The Alexandria Digital Library Project

Terence R. Smith James Frew Greg Janee Linda Hill

University of California at Santa Barbara, Santa Barbara, California 93106, USA
smithtr@cs.ucsb.edu

1 Introduction

The Alexandria Digital Library (ADL) Project[1] has developed digital library services and collections to support the organization, access, and use of collections of information with the use of geospatial reference [1]. The Project's operational component is the Alexandria Digital Library (ADL), developed under Phase I of NSF's First Digital Libraries Initiative and now operational as part of the California Digital Library (CDL). The Project's research and development component is the Alexandria Digital Earth Prototype (ADEPT) [2], supported under NSF's Second Digital Libraries Initiative.

The services of ADL and ADEPT provide Internet access to distributed collections of information that is explicitly or implicitly georeferenced. Georeferencing implies that any item in an ADL/ADEPT collection is associated with one or more regions (footprints) on the surface of the Earth. Footprints may be represented in terms of the latitude and longitude values associated with points on the boundaries of multi-polygonal areas, including the degenerate case of point polygons. Together, ADL and ADEPT may be viewed as an extensible set of services that support (1) geo-spatial search for, and access to, a full spectrum of spatially-indexed materials in multiple, heterogeneous collections; (2) the growth and creation of collections of geospatially referenced information; and (3) the use of such information in personalized and collaborative settings.

ADL's current (and growing) collections include a 4.8M-entry gazetteer, 2.4M georeferenced metadata records, and over 2TB of maps, images, and air photos. ADEPT provides access to these collections, as well as to a set of smaller collections that are being constructed to support educational activities at the undergraduate level. In this paper, we focus on the services and collections of ADL, since it is the operational component and provides a basis for the developments of ADEPT, which we briefly summarize.

2 Access and Meta-information

ADL is distributed. Its components may be spread across the Internet, as well as coexisting on a single desktop. An ADL installation may contain multiple collections, each with its own unique metadata schema. An ADL installation may also support multiple clients; for example, graphical interfaces, analysis environments, and gateways to other library systems. These characteristics imply the need for a robust solution to the problem of meta-information to support discovery and search. The goal of facilitating user access by providing services for issuing a single query against multiple heterogeneous collections requires that all collections available to a user support searchable metadata that can be reduced to a common, core form. In achieving this goal, we ADL provides mechanisms by which both the objects in a collection and whole collections can be characterized by common, core forms of metadata. We first focus on such forms for individual objects.

[1] The work described herein was partially supported by the NSF-DARPA-NASA Digital Libraries Initiative, under cooperative agreements NSF IR94-11330 and NSF IIS-9817432.

2.1 Basic Meta-Information

Meta-information in the current ADL catalog [3] is represented in terms of an essentially flat schema describing only those metadata fields used for search. The remaining metadata is described and pre-packaged into more flexible XML reports. The schema has approximately 400 fields, all viewed as fields from MARC 21, amplified by an expansion of MARC 21's "General Notes" field (field 500) into uniquely numbered and named note fields. An earlier meta-information model was based on the USMARC (MAchine-Readable Cataloging; now MARC 21) and FGDC (U.S. Federal Geographic Data Committee) standards. Each information object was represented by a record of approximately 400 fields in a relational database involving a hierarchical schema. Implementations of the schema, however, were characterized by poor performance, particularly in the building of records from large numbers of tables.

Two issues arise with such a metadata schema in attempts to achieve ADL's goal of making search over multiple collections easy for a user. First, the use of many (hundreds) of attributes poses its own problems: the user must be knowledgeable about the metadata schema and attributes existing for particular types of information and searches of any particular attribute may miss items that are relevant but lacking data in that attribute. Second, in a multi-collection digital library system, where the underlying metadata for each collection may be based on dramatically different standards and schema (e.g., USMARC, Dublin Core, GILS, ADL Gazetteer Content Standard, FGDC Content Standard), searching object-level metadata attributes directly is unworkable. ADL, for example, has two collections with different schema (its catalog collection and its gazetteer collection), while ADEPT, which is designed to be interoperable with ADL, supports its own schema with fields designed for educational purposes. Hence it is important to find ways of making different schemas simple to a user and compatible in terms of search middleware.

2.2 Search Buckets and High-Level Metadata Abstractions

One approach to this problem is to use generic search metadata, which may be viewed as a classification of object-level metadata attributes. This does not, however, solve the problems of searching over diverse and often complex metadata models. First, while users need not know the structure of the underlying metadata, a search against generic parameters is likely to access large numbers of objects in any collection. Second, it is important to have available controlled vocabularies to describe collection object attributes. Ideally, the controlled vocabularies associated with a generic search parameter will be the same as those used to catalog the underlying collection. However, one cannot assume that a 1:1 association can be maintained across multiple collections.

ADL has adopted an alternate strategy using *search buckets* [3], which comprise a small set of high-level metadata attributes intended primarily for querying (as opposed to describing) and as high-level containers for metadata from multiple underlying sources. All holdings accessible through ADL are queryable according to a set of search buckets that include: (1) *geographic locations*: in terms of the set of points, (bounding boxes, polygons, etc.) in latitude-longitude coordinates, specifying the holding's footprint; (2) *dates*: or date ranges associated with the content or preparation of the holding; (3) *types:* drawn from collection-specific domains which characterize the form (e.g., map, aerial photograph, etc.)] or content (e.g., hydrographic feature, airport, etc.) of the collection's holdings; (4) *formats*: or representations (online or offline) in which the holding can be delivered; (5) *assigned terms*: from the topics and themes (e.g., subject heading, index terms, keywords, etc.) of the holding, drawn from specified controlled vocabularies; (6) *subject-related text*: from the topics and themes of the holding (e.g. title, abstract, assigned terms, etc.) (7) *originators:* or words from author, investigator, publisher, and similar attributes;(8) *identifiers:* such as ISSN, ISBN, report number, URL. This set is extensible, but is

designed to be small. ADL does not require that collections populate every search bucket. They were chosen, however, with the expectation that they would be well-populated by most collections.

As abstract, high-level, structured metadata fields, ADL buckets differ from other metadata schemes such as content standards (e.g., FGDC) and other high-level definitions (e.g., Dublin Core) in two notable ways. First, buckets are a metadata abstraction system providing a formal means by which semantically similar metadata fields may be grouped to provide higher-level search and description capabilities. Objects populate buckets with both values and source metadata mappings. Collections summarize and abstract the metadata mappings, and allow clients to search by either entire buckets or by specific metadata fields that have been mapped to the buckets. Second, buckets are strongly typed, being either spatial, temporal, hierarchical, textual, or numeric. Associated with each bucket type is a standard representation (e.g., polygons and boxes for the Geographic Location bucket) and a set of standard operators (e.g., set algebra). The combination of an abstraction system with strong typing yields a metadata model that is flexible enough to accommodate a wide range of metadata, yet expressive enough to support powerful, targeted queries.

The combination of an abstraction system with strong typing yields a metadata model that is flexible enough to accommodate a wide range of metadata, yet expressive enough to support powerful, targeted queries. Each ADL collection must provide mappings between the search bucket attributes and its own specific metadata. Facilitating this mapping (and thus encouraging the growth of ADL) has been the prime instigation for keeping the buckets few and simple. Conversely, the buckets are the only searchable attributes that an ADL user can be guaranteed will address the entire universe of ADL holdings. Facilitating expressive queries has been the prime instigation for giving the buckets fairly precise semantics (more so than similar high-level metadata schemes such as the Dublin Core.)

2.3 Collection Meta-information

Since search in distributed digital environments can occur over many complex collections, it is important to characterize the nature of collections as well as individual items. In particular, such information can be used for: (1) collection "registration" with the search and retrieval and client software that will provide access to it; (2) network discovery, providing information to network search agents about what the collection contains; (3) user documentation or information about the collection and the digital library interface to it; (4) management of the collection to provide a focal point where information is stored (or referenced) pertaining to the collection.

A key function of collection metadata is to declare the metadata structures used to describe the objects within it, with reference to complete documentation for those structure and to specific versions of the structures. For each ADL collection, the mappings from the object metadata attributes to the search buckets, scan elements, and full metadata display are identified through the collection metadata. These constructs support the following functions: (1) search buckets provide common search parameters across various collections with a small number of top-level buckets; (2) scan elements are the attributes that will be displayed in the brief one-line display of items in result sets, and currently consist of title, date, type category, type of available format, and the short name of the collection; (3) the full metadata display provides a labeled, and perhaps standardized, presentation of an object's complete metadata. Collection metadata also references the controlled vocabulary lists for domain values, thesauri, subject heading lists, classification schedules, etc. that are used to represent the values for particular object metadata attributes. These references facilitate the use of the vocabulary tools in the user interface.

We have developed a collection metadata schema [6] that contains elements of both the databank service user-documentation sheets and the STARTS collection metadata. ADL collection metadata is designed to

contain all of the metadata needed by the middleware to add a collection to the ADL system, including the generation of user documentation. It is also a collection management tool in that it gathers together in one place all of the pertinent information about that collection. It is important to note that the portion of the collection metadata that describes the particular search capabilities for the collection will be specific to a particular user interface and set of search programs. If, for example, a particular collection is made available through different interfaces, the collection information would remain the same but the search options would change.

Attributes of ADL collection metadata are described using three attribute characteristics: (1) *type* (text, text and image, text and link, integer, relation, compound) defines the type of data to be entered for the attribute;(2) *domain* (free text, email text, phone text, List reference, numeric range, URL, map, tree, date) of an attribute can be defined as "freetext" or as a specific type of text that can be computer checked to some degree for quality control; and (3) *source of value*. We may distinguish two classes of collection metadata: *inherent metadata* derived through the automated analysis of the contents of a collection; *contextual metadata* is supplied by the collection provider or maintainer.

An XML DTD for ADL collection metadata has been defined that dynamically provides the middleware, and through it, the user interface client, with a subset of the attributes of the collection metadata. For ADL's internal collections, the creation and management of collection metadata is automated to the fullest extent possible. Inherent metadata is created by two sets of scripts and programs that are run periodically. The first set computes or extracts the required data from the ADL collections and processes it into simple, raw forms. The second set converts this raw data into any of several external forms. Contextual metadata is created as a set of text files under the control of a configuration management system. A set of scripts combines these files into any of several collection metadata versions. Currently, an HTML version with inline GIF images is created for user documentation and an XML version is created for collection registration with the user interface according to a template (the Document Type Declaration or DTD.)

As a tool for collection management, collection metadata also provides a central focus for collection documentation. The contextual information included, such as the mapping tables from underlying metadata to the search buckets and the full metadata display, can be easily consulted when linked to the collection metadata. Attributes useful only for internal management, such as notes about the acquisition and processing of the metadata and data, can also be added to the schema. These local-management attributes would not be displayed in the user documentation but would be accessible to the management staff.

3 Gazetteer Services

Gazetteer services play a critical role in geospatial access to information since it is typically difficult for users to specify a region of interest using sets of latitude-longitude coordinates. A gazetteer is list of geographic addresses of names, together with their geographic locations, or footprints, and other descriptive information. Footprints can be attached to any type of information about a geographic location, including maps, aerial photographs, and remote sensing images as well as text items, museum objects, specimens, data sets, and even music. Hence gazetteer services provide important support for geospatial search and querying by allowing user to answer “where” question such as “Where is Santa Barbara?” to translate between geographic names and locations; and to locate particular types of geographic features in a designated area.

The ADL gazetteer services [5] provide access to an online gazetteer of approximately 4.8M million geographic names, categorized by a hybrid of feature classes and types, that integrates the GNPS (non-U.S. names) and the GNIS (U.S. names) gazetteers, which are both accordant with the guidelines of the

Board on Geographic Names (BGN). Currently all names are georeferenced by point locations, since there are no large sets of gazetteer data with more general representations of spatial extent. In constructing a gazetteer and integrating its services into ADL, one must resolve issues concerning: *content standards*, generalized *spatial extents*, *feature types, temporal aspects*, and *quality aspects*, as well as issues concerning *multilingual gazetteers*.

In developing the Alexandria Gazetteer, we have resolved some, but not all, of these issues. We have developed a Gazetteer Content Standard (GCS) and a Feature Type Thesaurus (FTT). The GCS is designed to support contribution and sharing of gazetteer information. ADL has developed a relational database model for the GCS and is populating the database with sets of gazetteer data. The GCS specifies required and repeatable attributes. It supports contribution from multiple sources by (1) having metadata entries for each geographic name; (2) recording the contributor, source, and entry date for each attribute-value component of the entry; and (3) representing the relationships between geographic names through links between entries rather than by a thesaurus structure. The Feature Type Thesaurus (FTT) was built as a hierarchical scheme in accordance with ANSI/NISO Standard Z3919-1993. A large variety of sources were consulted in building the thesaurus. In populating the gazetteer, it was necessary to address a large number of feature type and other conversion issues including, for example, mapping incoming entries to the GCS relational database and incoming category terminology to the FTT; solving specificity and generality mismatch issues; developing definitions for categories to support conversion mappings; resolving cases in which sources of gazetteer information do not explicitly categorize their geographic names, as well as resolving issues specific to the two source gazetteers.

4 Collections

ADL's holdings are focused on collections of geographically-referenced materials and currently include over 2 TB of digitized maps, satellite images, aerial photographs, and specialized textual material. The two other collections currently supported by ADL include its catalog of over 2.4M items, and the gazetteer containing over 4.8M items. The architecture, described in the next section, is designed to be extensible with respect to new collections. The existing collections are growing on a daily basis with the application of well-established procedures for meta-information extraction and cataloging.

5 Architecture

The ADL architecture employs the three-tier model, shown in Figure 1 in which servers manage library collections, middleware implements standard services on these collections, and clients present these services to library users.

ADL servers are responsible for maintaining collections of metadata describing the library's holdings, and for implementing query and retrieval mechanisms against that metadata. An ADL holding is an object cataloged by ADL, but not necessarily “owned”" by ADL. A holdings metadata may include references to online representations of the holding (e.g. URLs), but this is not required since ADL can also be used as an online catalog for offline or non-digital geospatial information. As such, ADL servers are generalizations of traditional library catalogs. As long as they expose the proper interfaces to the middleware, ADL servers are otherwise autonomous. Hence a site can implement an ADL server to “publish”" specific collections that are accessed only through another site's middleware.

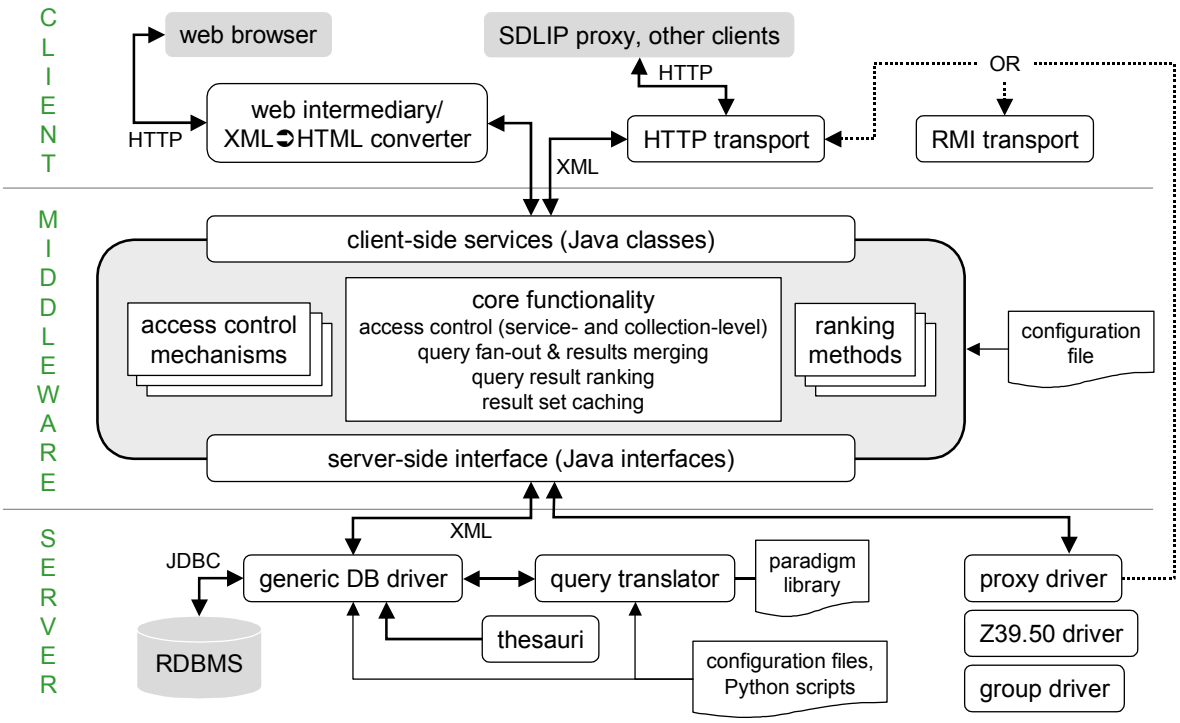

Figure 1 ADL Architecture

The core of the ADL architecture is the middleware layer, which maps an assortment of heterogeneous collection servers into a few standard client interfaces for metadata queries, metadata retrieval, and digital holding retrieval. These client interfaces are intended to be generic enough to support arbitrary clients. While stateless middleware is of value for attaining scalability and simplicity, the combination of stateless middleware and stateless clients severely limits the library's overall functionality. The ADL middleware architecture now implements, therefore, result set caching and other features to allow stateless clients to perform iterative queries.

The middleware's client interfaces are the primary public artifact of the ADL architecture, defining and circumscribing the library's ADL clients are responsible for presenting ADL services to library users. These users may be interactive (e.g. humans using a GUI), or they may be other programs using ADL as a data source. ADL clients are assumed to be capable of maintaining enough local state to support whatever notion of a session (global context) that they require. ADL clients may also support complex real-time user interactions (e.g. rollover help). All externally-visible data streams in the ADL architecture use XML-based encodings. All XML elements carry source attributes that reference the elements' semantics. XML DTDs have been defined for queries and metadata reports (replacing arbitrary HTML). Using validated XML instead of idiosyncratic formats makes it easier for other systems to connect to ADL, and permits richer client functionality.

5.1 Client Interfaces

The ADL architecture defines five standard client interfaces. A sixth *mapbrowser* interface allows a client to request a base map for a particular portion of the Earth's surface, specifying a projection and feature

layers (coastlines, highways, etc.) and simplifies the development of graphical clients that display maps as navigational tools.

The collections interface: allows the client to determine which library collections (logical groupings of library holdings) are accessible from the middleware. This interface exposes two methods: *list,* which returns a set identifiers for all the collections that the middleware has access to; and *scan*, which returns detailed metadata for the collection. These metadata are specified by the collection metadata content standard and include general information, domain specifications, and search capabilities. Collection metadata is returned by *scan* as an XML document.

ADL uses two XML-based strategies to encode and transmit metadata. *Semantic XML* uses tags that directly denote the semantic content of the metadata. This strategy is used when the metadata content is understood by ADL, as in the case of the collection metadata described above. A semantic XML document type definition (DTD) defines the structure and attributes of the metadata. (The complete DTD for the ADL collection metadata is available at http://alexandria.ucsb.edu/docs/metadata/ADL-collection-metadata.dtd.) *Syntactic XML* uses tags that describe only a hierarchy of named sections and, within sections, name-value pairs that associate values with arbitrary attributes. Provision has been made to reference the semantics of the attributes. This strategy is used when the semantic content of the metadata is opaque to ADL, as in the case of the full method of the metadata interface, which returns holding metadata of unknown type or origin.

The search interface: allows the client to query the searchable metadata in one or more of the collections accessible to the middleware. This interface exposes two methods: *start* immediately returns a query identifier and query results, in the form of holding scan records, are merged and ranked by the middleware until either the query terminates, or holding identifiers have been returned; *stop* terminates the query referenced regardless of its progress. This provides the client with an "escape hatch" for hung or long-running queries.

Clients formulate queries in terms of the ADL search buckets. The middleware server translates and forwards queries to the collection servers, which are responsible for evaluating queries in ways that are meaningful for their respective collections. The query passed to the *start* method specifies combinations of, and/or Boolean constraints on, the ADL search buckets. Each collection supported by the middleware must specify a mapping from its own metadata into the search bucket attributes. This mapping will almost always be many-to-one; i.e., there will inevitably be a loss of precision in querying the search buckets versus querying the collection-specific metadata directly. However, by exposing only a single high-level set of searchable metadata, the ADL middleware allows clients to be built that can both exploit search bucket semantics (e.g. spatially manipulate the "location" bucket), and search all of ADL via a single connection.

In the current ADL implementation (and using the geographic location bucket as an example), a search bucket is the union of: (1) a conceptual set of contents (e.g., "any geographic locations or regions associated with collection items"); (2) a set of allowable representations for the contents (e.g., geographic bounding boxes, points, and polygons); (3) a set of query operators (e.g., overlaps and contains); and (4) a name (e.g., geographic location) that uniquely identifies the bucket and its conceptual contents, allowable representations, and query operators. Uniqueness guarantees that identically-named buckets behave identically, regardless of the underlying collection. Search bucket semantics, and in particular query operators, are loosely defined in the current implementation, since mandating precise semantics can incur prohibitive performance penalties.

Each collection server registered with the middleware server is described by collection-level metadata, structured in XML according to an ADL-defined standard. The metadata includes a set of XML bucket

elements that describe the search bucket(s) supported by the collection. There is no requirement that collection servers support every bucket, but all collection servers are required, as a matter of ADL policy, to support the geographic location bucket.

The results interface: simply provides access to result sets stored by the middleware. Each result set stores query results in the form of scan records (in ranked order); the originating query; and associated query statistics. Result sets can be accessed at any time, including during query processing, thus allowing clients to poll the middleware and thereby track the progress of a query.

The metadata interface: provides access to partial and full metadata for specified holdings. A typical sequence is for the metadata interfaces to be invoked to evaluate the results of a search. This interface exposes two methods. *scan* returns a small subset of the specified holding's metadata, including: (1) those search buckets most likely to have a single, unambiguous value (location, date, type); (2) the holding's title; and (3) identifiers for the holding's collection and any parent-level metadata, as well as for the holding itself, which can be used in subsequent data or metadata retrievals. These attributes were selected to maximize the utility of the scan metadata for quick evaluation, while minimizing the amount of per-holding information that a client must manipulate. For example, the current ADL client automatically retrieves all scan metadata for all holding identifiers returned by the search interface. *full* returns all available metadata for the specified holding. As noted above, there is no guarantee that a client will be able to interpret any particular attribute in a given holding's metadata. However, the XML packaging always allows a client to unambiguously parse the metadata and, for example, display it in the hierarchy implied by the nesting of the group tags. The XML packaging also supports references to semantic definitions of the elements.

The holdings interface: allows the client to retrieve library holdings using their holding identifiers. These identifiers are usually obtained from the search interface. However, since ADL holding identifiers are persistent, they can be saved (or, for example, emailed to a colleague) and then passed directly to the holdings interface during a subsequent session.

The holdings interface exposes three methods. *thumbnail* returns a reduced-resolution image rendition of the corresponding holding as a MIME-typed byte stream. Thumbnail images are small (typically 100 by 100 8-bit pixels, compressed with GIF or JPEG) in order to minimize download time. They are intended to help a library user make quick "go/no-go" decisions about whether to pursue a (possibly much) larger, full-resolution version of the holding. *browse* also returns a reduced-resolution (but more detailed than thumbnail) image rendition of the corresponding holding. Browse resolution is informally defined as "big enough to make an informed decision as to whether the full-resolution holding is worth retrieving." Our browse images are typically about 500 by 500 pixels. *access* returns an XML document (the access report), from which full-resolution versions of the holding may be retrieved.

5.2 Implementation

The ADL Client: currently has two implementations. The first is a web interface for the CDL, which requires a standard Web browser as its interface, This interface is implemented with a thin server-side layer that translates the ADL middleware protocols into HTML pages. The second is a more powerful Java Interface to Geographic Information (JIGI), which is implemented entirely in Java and is distributed as a self-installing stand-alone application, for any platform (e.g., Windows, Macintosh, Sun, SGI) supporting release 1.1 or higher of the Java Runtime Environment. As a standalone application, JIGI has more functionality than a browser-hosted applet, and also avoids the incompatibilities between different browsers' implementations of the Java virtual machine.

The ADL Middleware: is responsible for implementing the client-middleware interfaces described above, and mapping them into interactions with an arbitrary number of metadata catalogs. Additionally, the middleware implements whatever access controls ADL requires. ADL clients communicate with the middleware via HTTP because of the latter's ubiquity and simplicity as well as the ease with which current HTTP servers can be extended to support the required level of functionality.

Each family of interfaces supported by the ADL middleware is bound to a particular URL. This eliminates the need for top-level switch logic to direct requests to the appropriate handlers, but does require that the client be aware of which URL implements which interface. All ADL interfaces are implemented as HTTP GETs or POSTs to the interface's URL. By convention, the pathname portion of the URL ends with interface/method. Method parameters are passed according to the CGI encoding conventions. Except as noted, return values are provided as ASCII text (MIME type text/plain). An error message may be returned in lieu of whatever other return values the method supports. The current implementation of the ADL middleware layer extends a generic HTTP server with servlets or Java components that run in the server's execution context. This replaces an AOLserver-specific implementation.

ADL middleware supports the access policy dictated by ADL. Currently the specific access control supported in the implementation is host-based, which is used to refuse connections from clients that are not running at an ADL-approved Internet address. This is currently used to limit access to ADL to only those hosts connected to an IP network managed by the University of California. To this end, the access-control module maintains an explicit list of network numbers from which it will entertain connections. This mechanism is inherently unscalable, but suffices to meet the UC-only distribution restrictions imposed by third parties on some proprietary materials in the ADL collections (e.g., commercial remote sensing imagery.)

The ADL middleware maps queries expressed in terms of XML DTD's and holdings requests into queries specific to the underlying ADL servers. If multiple servers, or multiple databases on a single server, must be queried, this layer handles the requisite fan-out/fan-in. Since we currently assume that multiple databases are disjoint, the fan-in process is currently simple collation, with no duplicate detection or other conflict resolution.

In the current implementation, a basic architectural assumption is that the underlying servers will expose views that are, as closely as the particular server allows, exact correlates to the ADL search buckets and metadata sections. Thus, the translation functionality of the mapping layer is currently limited to "transliterating" the ADL query language into various dialects of server query languages (e.g., SQL).

The ADL middleware maintains a pool of client connections to whatever underlying servers are currently supported. The database connection layer is responsible for presenting a single functional interface to these shared connections. This serves to both localize whatever special knowledge (e.g. database client library) is needed to communicate with a particular server, and to minimize the setup or teardown time associated with making or breaking database client-server connections.

The ADL Servers: will increase in number and diversity over time, requiring a formalization of the middleware-server interface. A single generic driver is used to connect to relational databases, while additional drivers allow collections to be supported by indexed files or gateways to non-ADL services. In general, the ADL architecture imposes only the general requirement on collection servers that they expose query and metadata interfaces that the middleware can map to the client interfaces described above. In particular, there is no requirement that the collection server implement a specific internal metadata schema, as long as appropriate views of that schema are exposed to the middleware.

While there is not an architectural requirement that the server be a database system, all currently implemented ADL collection servers are relational databases. Comprehensive relational schemata allow the collection servers to fulfill a custodial role that, while not required by the ADL architecture, is often important to the organizations supporting the collections. For these organizations, the collection server doubles as a long-term metadata repository, which would have to be duplicated if it could not be directly connected to ADL. Using relational databases for collection servers also means that the middleware can formulate queries in a semi-standard language (SQL). However, since SQL queries are schema-specific, this does not by itself guarantee a standard middleware-collection interface. What is also required are standard views in the collection databases. These views present the appropriate collection-specific metadata attributes in terms of attributes that are, as nearly as possible, equivalent to the ADL search buckets. As part of the middleware-collection connection protocol, the collection server returns the text of the specific SQL queries that apply to its "bucket view". The middleware then loads these queries into its query mapping facility.

The general strategy, then, is for the collection server databases to implement views that correspond as closely as possible to the relevant client-middleware interfaces, and to furnish the middleware SQL query templates for those views at the time a middleware-collection connection is established. This strategy has, so far, allowed us to support several different collections with quite different internal schemata, most notably a heterogeneous catalog with over 2.5M entries, and a gazetteer with 4.8M entries.

The collections servers in the current ADL are implemented using the Informix Dynamic Server -- Universal Data Option (IDS-UDO) database management system. IDS-UDO extends the standard relational data model with abstract data types, which we exploit to support queries at the collection level that more closely match the capabilities of the ADL search buckets. Specifically, we use the MapInfo SpatialWare spatial data types to implement spatial searches, and the Verity text data types to implement text searches. Compared to standard relational database types, these extended types provide both additional search functionality, and more efficient indexing for very large collections. However, there is nothing inherent in the middleware-collection interface that would prohibit collections being implemented in any particular metadata management system.

5.3 Interoperability

Much of the architecture of ADL is related to *interoperability*, which denotes the degree to which an ADL client can access non-ADL services and collections and the degree to which non-ADL clients can access ADL collections and services. Three aspects of the ADL architecture relate primarily to interoperability. First, the bucket architecture may be viewed as allowing ADL clients to interact with heterogeneous, ADL and non-ADL collections. Second, the creation of a client-side interface to he Simple Digital Library Interoperability Protocol (SDLIP, see http://www-diglib.stanford.edu/ testbed/doc2/SDLIP.) provides ADL clients with an ability to search non-ADL collections, as well as allowing systems that support SDLIP to search ADL collections. SDLIP specifies functional and interface standards for basic search and retrieval services and may be viewed as providing an interface between ADL's middleware and other systems. As such, it addresses only the structural aspects of such services while specific issues, such as query languages and metadata fields and formats, are left for higher-level standards to define, or for clients and servers to negotiate. The third aspect relating to interoperability is a server-side interface to the Z39.50 protocol. This permits ADL clients to search other collections supporting Z39.50 and systems supporting Z39.50 to search ADL collections.

6 Security

Security of collections and their use, while always an issue of importance for valuable collections, is of particular concern in digital libraries distributed over heterogeneous collections having variable requirements for access control and used in a wide variety of applications. The ADL project has been exploring the Safe Area of Computation (SAC) approach [8], which uses a collection of trusted devices to enforce the protection of users from the insecurity of specific applications. The goal of each such device, or SAC, is to provide islands of security that interact with an ocean of insecurity. The implementation of SAC technology is currently restricted to an ADL testbed supporting a special test subcollection.

7 Evaluation

The evaluation of ADL from a user's point of view has been a critical component of ADL activities during its entire development. We have obtained [4] useage scenarios from target user groups to develop user requirements and preferences and have evaluated the degree to which they were met using a variety of methodologies including; online surveys, ethnographic observations, internal evaluations of interface design, and analysis of the use of ADL in university classrooms, with the most valuable information typically coming from user logs. From these multiple sources, we have derived an understanding of the expectations of users concerning the interface, functionality, and content of ADL. We have incorporated the results of these evaluations into the iterative process of design and development.

8 The ADEPT Project: a Brief Summary

The main goal of ADEPT is to extend the services and collections of ADL by constructing (1) services for creating new collections and managing existing collections, creating new objects from client-supplied metadata, importing objects from remote collections into local collections, and personalizing collections with the use, for example, of personal preferences and concept maps; (2) collections of multimedia, georeferenced information including dynamic/simulation models of spatially distributed processes; and (3) graphical user interfaces employing the concept of a "Digital Earth". An important near-term objective for ADEPT is building prototypical collections of simulation models and related digital objects that support undergraduate learning in physical, human, and cultural geography and related disciplines and evaluating whether the use of such resources helps students learn to reason scientifically.

ADEPT's extensions to the ADL architecture diverge from traditional client-server architectures in being modeled as a set of distributed peers. Each peer supports the three-tiered architecture as well as one or more collections of objects, subsets of which may appear as collections in other nodes. Future directions for the ADEPT architecture include supporting a richer information model, search over multiple collections, and supporting dynamic content such as models of Earth processes and other executable programs.

Three prototypical ADEPT collections of multimedia objects are being constructed for use in lectures, self-paced labs, and individual and collaborative learning sessions. The goals of these prototypes are to provide information that aids in designing future collections and services and serving as a basis for evaluating the educational consequences of ADEPT services. These collections focus on (1) physical geography (including tectonic phenomena); (2) human geography (e.g., diffusion phenomena); and (3) cultural geography, (including religious sites.) Initial versions of the first two have already been tested in electronic classroom presentations. The collection building process involves (1) acquisition; (2) cataloging; (3) ingesting; (4) evaluating with respect to user scenarios; and (5) construction of ISO-

standard topic maps. The collections may be searched by concept and reorganized with different levels of granularity into personal collections for various pedagogic purposes. Catalog descriptions of objects employ the ADEPT Learning Objects Metadata Model (ADEPT LOMM) which is based on IMS, as extended by DLESE [9], to include the geospatial reference fields as developed by ADL, IEEE LOM, and a model developed within ADEPT for representing simulations. The model's key educational and pedagogical elements include (1) type of learning resource; (2) learning context; (3) interactivity of resource; (4) interactivity level; (5) description. The ADEPT LOMM elements are mapped to ADL generic query search buckets to enable searching using the ADL system/search engine

9 References

[1] Alexandria Project: *http://www.alexandria.ucsb.edu*

[2] The Alexandria Digital Earth Modeling System (ADEPT): Towards a distributed digital model of the earth in support of learning, 1999. *http://www.alexandria.ucsb.edu/adept/proposal.pdf*

[3] Frew, J., Freeston, M., Hill, L., Janee, G., Larsgaard, M., Zheng, Q.: Generic Query, 1999. Metadata for Geospatial Digital Libraries. In: Proceedings of the Third IEEE META-DATA Conference, Bethesda, MD. *http://computer.org/proceedings/meta/1999/papers/55/jfrew.htm*

[4] Hill, L., Carver, L., Larsgaard, M., Dolin, R., Smith, T.R., Frew, J., and M. Rae, 2000.Alexandria Digital Library: User Evaluation Studies and System Design. Journal of the American Society for Information Science, 51(3), 246-259.

[5] Hill, L., Frew, J., Zheng, Q.: Geographic names: the implementation of a gazetteer in a georeferenced digital library. D-Lib Magazine (January 1999) *http://www.dlib.org/dlib/january99/hill/01hill.html*

[6] Hill, L., Janee, G., Dolin, R., Frew, J., Larsgaard, M., 1999. Collection metadata solutions for digital library applications. Journal of the American Society for Information Science 50:13, 1169-1181

[7] Smith, T., Andresen, D., Carver, L., Dolin, R., Fischer, C., Frew, J., Goodchild, M., Ibarra, O., Kemp, R., Kothuri, R., Larsgaard, M., Manjunath, B., Nebert, D., Simpson, J., Wells, A., Yang, T., Zheng, Q., 1996. A digital library for geographically referenced materials. Computer 29:5, 54-60,

[8] dos Santos, A. L. M., 2000. Safe Areas of Computation (SAC) for Secure Computing, PhD dissertation, University of California, Santa Barbara, CA.

[9] DLESE: *http://www.dlese.org/*