et al.* 1991] has been commonly used in industry and in academia. (Recently, UML [Rumbaugh *et al.* 1999] has gained increasing attention for object-oriented modeling, where the OMT notation is included as part of the UML notation.) The advantages to OMT include its support for analysis and design, and the simplicity in notation. OMT comprises three complementary models, each of which is used to capture a specific perspective of the system. The *object model* describes the static, structural data aspects of the system. The object model captures the objects of the system and the relationships between the objects. This model describes “what” the system is. The *dynamic model* depicts the temporal and behavioral aspects of the system, thus describing “when” system activities occur. Finally, the functional model depicts the flow and the transformations of

the data within the system, which describes “how” the functionality of the system is achieved. Respectively, entity-relationship diagrams, state transition diagrams, and data flow diagrams are used to represent the object, dynamic, and functional models. Clearly, the notation for each model must be simple to construct and understand. Recent work [Bourdeau and Cheng 1995] enables rigorous analysis of each of the models, including consistency and completeness checks at the model level prior to the implementation phase.

3 THE ENFORMS ARCHITECTURE

The use of distributed information systems has seen a significant increase due to improvements in hardware and software technology. In addition, given the variety of data managed by these distributed information systems, increasingly, these systems have become distributed multimedia information systems. Many issues impact the longevity of these systems. Two key issues are *adaptability* to changes in the computing environment and *configurability* to changing data formats. This section describes the object-oriented development of ENFORMS, an information system developed at Michigan State University. For the remainder of this paper we use the convention that the term “ENFORMS I” refers to features and concepts related to the early version of ENFORMS, the term “ENFORMS II” refers to the features and concepts related to the subsequent version of ENFORMS, and the term “ENFORMS” refers to features and concepts related to both versions.

3.1 Goals and Motivation for Using Object-Orientation

The ability to cope with change is often a defining characteristic of whether or not a software system survives changing computing technology and environments. Given the significant advances in networking technology, information systems have flourished, but these systems must also be able to accommodate dynamic computing environments. That is, these systems must be flexible enough to support changing operating environments, changing data and data formats, and changing data analysis tools without requiring significant modification effort. In the context of information systems, three main properties that facilitate coping with changes are *adaptation*, *integration*, and *configuration* [Gladney *et al.* 1994].

Adaptation is the degree to which the architecture and design of a software system can facilitate modifications. In addition to the *maintainability* of software code, the adaptability of a system is also impacted by the products of good software engineering, including the existence of a set of requirements specifications, design specifications, and testing specifications that have been appropriately modified to reflect modifications in software code.

Integration refers to the degree to which an information system can incorporate new sets of data that may be different in format and nature to any data currently residing in the system. In addition to being able to incorporate this data, integration is impacted by the ability to manage these new resources in such a way that the core data is hidden from end users, but where the access methods are well-defined. Media integration is often associated with the ability to construct new multimedia objects (i.e., authoring). To this end, an information system with a high-degree of integration should provide the facilities for associating objects and their access methods while seamlessly incorporating these new objects into the system.

Finally, *configuration* is the degree to which an information system can cope with different domains of data. Configuration can cover many areas related to information systems including browsing, access, and retrieval methods. That is, the ability to tailor a system at run-time so that the browsing, access, and retrieval methods facilitate activities natural to the domain of interest makes a system highly configurable.

The remainder of this section describes the development of ENFORMS that has been designed and implemented with the intent of maximizing the three criteria of adaptation, integration, and configuration through the use of object-oriented technology and a rigorous development process.

3.2 Analysis Guidelines

Distributed multimedia information systems can be generally considered under the broad umbrella of digital libraries that generally share the same underlying constraint; they are based on the search, selection, and delivery of information that has been aggregated into some form that can be distributed easily. The means by which that data is disseminated depends on the nature of the data as well as the methods that have been used to catalog (classify) the data.

An *item* can be a single data entity or a collection of data that has been aggregated into a semantically related group or *resource*. As a group, an item is known as an *aggregate item*. Single data entities are referred to as *atomic items*. The set of programs that define the semantics for the item are referred to as the *item manipulators*. An *Item classification* scheme, or *search model* is the method that is used to catalog or *classify* items. The following guidelines provide a systematic method for developing information systems that are *item* and *item classification* centered. The steps are:

1. Analyze characteristic data
2. Choose classification model
3. Analyze the storage and ownership constraints

3.2.1 Data Analysis

The data that is collected for use by an information system can determine the methods that will be used to access the data. Before an information system can be built, representative data to be accessed must be identified and analyzed. Specifically, there are two types of analysis activities:

- Atomic item access
- Aggregate item access

Atomic item access refers to the methods that will be used to manipulate atomic data items. For instance, when given an image, specific tools must be defined that allow for the manipulation of that image. *Aggregate item access* refers to the set of methods required to manipulate aggregate items.

Simple modeling of the items located in an information system archive serves as the basis upon which the remainder of the system is built. Decisions about how to browse, search, and retrieve items are all based on the items and their access methods.

Many activities in the browse and search of items involve abstract representations of the actual items. Although network technology has improved in performance, the delivery of actual items can still be a source of delays in response time. An *item descriptor* is a meta-item that describes the characteristics of an item. Item descriptors are smaller in size and avoid unnecessary network traffic. By using item descriptors, delivery of actual items is only performed when a specific manipulation of an item is requested by the user.

3.2.2 Search Models

Another important activity in the analysis of representative data for an information system is the creation of classification paradigms or *search models*. A search model determines how a collection of items (either atomic or aggregate) are classified so that a system can be browsed and items can be retrieved using a particular *search engine*. Examples of classification methods are *hierarchical*, *spatial*, and *temporal*. In addition to facilitating browse and retrieval, the search model determines how new items are added to an archive.

3.2.3 Storage and Ownership Constraints

The constraints of data storage (where the data is stored) and data ownership (who has access to the data) can have an impact on the architecture and topology of an information system. These constraints determine whether or not a system uses a client-server architecture, or whether it is a protected single machine system. The impact of these decisions play major roles in the delivery and access of items.

3.2.4 Object-Oriented Analysis

Analyzing characteristic data, choosing the appropriate classification model, and determining the storage and ownership constraints make a direct exploitation of the advantages of object-oriented analysis and design. The focus of an information system is an item. Just as an object in the implementation of a software system has both data attributes and methods, items have attributes (the data itself) and methods (the manipulators).

By separating the data from a specific search model, support for cross-referencing across search models is facilitated. This ability is made possible by the treatment of items as autonomous data objects. In addition, the use of item descriptors facilitates the maintenance of the partition between the search model and the data items. As such, flow of data during browse and search is limited to small descriptions of items rather than large collections of items.

3.3 ENFORMS I

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 has 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 [Cheng *et al.* 1994]. The graphical notations of the *Object Modeling Technique* (OMT) [Rumbaugh *et al.* 1991] are used to represent all models. Due to space constraints, only the object models are presented here. A more detailed description of the object-oriented analysis and design, including all three OMT models can be found in [Gannod and Cheng 1996].

Figure 2 gives the object model for the ENFORMS architecture, in which the model is the result of several iterations of refinement. 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 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 the discussion, the bold roman font is used to denote class names.

One of the primary goals of the ENFORMS project has been to provide users with convenient access to large amounts of multimedia information that may be distributed across many sites. As such, Distributed

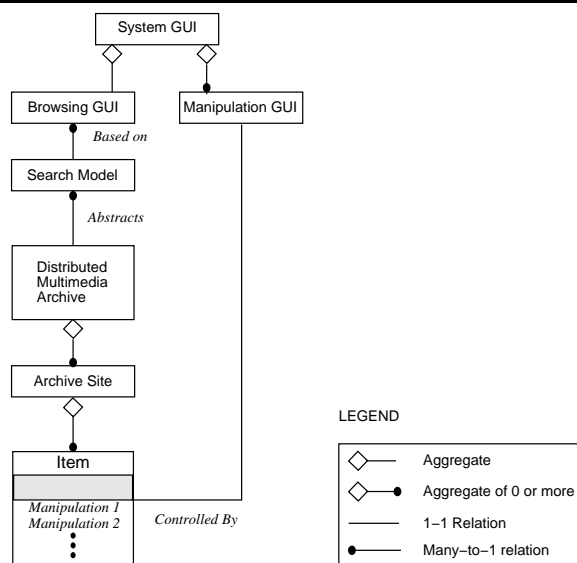


Figure 2: Object model for ENFORMS I

Multimedia Archive comprises zero or more Archive Sites, each of which contains zero or more Items. 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*.

The system is accessed through 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 GUI*s based upon it, where a search model defines how items are managed, classified, and retrieved. Examples of different search models include spatial, temporal, and hierarchical. *Items* are *Controlled By* multiple *Manipulation GUI*s; 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 GUI*s.

Search Model. Figure 3 shows the object model for a generic search model. A *Search Model* is an aggregation of *Classification* and *Item Classification*. An *Item Classification* is a construct for describing how a given item fits within a given *Classification* scheme. For instance, in the domain of zoology, the Bobcat “item” falls under the feline “classification”. In the geography domain, the North America “item” could fall under the western hemisphere “classification”. A search model provides the means by which a classification scheme can be browsed for subjects of interest. These subjects can then be used to create a *Query Formula* that enable the system to match the *Item Classification* to the specific *Item Descriptors* via the *Local Control*

Broker. As shown in Figure 4, the Local Control Broker realizes the conceptual aggregation of Archive Site 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, where each manipulator is parameterized by its own descriptor including parameter values, brief description, and title presented to the user (e.g., “View Image”).

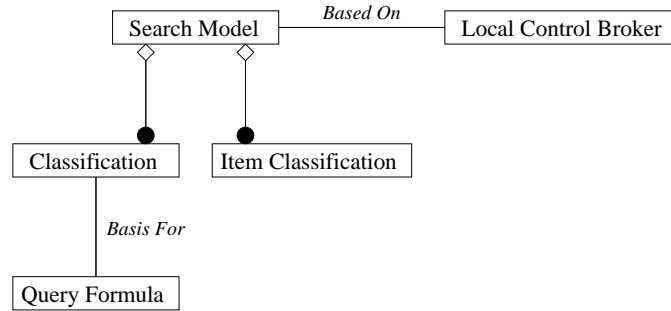


Figure 3: Generic Search Model

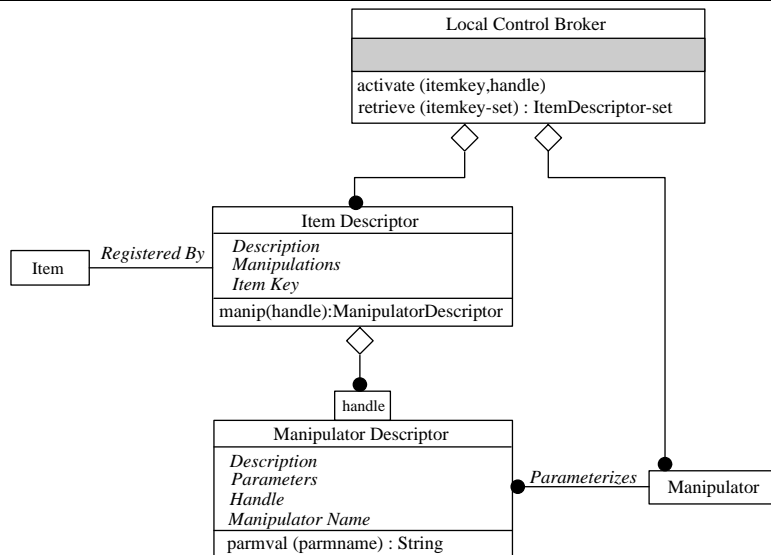


Figure 4: Analytic model of Archive Site

An interesting characteristic of the Search Model object model is the lack of an association between Classification and Item Classification. This feature allows items to have multiple Item Classifications. For instance, in the environmental science domain, the “ozone” item can fall under both the “atmospheric component” and as a “harmful gas” classifications.

ENFORMS I 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 5. 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 5 contains a simple example hierarchy for the data of the system.

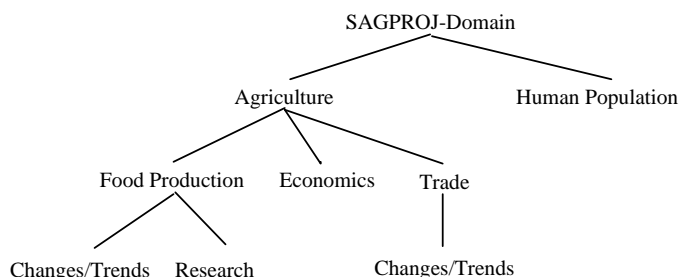


Figure 5: An example IPAR classification hierarchy

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

3.4 ENFORMS II

Two major differences distinguish ENFORMS I from ENFORMS II. First the distributed framework was refined to a client-server architecture. Second the search model component was expanded to handle multiple search models simultaneously.

Client-Server Framework. We refined the distributed framework further by extracting the appropriate components of the Search Model and Archive Site classes (shown in Figure 2). This refinement facilitated the derivation of 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 6 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 6 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 IPAR model as well as search models that allow for spatial- and temporal-based searches.

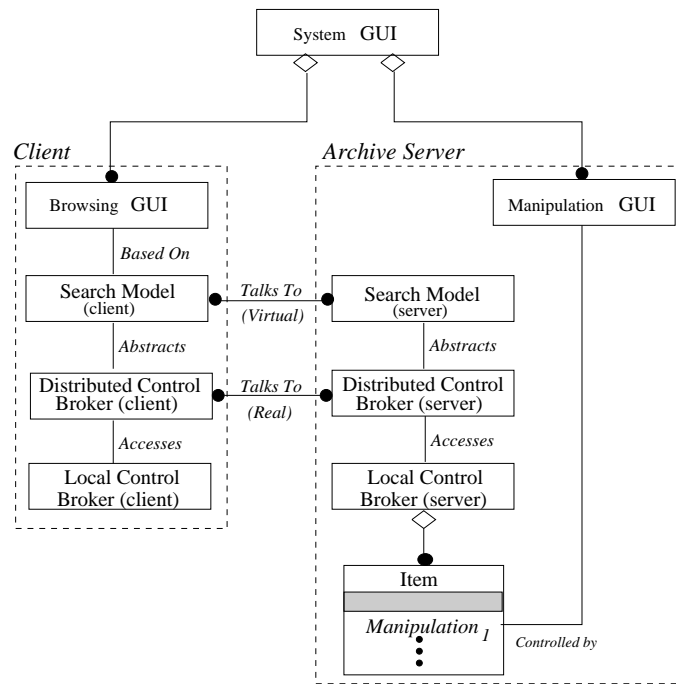


Figure 6: The client-server architecture

Expanded Search Models. *Items* in the ENFORMS I system were restricted to a single search model paradigm, that is only one search model was available at a given instance of the system. However, the design of the system allowed different models to be defined for other instantiations. Furthermore, for the early systems, the items were accessible only through a hierarchical search mechanism. In contrast, the ENFORMS II supports in addition to hierarchical search, the use of *spatial*, *temporal*, and *hypertext* search, all of which are available within a single instance of the system. Figure 6 depicts the new architecture and identifies the client and server portions of the system. The major changes from the ENFORMS I architecture are that the *System GUI* now has zero or more *Browsing GUIs* and that each *Browsing GUI* has one and only one *Search Model*. Thus, ENFORMS II supports multiple search models each with its own browsing GUI.

Figure 7 gives the refined object model for the *Search Model* class. Each search model is made up of a

domain dictionary that defines the domain for the query construction, and a classification index that contains itemkeys that are indexed, based on the query formula for each search model. Each itemkey identifies an item descriptor contained in the local control broker that registers individual items.

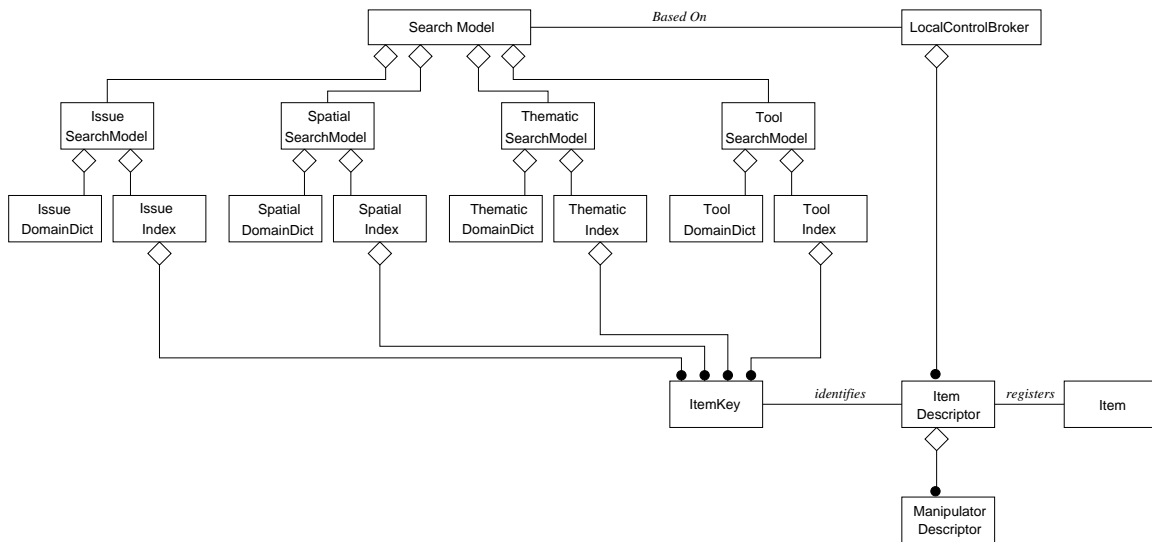


Figure 7: Search Object Model for ENFORMS II

3.5 Classification Software

A main feature of the ENFORMS system is the ability to easily add new items to the archives. To facilitate that activity, we developed an administrative system called CLASSIFY. This tool supports two primary activities. First, it enables data custodians (maintainers of data) to create or modify item classifications. That is, item classifications can be modified based on the search model(s). Second, CLASSIFY enables data custodians to create new items by defining the data and the set of tools that provide access to the data.

Domain Classifications. In both versions of the ENFORMS system, a **Classification** is a class that implements a cataloging feature. For instance, items in ENFORMS II can be classified (catalogued) hierarchically and/or spatially. The CLASSIFY tool allows a data custodian to construct a *classification scheme* for the domain of items contained in an archive, thus termed a *domain classification*. Using the products of the CLASSIFY tool at run-time, ENFORMS instantiates a graphical interface that is displayed to the user during the browse activity.

Item Creation. There are two activities that must be performed when a data custodian needs to add a new item to an ENFORMS archive. An item must first be *described*, and then the item must be *classified*. Both of these activities are supported by the CLASSIFY tool.

The act of describing an item using the CLASSIFY tool essentially allows a data custodian to create an **item descriptor** (meta-data description of type, manipulators, classification). As such, a data custodian is responsible for defining the access methods for the new data item (i.e., the item manipulators) by either using an existing manipulator, or by describing new manipulators. Once an item has been properly described, it must be placed into the archive. Placement of an item into the archive requires describing the item using an **item classification**, an activity also supported by the CLASSIFY tool. As was the case with the domain classifications, the products of the CLASSIFY tool are used by ENFORMS at run-time to determine the presentation of the archive contents to the user.

4 APPLICATIONS

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, 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.

The ENFORMS project has been a multidisciplinary effort involving researchers from more than twenty-two different departments from Michigan State University and numerous scientists from CIESIN (Consortium of International Earth Science Information Network), EPA, USDA, and NASA. ENFORMS has been designed to provide access to data and data integration utilities in order to facilitate decision making processes relevant to environmental policies. The user constraints included a user-friendly system, usable by non-computer scientists, focused search capabilities, and transparency of the distributed and heterogeneous nature of the data and analysis utilities. 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.

Based on the original object-oriented architecture, there have been three separate instantiations of

ENFORMS for use in three disjoint projects. The objectives for each of the projects are described briefly below in chronological order. For each project, a description of the user needs analysis is given, followed by a discussion of the data types and data processing utilities needed by the respective projects. It is noted that with each instantiation of ENFORMS, the majority of the original architecture was preserved. The most significant difference between ENFORMS I and II is in the search/browse models. Different data manipulators are included for the three instantiations, but the incorporation of different manipulators does not require any changes to the overall architecture. For a given instantiation of ENFORMS, the CLASSIFY tool was used by the application scientists to define the classification hierarchy, individual item classification, and individual item descriptors (e.g., textual descriptions, relevant manipulators, associated items).

Saginaw Bay Watershed Regional Analysis. The objective of the Saginaw Bay Watershed Regional Analysis project (under sponsorship from NASA) was to focus on the sustainability of food production systems in the Saginaw Bay Watershed region in Michigan, with a particular emphasis on projections into the coming century. Ongoing work by numerous scientists at Michigan State University (MSU) was used to accelerate the user needs analysis. These researchers analyzed responses from user surveys in order to determine the most pressing questions involving human use of natural resources in food production. From the user survey, it was determined that the relevant areas of study included land and water features, agriculture, animal populations, human populations, pollution, climate, and environmental health.

Data and Analysis. Application science members of the project team analyzed the user input in order to identify data and the different types of data analysis that would help to address some of the users' concerns. Once the data was identified and acquired, it was classified according to the perspective obtained from the survey study. Types of data included textual documents, photographs, images, graphs, satellite imagery, georeferenced data, and digital elevation maps. The system also provides access to parameterized empirical and predictive models for the studies of insect population and agriculture production in the region, respectively.

US/Mexico Border Water Quality and Quantity. ENFORMS I has also been used to facilitate data access and analysis for information specific to the El Paso/Ciudad Juarez area in west Texas, southern New Mexico, and Mexico. The specific objective of the USDA sponsored project was to provide a graphical-based environment for accessing the integration of socioeconomic and water quality/quantity data.

The area of focus is a critical region receiving a great deal of attention due to the North American Free Trade Agreement. In general, water quality and water quantity issues are of vital concern to planners in the region because of the arid climate, limited water supply, forecasted increases in population, and the

need for water in agricultural production in the region.

Data and Analysis. The project specifically focused on water quality and quantity issues in the US/Mexico border region. The data relevant to the project included information for both surface and ground water. The system provided access to three major types of data for the project: text data, spreadsheet data, and image data.

The users, comprising research scientists and regional planners, emphasized the need to be able to perform analysis of the spreadsheet data related to water quantity and water quality in the wells supplying groundwater to the region. In addition to standard spreadsheet analysis, the users also requested the ability to dynamically generate graphs from the spreadsheet data that enabled them to determine correlation between different parameters of well data. We developed graphing utilities to support the integration of temporally referenced data and individual parameters, such as chemical levels, well pumpage rates, and water levels. All of the data analysis needs were handled by providing access to the appropriate manipulators, some of which were developed by the ENFORMS team. The application scientists, again based on user input, used the CLASSIFY utility to construct the class hierarchy, classify individual items, and define manipulators for the items. There was no need to change the core of the architecture of ENFORMS I to include the new manipulators.

Great Lakes Regional Decision Support System. An *environmental quality assessment* is defined to be a process that delineates the extent of environmental impact within a given geographic area. It involves the identification of potential sources of contamination and an evaluation of the type and degree of ecological impairment within the given area. A *monitoring needs assessment* is defined as a process to evaluate the adequacy of monitoring efforts for a given set of parameters and/or geographic area and to identify further monitoring requirements. The objective of ENFORMS II (under sponsorship from the EPA) was to study the impacts of water monitoring at a regional level (Great Lakes) on environmental quality and monitoring needs assessments.

Due to the heavy emphasis on spatially and temporally-referenced data, the ENFORMS search model and analysis utilities were expanded significantly, thus warranting the name ENFORMS II.

Data and Analysis. Historical monitoring data is extremely important in the assessments of this project, providing insight into environmental change and, more importantly, trends. Several data sets were identified to be important to water quality assessment activities.

The two main data types are *georeferenced point data* and *ancillary data*. Georeferenced point data (also known as *point data*) is a set of attribute values that is indexed using the latitude and longitude of

the site where the data sampling took place. Ancillary data consists of images, charts, documents, MPEG movies, etc. Although ancillary data may be specifically related to a particular geographic area, the main characteristic that differentiates ancillary data from point data is that point data contains a *sampling* of some attribute, while ancillary data serves to describe, explain, show, or illuminate facts about the subject being examined. Point data more naturally lends itself to further analysis by Geographic Information Systems (GIS), spreadsheet programs, or statistical analysis packages.

There are four different kinds of analysis that are available in the ENFORMS II: Point Tool, Full Spatial, Guide, and IPAR. Point Tool is a tool that supports the manipulation of spatially referenced data that is not numeric in nature (i.e., it is ancillary). Full Spatial analysis supports the manipulation and analysis of spatially referenced tabular numeric data via the use of some Geographic Information System (GIS) tool. Guide analysis provides facilities for viewing hypertext documents related to certain environmental areas of interest. Finally, IPAR analysis supports the manipulation of data items based on their access methods.

One of the IPAR access methods is parameterized modeling. The system supports both predictive model-based analysis and historical data analysis, both of which required sophisticated visualization and animation capabilities. The analysis of the historic data have both spatial and temporal dimensions, which enables users to analyze data based on spatial location or time periods. Most of the spatial analysis requirements can be carried out with the help of a particular geographical information system (GIS), which is designed specifically to manipulate the spatial relationships.

5 LESSONS LEARNED

The ENFORMS project has evolved significantly over the past several years. The system has been used to satisfy the requirements for three disjoint and separately funded projects. During the evolution, the development team applied rigorous software engineering practices in the context of object-oriented analysis, design, and implementation.

The remainder of this section overviews the development process and describes three areas of lessons learned from the ENFORMS project. First, we discuss the impact of the execution of a rigorous application of object-oriented analysis, design, and implementation. Second, the advantages to using object-orientation for the ENFORMS project are discussed. Finally, we share our views on the perceived disadvantages of using object-orientation with respect to this project and the current state of the art in object-oriented technology.

5.1 Overview of Development Process

For each project, four tasks were performed in order to understand the requirements of the project. First, a *user needs analysis* was performed. The objectives of this study were to identify the users of the system, what problems are of interest to users of the system, and what types of answers are sought for the questions. Second, based on the results of the user needs analysis, a group of application scientists assisted in identifying the sets of data to be used to address the problems and answers of the users. In most cases, the required data came from different sources, were stored in numerous formats, and required different access utilities. Therefore, numerous utilities have been developed to provide unified access to the relevant data, making the heterogeneous formats and distributed nature of the data transparent to the user. The third task was to determine what types of analysis tools were needed by the users in order to manipulate and integrate the data as a means to find answers to the questions as well as facilitate the decision making processes. The final task was to determine how the information in the system should be presented to the user. This task included determining the search (browsing) model, the type of interface to the analysis tools, and the type of presentation for analysis results. This last task relied heavily on the use of the CLASSIFY utility as it provides a user-friendly mechanism to be used by the development team and the data custodians for determining the presentation of information, how data is accessed and manipulated, and a facility for classifying the data.

Once the requirements were gathered the development team spent a significant amount of time developing graphical models of the system, first from the analysis perspective, focusing on desired functionality. “Throw away” prototypes were used to facilitate communication with the customer to ensure an accurate understanding of the customer’s requirements and expectations. In addition to modeling the system architecture, the structure of the data was also modeled to facilitate the development of the browse/search component of the system. The analysis models were then refined and implementation details added to obtain design models. The design models and design document described specific algorithms used, the interface between the various parts of the system, identified the classes from class libraries that were used in the implementation, and design constraints imposed by specific customer conditions, such as target environment, platform, and integration requirements with existing systems. In general, models were refined several times during the analysis and design stages. Based on the details from the design document, the development team implemented the system in the language C++. It is emphasized that the analysis and design stages of the project were critical to the overall success of the project. As a means to facilitate technology transfer, we specifically requested that the analysis and design documents be included as deliverables for all projects. Based on the success of the ENFORMS project, the different project sponsors have since requested similar documentation from other projects.

Finally, configuration management played a critical role to the success of the overall development

process and the entire ENFORMS project. Configuration management is an absolutely necessary component of any reasonable-sized project involving a team of developers. Given that there was temporal overlap between projects, the development team was forced to practice disciplined use of configuration management for all types of changes to the system and to the documentation. After identifying a need for change, the team reviewed a description of the change and its impact (local and global). After the change was implemented, it was thoroughly reviewed and tested (for implementation changes). Testing occurred both locally and at the global level (localized function and global impact). Next, the change was tested in the customer's environment to ensure consistency with previous functionality and environment. Finally, a new build of the system was derived and released to the customer.

5.2 Impact of Development Process

In many instances, the customer had a high-level view of the objectives of the system. The use of models during the requirements understanding and analysis stage greatly facilitated communication between the development staff and the customer. Particularly, when used in conjunction with the prototype, the models assisted the client in understanding the overall structure of the system and its potential for expansion for future needs. Also, the team benefitted from group interaction during the construction and refinement of the graphical models. Particularly during the beginning stages of a new project, the modeling task enabled significant discovery about many aspects of the projects. The models played a critical role in training new development team members to work on the project. The original development team consisted of four computer scientists working with more than twenty application scientists and users. During the evolution of ENFORMS, staff members have turned over and the development team has grown to a size of ten to fifteen developers working with new and original application scientists. Based on the graphical models, it was straightforward to understand the overall architecture, thus enabling even new staff members to take one component and refine it with design and implementation details. Because of the simplicity of the models, even staff members who were not familiar with object-orientation were able to quickly learn the basics and participate actively in the development process.

Critical to the development process was the continuous feedback from users of the system. In all instances of the ENFORMS system, preliminary prototypes were made available to the customer and representative customers for feedback on the interface, search/browse framework, data analysis utilities, and presentation of results. Because object-orientation was used to decompose the system into its major components, changes to any of the above areas were localized and had little impact on remaining system elements. In the case of the interface and search/browse framework, the use of meta-data descriptions of data enabled the development team to give a completely different appearance to the interface and browsing by allowing

the application scientists to change the data classification scheme and individual item descriptors (using the CLASSIFY utility). The impetus for this particular change was feedback obtained from user workshops in which representative users for each of the respective projects were asked to use prototypes of the system to browse for information and perform analysis on data relevant to their specific environmental concern.

Finally, because the development team followed a rigorous development process guided by design, the actual development of source code was straightforward and time devoted to its completion was less than 20 percent of the overall project effort. The object-oriented design of the system minimized the dependency between different components of the system, thus enabling the developers to maximize parallel effort during the coding and preliminary testing stages. The integration of the different components was straightforward and required no rework of the design. Particularly, in the case of ENFORMS II, where the complexity of the system grew significantly in terms of data, search engines, and data integration utilities, the smooth integration of the numerous components pleasantly surprised the entire development team. The configuration management rules enabled the development and maintenance of several versions of ENFORMS simultaneously. Some versions differed only by the graphics library used for the user interface or differed by the operating system environment. Nonetheless, the configuration management system was the only reasonable framework that enabled the development team to preserve the integrity of the different versions, while supporting a systematic evolution of the system.

The administrative utility CLASSIFY significantly reduced the burden on the development team since application scientists were able to handle all aspects of data presentation, organization, and manipulation association. Users of CLASSIFY had no need to know about the design or implementation of ENFORMS, nor was there a need to be a computer expert.

5.3 Advantages to Using Object-Orientation

The advantages to using object-oriented design and implementation are discussed in the context of maintenance, flexibility, and reuse.

Maintenance and Evolution. The overriding advantage to using object-oriented development techniques for this project was the ease in which changes could be made during the maintenance phases and the re-engineering of ENFORMS. During the beginning stages of the ENFORMS project, the requirements from the customer were quite high-level. As a working system evolved, the customer, through the use of the system, developed new requirements and expectations for the system. The object-oriented design and implementation enabled the development team to incorporate the new features without significantly changing the underlying architecture. Part of this ease is due to the flexibility built into the original architecture and the support for abstraction and information hiding characteristic of object-oriented frameworks.

Flexibility. The object-oriented design provided significant flexibility in the maintenance of data types, data manipulators, and data integration utilities supported by the system. Given the wide variety of data used in environmental science study, this flexibility was absolutely essential to the success of the project. Also, the use of the meta-data descriptors made the system configurable at run-time. That is, the meta-descriptions for the data classification, individual item descriptions (e.g., “Map of Great Lakes Region”, “Animation of gypsy moth growth”), and item manipulators (e.g., image viewer, text search, data model) enabled the actual graphical user interface for the system to be determined (configured) at run-time. Therefore, for a given set of data, depending on the meta-data descriptions, the resulting system could look significantly different, including menus, buttons, and other graphical interface components.

Another aspect of flexibility afforded by the object-oriented design is the ease with which modifications to specific components could be made without affecting the others. For example, based on customer’s needs, the original graphical user interface was developed using Sun’s OpenWindows graphics library. Another customer required the use of Motif to enable the integration of ENFORMS into a larger framework. The customer also wanted a different presentation style for the search component. Because the graphical user interface was developed as two separate entities, one specific to a graphics library and a “shadow” layer that provided generic GUI services and was interfaced with the underlying architecture. This separation enabled us to develop a completely different GUI (not just a translation between Motif and OpenWindows library calls) that required only the generation of the new Motif interface that was then integrated with the appropriate components of the “shadow” layer.

Another example of flexibility is the ease with which the port of the ENFORMS system to different operating environments was completed. Currently, ENFORMS operates in three UNIX-based environments: SunOS 4.1.X, Solaris, and Linux. While all three are UNIX-based, there are significant differences in the way system-oriented services are handled, such as networking and general communication services. The port from SunOS to Solaris and Linux were both handled in a straightforward fashion, where the changes were limited to the distributed services component and portions of the graphical interface component (not the “shadow” layer). In most of the cases, the changes were limited to a small number of classes.

Reuse. This project has reaffirmed to the entire development team that object-orientation does indeed promote reuse. Reuse is viewed from both the fine-grained and coarse-grained perspectives. Fine-grained reuse was in the form of individual classes that were reused, both from a class library and through inheritance. Coarse-grained reuse occurred in the form of analysis and design reuse, as well as entire components of the source code being reused. The latter case occurred in the evolution of ENFORMS I to ENFORMS II. On a smaller scale, coarse-grained reuse occurred when ENFORMS was ported to different operating platforms or adopted a new look through a new graphical user interface. Of course, the most significant example of

coarse-grained reuse is the use of ENFORMS for two separate applications and two separate customers (NASA and USDA). The entire architecture was reused, and all that changed was the meta-data descriptions, data, and the manipulators.

Lessons Specific to Multimedia Applications. While many of our lessons learned from the development of ENFORMS can be applied to non-multimedia applications, we found that several aspects of multimedia applications enabled us to maximize the advantages of object-orientation. First, multimedia applications typically involve multiple types of data requiring different types of processing utilities, such as video players, audio players, etc. For the purposes of ENFORMS, the user was interested in performing different types of environmental analysis and should not need to know which type of data viewer or manipulator should be used to process the data of interest. Object-orientation enabled us to create generic data objects, each of which had one or more manipulators. After the user's query, the system could return the appropriate data objects and invoke the appropriate data manipulator, depending upon the user's analysis requests. This type of abstraction enabled us to maintain flexibility and extensibility in determining which manipulators were appropriate for a given object type. As the system evolved, we could continue to add more manipulators without changing the architecture of the system. Second, multimedia applications typically involve some type of data integration of different types of data, such as audio with video or animation of GIS maps. Object-orientation enabled us to use inheritance as a mechanism to define different types of data integration capabilities based on the types of environmental analysis needed for the system. Third, because multimedia applications tend to involve a number of different types of data manipulators and data integrators that can be integrated and composed in a number of different combinations, the use of object-orientation greatly facilitates the reuse and integration of these manipulators.

5.4 Disadvantages to Using Object-Orientation

Most of the disadvantages to object-oriented development surfaced during the implementation and testing phases. The development team searched for some type of standard class libraries for this project. None at the time seemed suitable for our needs. In the end, we modified an existing library to develop a hybrid. The library lacked detailed documentation, thus making it difficult for new development team members to be proficient in its use. While the abstraction offered by object-orientation during analysis and design is invaluable, during implementation, it can cause potential problems. For example, a development team member may learn about a specific method that provides a needed service. But due to the class hierarchy, it may be very difficult to actually identify the class to which that method belongs. The abstraction infrastructure also made it difficult to detect the source of memory leaks. Finally, global objects facilitated reuse, but also caused the same type of problems that global variables cause in structured programming,

that is, global side effects.

Currently, no commonly accepted standards for testing object-oriented software exist, even though several techniques have been proposed [Binder 1994]. As a result, we developed a hybrid approach based largely on traditional testing techniques. First, we applied unit testing at the individual method level. Next, we performed integration testing between classes based on the different types of associations (e.g., aggregation, inheritance, and binary). Finally, we applied a variety of system tests, including alpha, beta, and validation (acceptance).

From the modeling perspective, a few deficiencies with OMT were identified. While there are clearly advantages to modeling a system from complementary perspectives, where each modeling notation is simple, there are also disadvantages. The most prominent disadvantage is the lack of a well-defined technique for integrating the three models. This lack of a well-defined integration introduced some confusion during the implementation stage, particularly in the structure of some of the methods. It is anticipated that as the object-oriented technology matures, refinements will be made to current modeling techniques to overcome their current deficiencies. Towards this end, we have recently developed formal semantics for all three OMT models [Bourdeau and Cheng 1995; Wang *et al.* 1997; Wang and Cheng 1998a; Wang and Cheng 2000], including a systematic process for constructing the models [Wang and Cheng 1998b] that explicitly addresses the integration of all three models. The formal semantics have enabled us to perform automated analysis of the OMT models for ENFORMS [Campbell *et al.* 2000], thus serving as a complementary technique to testing for verification and validation of the system.

Finally, the system has been implemented in the C++ language. The language, itself, caused a few problems since not all members of the development team were proficient in its use and familiar with all the subtle features when initially working on the project. This lack of a complete understanding of all the features of the language caused a few cases of code redundancy and introduction of memory leaks.

6 CONCLUSIONS AND FUTURE INVESTIGATIONS

ENFORMS provides access to data in two stages: the system is queried by the user for relevant data items, and located data items can then be manipulated by a variety of integrated tools. Tool integration is one of the key features of the system. Query support is achieved through an abstract representation of the data items stored in the archive; this abstraction is implemented in the search model of the system architecture.

We have found that the object-oriented design and implementation greatly facilitated the development process with respect to the integration of components developed by a group of people and the extensibility of the system for new requirements obtained from user feedback. Because the data and the integration tools

are not “hard-coded” into the software framework, ENFORMS can be modified to accommodate new data integration tools and datasets from other regions of interest. Furthermore, the data classification scheme can easily be changed, thus defining a new interface for the system that is tailored to a specific audience or type of user. Due to the rigorous software development process practiced by the software engineering team, the results from this project have been used for more than one specific application.

It is anticipated that with continued use of ENFORMS, additional functionality may be desired by existing and new clients, including new search mechanisms, new mediums of data, new data integration facilities, and operation on new platforms. We are investigating the integration of distributed models that are applicable to datasets residing on the same or different sites, where the distributed nature of the models and the data is transparent to the user. Also, with the widespread existence of multimedia authoring utilities, we are also exploring the automated and dynamic generation of multimedia formats of user-retrieved data (e.g., generation of animations of predictive model results). Currently, we provide manipulators for playing animations that have been previously created. ENFORMS has also been used to address decision support activities in the manufacturing domain for both commercial and federal projects. This latest instance of the ENFORMS project specifically addresses clients accessing the system through the world-wide web.

7 ACKNOWLEDGEMENTS

This work has been sponsored in part by EPA, NASA, USDA, Consortium for International Earth Science and Information Network (CIESIN), Michigan State University, Defense Logistics Agency, and NSF grants CCR-9209873, CCR-9407318, and CCR-9633391. The success of the ENFORMS project has only been possible due to the devoted efforts of a number of people, including Robert Bourdeau, David Robinson, Steve Schafer, Joe Sharnowski, Yonghao Chen, Dan Judd, Paul Fraley, Heather Richter, Stephen Wagner, and Enoch Wang. Several application scientists provided significant effort in the organization of data, classification, and usability of the systems: Bryan Pijanowski, Stuart Gage, Jon Bartholic, Lois Wolfson, T. Kang, and Todd Zahniser. Finally, this work was performed while the second author was a doctoral student at Michigan State University.

REFERENCES

- [Binder 1994] Binder, R. (1994), “Object-oriented software testing,” *Communications of the ACM* 37, 9, 28, special issue on object-oriented testing.
- [Bourdeau and Cheng 1995] Bourdeau, R. H. and B. H. C. Cheng (1995), “A Formal Semantics of Object Models,” *IEEE Trans. on Software Engineering* 21, 10, 799–821.

- [Bourdeau *et al.* 1996] Bourdeau, R. H., B. H. C. Cheng, and B. Pijanowski (1996), "A Regional Information System for Environmental Data Analysis," *Photogrammetric Engineering & Remote Sensing*, 7, 855–861.
- [Bourdeau *et al.* 1993] Bourdeau, R. H., B. Pijanowski, and B. H. C. Cheng (1993), "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.
- [Campbell *et al.* 2000] Campbell, L., B. H. Cheng, and E. Y. Wang (2000), "Enabling Automated Analysis Through the Formalization of Object-Oriented Modeling Diagrams," In *Proceedings of IEEE Dependable Systems and Networks (FTCS-30 and DCCA-8)*, New York, NY.
- [Cheng *et al.* 1994] Cheng, B. H. C., R. H. Bourdeau, and G. C. Gannod (1994), "The Object-Oriented Development of a Distributed Multimedia Environmental Information System," In *Proc. of 6th International Conference on Software Engineering and Knowledge Engineering*.
- [Coad and Yourdon 1990] Coad, P. and E. Yourdon (1990), *Object-Oriented Analysis*, Yourdon Press, Prentice Hall, Englewood, New Jersey.
- [Gannod and Cheng 1996] Gannod, G. C. and B. H. C. Cheng (1996), *Multimedia Information Storage and Management*, chapter The Object-Oriented Development of Multimedia Information Systems, Kluwer Academic Publishers.
- [Gladney *et al.* 1994] Gladney, H. M., E. A. Fox, Z. Ahmed, R. Ashany, N. J. Belkin, and M. Zemankova (1994), "Digital Library: Gross Structure and Requirements: Report from a March 1994 Workshop," In *Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries*.
- [Rumbaugh *et al.* 1991] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991), *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, New Jersey.
- [Rumbaugh *et al.* 1999] Rumbaugh, J., I. Jacobson, and G. Booch (1999), *The Unified Modeling Language Reference Manual*, Addison-Wesley.
- [Sharnowski *et al.* 1995] Sharnowski, J. L., G. C. Gannod, and B. H. C. Cheng (1995), "A Distributed Multimedia Environmental Information System," In *Proc. of IEEE Int. Conference on Multimedia and Computing Systems*, Washington, DC.
- [Wang and Cheng 1998a] Wang, E. Y. and B. H. C. Cheng (1998a), "Formalizing and Integrating the Functional Model into Object-Oriented Design," In *Proc. of International Conference on Software Engineering and Knowledge Engineering*, selected as Best Paper.
- [Wang and Cheng 1998b] Wang, E. Y. and B. H. C. Cheng (1998b), "A Rigorous Object-Oriented Design Process," In *Proc. of International Conference on Software Process*, Naperville, Illinois.
- [Wang and Cheng 2000] Wang, E. Y. and B. H. C. Cheng (2000), "Formalizing the Functional Model Within Object-Oriented Design," *International Journal of Software Engineering and Knowledge Engineering* 10, 1, 5–30.

- [Wang *et al.* 1997] Wang, E. Y., H. A. Richter, and B. H. C. Cheng (1997), "Formalizing and Integrating the Dynamic Model within OMT," In *Proc. of IEEE International Conference on Software Engineering (ICSE97)*, Boston, MA.
- [Wirfs-Brock *et al.* 1990] Wirfs-Brock, R., B. Wilkerson, and L. Wiener (1990), *Designing Object-Oriented Software*, Prentice Hall, Englewood, New Jersey.
- [Yourdon and Constantine 1978] Yourdon, E. N. and L. L. Constantine (1978), *Structured Design*, Yourdon Press.