An Adaptive Document Management System for Shared Multimedia Data[1]

T.M. Wittenburg and T.D.C. Little

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu

MCL Technical Report 05-02-1994

Abstract–We present a methodology for managing information resources by permitting aggregations of mixed-media documents to be interactively defined, shared, and managed using relational database management technology in concert with an extended object-oriented data model. This hybrid approach, based on our *polymorphic object model* (POM), allows the definition and instantiation of new types of compound documents without requiring modifications to be made to the underlying database schema. Existing documents and results from database queries are dynamically organized into individually meaningful categories as objects using a predicate-based mechanism. Furthermore, we developed a prototype document management system based on the POM. Analysis indicates that gains in flexibility are traded for slightly increased amounts of meta-data management overhead and document retrieval latencies when compared to conventional DBMS technology.

Keywords: Multimedia object and database models, data sharing, data correlation, heterogeneous distributed objects, document models.

[1]In *Proc. 1st IEEE Intl. Conf. on Multimedia Computing and Systems*, Boston, MA, May 1994, pp. 245-254. This work is supported in part by the National Science Foundation under Grant No. IRI-9211165.

1 Introduction

The shift toward client/server computing is contributing to the decentralization of the information management functions in many of today's globally distributed corporations. Similarly, as corporate, government, and academic computer networks become larger and more widespread, increased numbers of heterogeneous mixed-media information resources have become available. The convenient availability of increasing amounts of electronic information has created the need for a new generation of flexible information management tools. These new tools will be able to operate in a decentralized or globally distributed environment in such a way as to provide individuals within an organization the information necessary to both monitor corporate progress and to collaborate with counterparts. This new generation of information management tools will enable corporate information resources to be used in ways which effectively permit centralized management to be realized without requiring either a corresponding corporate or management structure to be established.

In an era of global competition, substantial economic rewards often belong to the enterprise which is able to produce quality products tailored to the needs of a target market in a shorter amount of time than its competitors. This governing dynamic has forced many forward-thinking enterprises to re-evaluate every aspect of their product production processes in order to increase overall efficiency. Part of this streamlining effort has included increasing the level of sharing of informational resources within the enterprise.

Because databases are often created at the project or department level, the information resources of an enterprise as a whole have rarely been conceived and developed as part of an enterprise-wide architecture. Consequently, enterprise-wide information resources can be viewed as a federation, implemented on different platforms and often with partially or completely incompatible database products. A capability to access individual databases and information resources within a federation is needed along with an ability to combine information from these separate sources into a single environment. In addition, there are many examples such as the photographic library of a newspaper publisher or an insurance company's document image library, where a significant information resource can exist in a form which is external to a corporate database. Such resources, often implemented using devices such as optical jukeboxes, appear to the information manager as part of a local or remote file system. An effective information manager must be capable of accommodating resources which exist both internal and external to legacy databases. A separate challenge arises once enterprise information resources become integrated. A means of automatically selecting relevant documents from the large available pool is required as well as a means of

organizing such documents in ways unique to each individual.

Our approach addresses these aforementioned requirements by permitting:

- The formation of composite objects with components internal and external to a database,
- automated selection and organization of documents into meaningful categories, and
- new data types to be created at run-time.

Because our approach provides the ability to create new data types during run-time, it is possible to use our approach to provide the effect of modifying an existing database schema without requiring a new schema definition or its related application to be recompiled or relinked. This action can occur without affecting the existing database, its schema, or the manner in which it functions.

The remainder of this paper is organized as follows: Section 2 describes the context of the research. Section 3 summarizes related work. Section 4 provides both an overview of the functional elements of the POM as well as a detailed description of the POM. Section 5 provides a summary of the prototype document manager. Section 6 concludes the paper.

2 The Context: Shared-Data Investigative Work

We focus on application areas in which multimedia data are shared as part of a collaborative investigative process. Examples of such application areas include: law enforcement investigation, medical diagnosis, environmental analysis, intelligence analysis, and exploratory seismology. In many respects the analytical processes carried out by investigators in these application areas resemble that of a detective solving a case. In each of these cases, the investigator often must assemble relevant information from current and historical sources. The assembled information can then be analyzed from several perspectives. The analysis process can involve the collaborative effort of a group of subject-matter experts. The investigator uses the observations of the group in conjunction with personal experiences to construct supportable hypotheses. After a thorough review process, the hypotheses and findings can be reported as a finished product. We observe from Webb [9] that the nature of this analysis process cannot be characterized in such a way that a single conventional database schema could be constructed which would be suitable for all types of analytical situations. Indeed

the investigator must frequently determine and refine multiple possible models of a given real-world situation as that situation develops.

A common thread among these applications is that the information structure to be modeled is either not clearly understood or frequently evolves. A challenge inherent in using traditional databases (e.g., relational) to manage the information needed to support these applications is that the structures of most databases created with current technology are resistant to change. More specifically, a conventional database schema is designed based on a well defined model of a given real-world scenario. When the scenario on which the model is based subsequently changes, the corresponding database schema may need to be modified to reflect the changes. Conventional technology requires that the database application be rebound (i.e., compiled and linked again) with the modified schema. Since most investigators are not adequately trained as database administrators, a method for managing multimedia data items is required which can accommodate dynamic structural adaptations.

The document manager must also be capable of dynamically organizing information into categories that are meaningful to each investigator. It is also desirable that information that has been registered with the document manager be indexed (or cross-referenced) in such a way that appropriate information items can be retrieved when a new or unforeseen development arises. This process may involve the definition of a new information category and new criteria for information to be assigned to the new category. An aspect of photographic images in this regard is that they are usually collected in order to satisfy a specific objective which has been determined beforehand. When the image is captured, it usually contains additional information which, although not directly applicable to the current objective, can be crucial to answering questions which arise long after the image was collected. An example is an X-Ray image originally taken to verify a rib fracture which unexpectedly reveals the initial stages of a pulmonary infection. This characteristic of sensor data (e.g., imagery, video, audio) places a premium on the ability to dynamically associate information about the contents of the sensor data (meta-information) with that sensor data so that it can be subsequently retrieved when new situations arise for which it is applicable.

We present a multimedia information management approach suitable for application areas where the information structure to be managed is either federated in nature or is subject to frequent structural changes. Both the Polymorphic Object Model (POM), and the prototype Reconfigurable Document Management System based on the POM (called *Rachel*) were developed during this research to address the need for a flexible document management system which satisfies the requirements described above.

3 Related Work

The need for information management methodologies suitable for modeling unstructured or rapidly evolving situations has been documented by a number of investigators including [3, 4, 5, 8]. Johnson et al. [5] describe a database of semantic networks which can be managed in applications where the organization of the data is unknown or is continually changing. Tsuda et al. [8] describe an object-oriented approach in which the structure of an information object can be transformed upon receipt of a "structural message." Gesmann et al. [3] describe a unique database system and query language designed to manage and access molecular models whose structures can be dynamically modified. Each of these approaches address in some way the need for information management systems which can be dynamically reconfigured to meet the demands of new situations.

Gory et al. [4] describe a Virtual Notebook System based on a hypermedia approach which organizes multimedia information into loose collections called notebooks. A significant degree of flexibility is provided in two areas: (1) Information items can be arbitrarily arranged on the pages of a notebook and (2) Navigational paths between pages can be arbitrarily complex. The system does not, however, provide the capability to create new aggregate information types. The greatest drawback of hypermedia-based approaches is that the links between pages must be maintained manually. For large notebooks, the process of modifying links can become quite time consuming, particularly in situations where the notebook must be frequently updated.

Approaches to information management based on workflow modeling have been successfully employed by Franklin [2] in more structured and repetitive processing environments such as insurance offices which process claims. The approach outlined by Franklin is not suitable for the applications considered in this paper since the specific sources of information which contribute to finished investigative reports can vary considerably. Furthermore, the process used to create such products cannot be codified since the flow of information accesses is a largely a function of the thought processes of the investigator (or team of investigators).

Kenworthy [6] and others [13] have developed multimedia information management systems for military purposes (e.g., battlefield information). Most of these systems have been designed to operate in highly specific environments and therefore lack the degree of flexibility required to address the applications considered in this paper.

We propose an approach which differs from those described above in several significant

ways: We present in this paper a prototype analytical tool which enables multimedia information to be dynamically organized and reorganized in softcopy so that the general process of understanding the significance of the data can be facilitated. The emphasis of this research is on providing an enhanced means for organizing data to enable the investigator to discover patterns and derive additional information from existing collections of data. Wurman [12] uses the metaphor of *hats* (i.e., units of information) and *hat racks* (i.e., frameworks in which to understand information) to describe this problem: Wurman describes "the visual hat racks (maps, diagrams, charts, lists, and time lines) that help us understand how our world is organized. Information hats can be hung on these racks so that patterns, connections and relationships formed from such adjacencies are revealed." Wurman further states that there are only six general ways to organize information: Alphabetic, Time, Location, Continuum or Magnitude, Category, and Hierarchy.

The general understanding process described metaphorically by Wurman resembles aspects of Webb's discussion [9] of the process of long-term intelligence analysis which involves juxtaposing information from disparate sources in order to discover smaller patterns which in turn support hypotheses for more significant assertions. Our approach is based on an extended object-oriented data model which is described in detail in the following section.

4 The Polymorphic Object Model

In this section we introduce the Polymorphic Object Model on which our prototype document manager is based. The POM itself is an extension of an object-oriented data model called O_2 [1, 7]. In order to provide a reference point for understanding the POM, Table 1 compares several key aspects of the POM with the traditional Relational Data Model (RDM). Table 1 indicates that the POM provides interactive definition at the tuple and collection levels of organization. These two features provide the POM's ability to accommodate dynamic creation of new data types.

4.1 Overview of the POM

The components of the POM are summarized graphically in Figure 1. In our model, information objects are organized into three general levels. The lowest level is called the *native* level which consists of a set of early-bound data classes which represent fundamental units of data. Examples include: integers, items of text, sensor data files, reports, or graphical

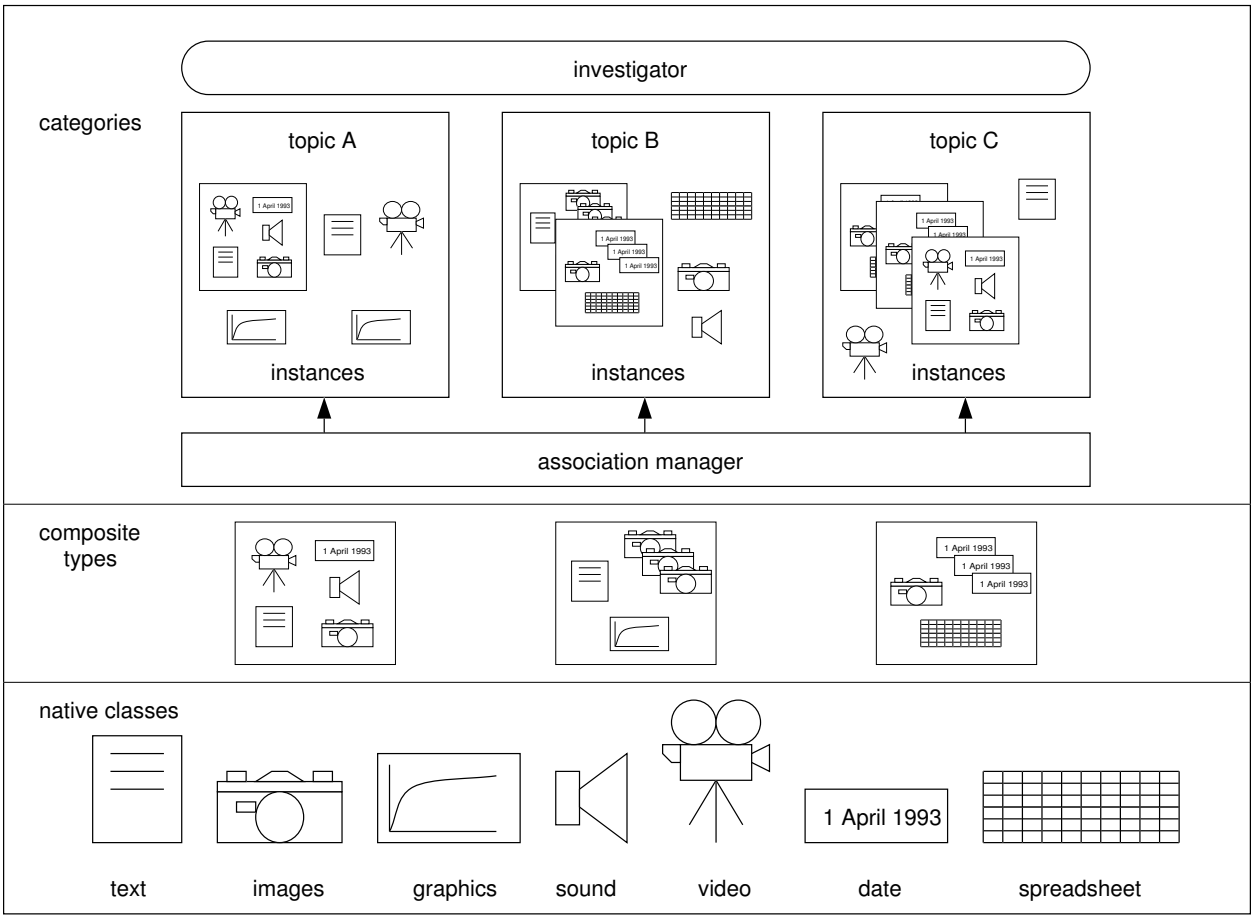

Figure 1: Summary of the Polymorphic Object Model

models. The native level is analogous to the set of primitives used in a graphics system in the sense that attributes of higher levels of organization in the POM are fundamentally determined by the properties of members of the native level. As we shall see, native classes are not restricted to consist of simple data entities but can represent fundamental data retrieval operations as well such as a database query. Instances of native classes are called native objects.

The second level consists of a set of *composite* types each of which can be interactively defined (during run-time) by specifying an ordered set of native classes. A composite data type is similar to an empty *knowledge frame* in the sense that a composite type represents a pattern of pre-defined slots (i.e., placeholders) in which information items can be subsequently placed. A composite object is an instance of a *composite type*. A composite object is a list of references to native objects. Each native object referenced is called a *component*

of the composite object. It is not required that all components in a composite object be defined in order for a composite object to exist. This condition permits composite objects to contain optional components.

At the third level, composite objects can be associated together by the interactive formation of subject matter categories which are of interest to the investigator. Criteria for object membership in a category can also be defined interactively for each category. These criteria are called *membership* predicates. Objects whose components satisfy the membership criteria of a given category can be assigned to that category by an assignment operator defined over the set of categories and composite objects. The capability to associate composite objects with applicable categories represents an ability to form dynamic links between information objects. When coupled with a display function, this capability is analogous to automatically creating a set of cards (a card displays an individual composite object) *and* the links for navigating between cards. Categories can also contain categories.

In environments where new data objects are continually entering the processing environment [10], a document manager based on the POM can repeatedly apply the membership predicates of each category to incoming data objects upon their arrival. The investigator can thereby automatically gain access to the most current information which is specifically relevant to the current needs.

Table 1: Comparison of the Relational Model to the Polymorphic Object Model

Feature	**RDM**	**POM**
Lowest Level of Organization	Data Element Dictionary	Set of Native Classes
Tuple Level Organization	Record	Composite Object Type
Method of Defining a Tuple	DDL Specification (Fixed)	Composite Type Definition (Dynamic)
Collection Level of Organization	Relation (Set of Records)	Category (Set of Objects)
Method of Defining a Collection	Manual Record Entry	Membership Predicates (Automatic Assignment)
Retrieval Method	SQL Queries	Predicates
Inter-Collection Organization	Relations on Keys (1:1, 1:Many, Many:Many)	Many:Many Links Formed
Viewing Method	Fixed Format Forms Generation	Multi-Format Forms Generation
Data Manipulation Language (DML)	4GL	Currently Consists of Search and View Functions

4.1.1 Differences between O_2 and the POM

The Polymorphic Object Model described in this section is a variation of the O_2 object-oriented data model developed by Lecluse, Richard, and Velez [7, 1]. In this section we describe similarities and differences between O_2 and our proposed model.

The first major difference between O_2 and POM is in the implementation method. Lecluse defines three types of objects, *basic objects*, *tuple-structured objects*, and *set-structured objects* and indicates that objects defined in O_2 are bound at run-time. In contrast, the POM defines two kinds of objects: **Native**, and **Composite**. Native objects incorporate the features of both O_2 basic objects and tuple-structured objects, and are bound at run-time.

Our POM defines a set of composite objects which closely resemble the O_2 *set-structured* object which is a list of pointers to other objects. The major difference between these models is that the POM defines composite types and their instances not at run-time (as in O_2) but during run-time, that is, both composite type definitions and their instances are unbound.

Another departure from O_2 is in the area of composite functions. The POM defines functions on composite objects which are compositions of the functions of the composite object's native members. In addition, the POM defines a categorization operation not found in O_2 which permits both native and composite objects to be clustered together in different ways based on a set of dynamically defined predicates associated with each category.

Finally, O_2 does not permit members of any object to be named. The POM introduces notation which permits an object or object component to be designated by either a name or, in the case of composite objects, an ordinal value indicating the objects position in an ordered set. In the following we provide definitions for each component of the POM.

In the remainder of this section we describe the elements of the POM in greater detail.

4.2 Native Classes

Native classes represent the basic kinds of data such as text, number, date, time, image, and sound. Native classes must contain in their definition, all methods necessary to implement the Data Manipulation Language (DML). In this case the DML contains functions for locating and viewing native objects. Native objects can be stored internally either in a local database management system (DBMS) or externally using a locator (described later). An object-oriented database can be used to provide persistence for non-record-oriented native objects.

Eventually native classes can also be defined which are, in fact, queries to existing (legacy) databases. A requirement for such a native class is that the query result also be a member of the set of native classes. This requirement ensures that the display method for any composite object containing a database query as a component is a well-defined function. These are defined formally as follows:

Definition 1 *The set of native classes* $\mathcal{N}$ *is defined as* $\mathcal{N} = \{N_1, N_2, ...N_\sigma\}$ *where* σ *is the cardinality of* $\mathcal{N}$ *(i.e.,* $\sigma = |\mathcal{N}|$*). Each* $N \in \mathcal{N}$ *is a* **native class** *and consists of the ordered pair* $N = (D, F)$*, where* $D \in \mathcal{D}$ *is an ordered set of data members and* $F \in \mathcal{F}$ *is an ordered set of native functions:*

$$D = \{d_1, d_2, ...d_j\}, j = |D|$$

$$F = \{f_1, f_2, ...f_k\}, k = |F|.$$

Given several sets of native functions (F's), for every function, $f \in F$, there must also exist a matching function name which performs the same function in every other F defined for every $N \in \mathcal{N}$. In other words, all defined native classes must share a set of polymorphic functions. More specifically, if f_1 is a display function, then every $N \in \mathcal{N}$, must have an associated display function. This condition permits composition functions to be constructed as described in Theorem 1.

Definition 2 *By Definition 1, each* $N_i \in \mathcal{N}$ *contains a set of data members* D_i*. The instance* t *of each* $d \in D_i$ *is called a* **native object** *and consists of an ordered pair* $t = (v, l)$ *where* v *is the value and* l *is a unique identity.*

4.3 Composite Types

Composite types are defined interactively during run-time as combinations of members of the set of native classes. Since composite types are in effect unbound, they are called *types* instead of *classes* in order to distinguish them from the early-bound *native classes*. An example of a composite type is an object formed by the addition of meta-information to a scanned image to permit the image to subsequently be sorted and categorized. In this example the investigator defines a composite type MyImage which has the following components:

- Creation Date, which is an occurrence of the native class *Date*,
- Creation Time, which is an occurrence of the native class *Time*,
- Content Summary which is an occurrence of the native class *Text*.

Composite types provide the investigator the needed capability to create new abstractions which are unique to the scenario at hand. These are defined formally as follows:

Definition 3 *A set $\mathcal{C}$ of* **composite types** *is defined as $\mathcal{C} = \{C_1, C_2, ...C_\gamma\}$ where $\gamma = |\mathcal{C}|$. Each $C \in \mathcal{C}$ is itself an ordered set of references to either native classes or composite types. That is, for each $C \in \mathcal{C}$, $C = \{r_1, r_2, ...r_\mu\}$, where $\mu = |C|$. Every $r \in C$ is called a component and is an ordered pair (C', R) where C' is a component name and $R \in \{\mathcal{N}, \mathcal{C}\}$.*

Because each $C \in \mathcal{C}$ may contain a component which is a member of $\mathcal{C}$, this definition is recursive. Also, because nesting of composite types is permitted and N can refer either a composite type or a native class, each nested type must decompose to d's which are members of $\mathcal{N}$ in order to satisfy the definition. Each $C \in \mathcal{C}$ is defined during run-time and is implemented as an unbound object.

Definition 4 *A* **composite object** *is defined as an ordered set of instances of the native classes contained in the corresponding composite type definition.*

The ordering of the composite object components forms a bijection with the composite type to which the object belongs. Component names within a composite type must be unique, however, because each $C \in \mathcal{C}$ is an ordered set, component names are not necessary to specify a unique $r \in C$ since the ordinal digit representing the position of $r \in C$ is sufficient for its identification.

Theorem 1 *A composite function can be applied to any composite object $C \in \mathcal{C}$.*

For example, a composite function F_1 applied to any $C \in \mathcal{C}$ is in fact a composition of the polymorphic member functions f_{1_q} of each $r \in C$. This is true because each $r \in C$ is an ordered pair containing an $R \in \{\mathcal{N}, \mathcal{C}\}$ and each R ultimately has a set of member functions. If $C = \{r_1, r_2, ...r_\mu\}$ and each r contains a (C', R) through recursion each R refers to a native class which consists of an ordered pair (D, F). Consequently, $F_1 = f_{1_q} \circ f_{1_{q+1}} \circ ... \circ f_{1_{q+n}}$ where n equals the nondegenerate number of native classes referenced in the recursive definition of C.

4.4 Association Manager

The association manager associates composite objects based on common features that are specified interactively using a set of membership predicates from each category definition. The association manager provides the following capabilities:

- Dynamic specification of the nature of objects to be assigned to individual categories (i.e., decides what kind of objects can belong to each category).
- Management of a category hierarchy (i.e., permits a category to belong to other categories).
- Dynamic assignment of accessible data objects to appropriate categories.

In a law enforcement context, for example, the investigator can use the association manager to automatically assign new data objects to the appropriate cases and then browse through located documents on a case-by-case basis.

Definition 5 *A set of categories $\mathcal{C}^c$ is defined over the sets of native and composite objects as $\mathcal{C}^c = \{C_1^c, C_2^c, ... C_u^c\}$, where $u = |\mathcal{C}^c|$. Each* **category** *C_1^c is defined by a set of predicates. Each predicate is an ordered set of Horn clauses of the form:*

COMPONENT.OPERATOR.VALUE.[CONJUNCTION];

A category is defined by a name and a set of predicates: $C^c = (Name, P)$ where $P = \{p_1, p_2, ...p_v\}$ and $v = |P|$.

A predicate p consists of:

- Operands that are constants or composite object component identifiers.
- Arithmetic comparison operators ($<, =, >$).
- Native class dependent operators such as "contains" for $N =$ "Text."
- The structural operator "Exists" which evaluates to true iff the component has been assigned a persistent value.

- Literal constants.
- Logical conjunctions which, where necessary, separate each predicate into a series of Horn clauses.

Definition 6 *An object is* **assigned** *to a category B if its attributes match a condition specified in one or more predicates in B. Assignment is denoted by the operation $\alpha_p(\mathcal{C})$ on composite objects $C \in \mathcal{C}$. I.e., assignment occurs when each Horn clause in predicate p evaluates to true.*

A composite object $C \in \mathcal{C}$ can also be directly assigned to a category by an edict $\alpha_e(\mathcal{C})$. Thus, membership in a category $C^c \in \mathcal{C}^c$ is granted iff an $\alpha_p(\mathcal{C})$ exists or an edict $\alpha_e(\mathcal{C})$ exists:

$$M(C^c, C) = \alpha_p(\mathcal{C}) \vee \alpha_e(\mathcal{C})$$

4.5 Event Filters

Considering that dates, times, and earth geo-coordinates can exist as either native or composite objects, categories can be defined which actually represent potential spatial or temporal events of interest to the investigator. Such categories can be more accurately called *events*. Events can be defined and detected using filters constructed from predicates of the same form used to define associations between objects and categories. The event filters actually permit the investigator to monitor or detect pre-defined events. Once a filter has detected an event it could send an electronic-mail message to a pre-assigned list of investigators. The message would indicate that data objects have been received which satisfy the criteria for a designated event and request that the appropriate categories be browsed by investigators.

4.6 Component Locators

In order to satisfy the need for distributing a composite object's components over several locations, a location specification called a *locator* can be provided for each component of the document as part of the object instantiation process. The locator permits each component of a composite object to be retrieved from an appropriate source.

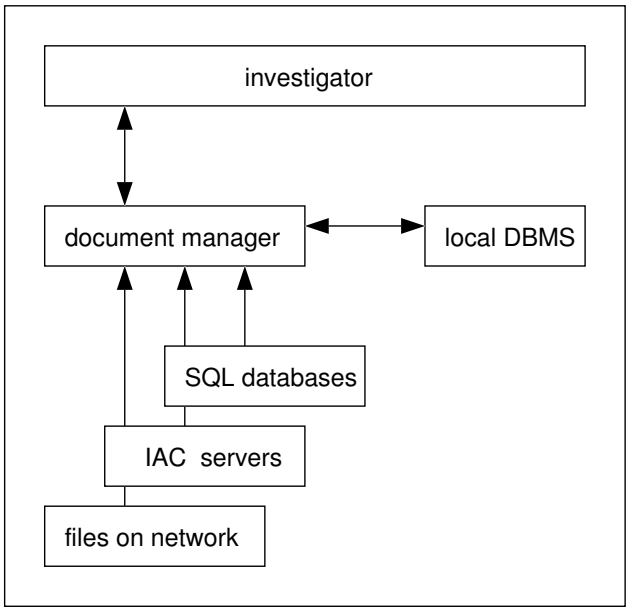

Figure 2: Provisions for Local and Remote Information Access

The methodology for accessing object components which exist in known formats is to define an appropriate native class with the method necessary to accept the desired format. An attractive alternative to this approach is to utilize an appropriate inter-application communication (IAC) mechanism (e.g., Dynamic Data Exchange (DDE) in the PC environment or ToolTalk in the Sun/Unix environment). This alternative approach would provide a native class which implements a client capability for the appropriate IAC mechanism. A POM-based document manager's access options are illustrated in Figure 2 and include the following external information sources:

- The document manager's local database management system.
- SQL accessible databases.
- Local or remotely networked files.
- Documents accessible via an inter-application communication (IAC) mechanism.

5 A POM-Based Document Manager (Rachel)

In order to demonstrate the practical advantages of the POM, we have developed a prototype document management system called Rachel.

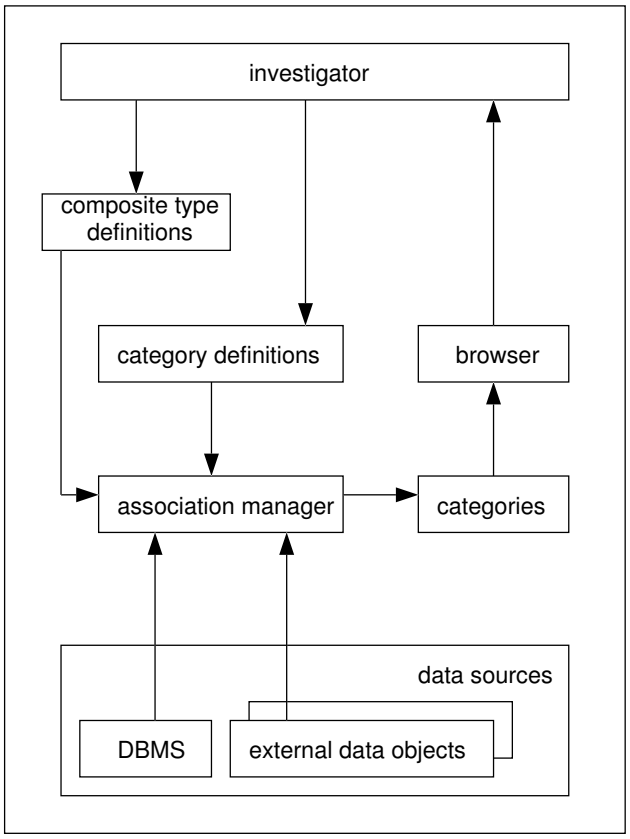

Figure 3: Components of the Rachel Document Management System

Figure 3 illustrates the functional components of the Rachel prototype document manager as well as the general relationships between the subsystems. Details of the Rachel architecture and software environment are not described here (see reference [11]).

5.1 Dynamic Type Creation

The following native classes are provided by the prototype: Short Text, Article, Integer, BMP (bit-mapped graphic), Image, and Date. An interactive GUI is provided which permits the investigator to define composite types which are tailored to current needs. Objects can then be interactively created and appear as entries in the object catalog.

5.2 Predicate Formation

Category names can be interactively defined as well as predicates for each defined category. Predicates are created interactively using a Query by Example (QBE) screen in which all elements of a predicate (except literal values) can be selected from the GUI using a mouse. The predicate creation screen also serves as an interactive query processor. Predicates can be created independent of a category and immediately applied to the entire collection of objects. Search results appear as a list of objects. Each object selected by the predicate can be viewed individually. This feature is useful for testing the validity or resolving power of individual predicates. Once a predicate has been tested in this way it can be added as a membership predicate to any existing category. At this point the investigator can invoke the association manager which assigns all accessible objects to the appropriate categories. The result of the assignment process is a set of category-based information packages which can be browsed. Individual composite objects can be viewed by selecting a category and then clicking on the member object's name or identity.

5.3 Browsing

Once documents have been assigned to categories, they can be viewed interactively by a browsing mechanism which is briefly described here. Two browsing or viewing styles are provided on a per-category basis in the Rachel prototype. The first style is *sequential* in which the objects are displayed in serial order. In the case of a nested composite object (i.e., a composite type which contains composite types), the corresponding object is displayed at the highest level first. Lower level composite objects are displayed as named icons at the parent object level. The lower level composite object is viewed by "double-clicking" its icon. If an individual component is anything other than a text or numeric native class (e.g., image, long text, sound file) that component first appears as a named button. The component itself is viewed by pressing the button which calls the appropriate display method (which was defined as part of the native class) which opens a window to display the appropriate object. These "native" windows are tailored to the characteristics of the native object itself.

The second viewing style is *tabular* in which all objects of the same type within a category are grouped into a tabular format. One table is created for each group of objects of the same type. In a manner similar to that described for the sequential viewing style, non-text components such as images and sounds are iconified table entries. The iconified objects are viewed by "double-clicking" the icon. The tabular viewing style provides a means of creating

traditional tabular reports.

5.4 System Experience

Our prototype implements all of the basic aspects of the POM described previously. We performed tests of Rachel by interactively creating and instantiating a number of composite types. These included the type **Country** with components such as raster maps (accessed from the local file system using a locator) and selected country statistics stored in Rachel's local database. A set of **Country** objects were then populated in the Rachel database. We created categories which selected **Country** objects with, for example, large populations (large defined by a predicate) or higher than average rainfall. Our experiences generally indicate that this approach is viable for the situations for which it was designed. More rigorous testing is required, however, in order to characterize Rachel's performance on very large datasets and distributed-data environments. To this end, we are seeking deployment on a broader realm of data as offered by the information resources available on the Internet.

6 Conclusion

An approach to multimedia document management has been presented which addresses the need to (1) accommodate a high degree of structural flexibility in the data model, (2) provide an automated method for locating and aggregating information items applicable to the needs of individual investigators, and (3) provide an increase in the effectiveness of collaborative investigative processes through reusable object categories and data sharing.

POM-based approaches offer several advantages in applications where the structure of the information model is unknown or changes frequently. In scenarios where the situation being modeled has a well known structure which does not change frequently, POM-based approaches do not appear to offer an advantage over conventional database approaches. The high degree of flexibility provided by this approach should be accompanied by a policy that provides guidelines for the creation of new types and categories in ways which facilitate the long-term efficiency of the document management system.

We are investigating several extensions to this work:

1. In the area of additional native classes, the feasibility of adding IAC links to word processor documents or spreadsheets is being investigated in order to access documents

in existing formats without requiring the document manager to have specific knowledge of the formats themselves.

2. The performance of the category assignment operator over a network is uncharacterized at this time. The development of a database server which is capable of (a) receiving individual packets each containing requests for multiple database items and (b) returning multiple items retrieved from the database would be an important step in determining the viability of this approach for very-large document collections.

3. The optimal amount of storage overhead required to maintain the control lists should be characterized with attendant tradeoffs. In the case of very large document collections, the amount of control information can grow large enough to justify its storage on a separate server.

6.1 Acknowledgements

The authors acknowledge Ed Green and the support of the MITRE Corporation. The authors also appreciated helpful discussions with Arnon Rosenthal.

References

[1] Altair Consortium, "The O_2 Object-Oriented Database Management System," *Comm. of the ACM*, October 1991, Vol. 34, No. 10, pp. 35-48.

[2] D.P. Franklin and G.M. Hoek, "PC LAN Document Imaging at Work in Insurance," *Advanced Imaging*, June 1992, pp. 18-20.

[3] M. Gesmann, A. Grasnickel, T. Harder, C. Hubel, W. Kafer, B. Mitschange, and H. Schoning, "PRIMA – A Database System Supporting Dynamically Defined Composite Objects," *Proc. ACM SIGMOD Intl. Conf. on the Management of Data*, June 1992, San Diego, CA, Vol. 21, Issue 2.

[4] G. A. Gory, K.B. Long, C.P. Jung, and B.D. Meyer, "The Virtual Notebook System: An Architecture for Collaborative Work," *Journal of Organizational Computing*, 1991, Vol. 1, No. 3, pp. 233-250.

[5] R. Johnson, M. Goldner, M. Lee, K. McKay, R. Schectman, and J. Woodruff, "Unstructured Scientific Data (USD) – a Database Management System for Scientific Research,"

Proc. ACM SIGMOD Intl. Conf. on the Management of Data, June 1992, San Diego, CA, Vol. 21, Issue 2.

[6] T. Kenworthy, "Information Management for Military Applications," *Advanced Imaging*, June 1992, pp. 40-50.

[7] C. Lecluse, P. Richard, and F. Velez, "O_2, an Object-Oriented Data Model," *Proc. ACM SIGMOD Conference*, Chicago, 1988, pp. 424-433.

[8] K. Tsuda, K. Yamamoto, M. Hirakawa, M. Tanaka, and T. Ichikawa, "MORE: An Object-Oriented Data Model with a Facility for Changing Object Structures," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 3, No. 4, December 1991.

[9] D.Q. Webb, *Computer-Aided Tools for Analysis of Science and Technology (CATALYST)*, Office of Scientific and Weapons Research, Science and Technology Division, U.S. Government, October 1989.

[10] E. Williams, and T. Myers, "Storage Estimates for Scenarios within the Intelligence Community," *Intl. Proc. of Supercomputing Science*, Monterey, CA, 1991.

[11] T.M. Wittenburg, and T.D.C. Little, "A Polymorphic Data Management System for Multimedia Database Applications," Tech. Rept. 12-29-1992, Multimedia Communication Laboratory, Boston University, 1992.

[12] R.S. Wurman, "Hats: A Metaphor for Units of Information, *Design Quarterly*, Issue 145, MIT Press, Cambridge, MA.

[13] S. Zenlea, "Constant Source," MITRE Technical Report #27559, 1987.