The ADEPT Digital Library Architecture

Greg Janée
Department of Computer Science
University of California, Santa Barbara
+1 (805) 893 8453
gjanee@alexandria.ucsb.edu

James Frew
Donald Bren School of Environmental Science and Management
University of California, Santa Barbara
+1 (805) 893 7356
frew@bren.ucsb.edu

ABSTRACT

The Alexandria Digital Earth ProtoType (ADEPT) architecture is a framework for building distributed digital libraries of georeferenced information. An ADEPT system comprises one or more autonomous libraries, each of which provides a uniform interface to one or more collections, each of which manages metadata for one or more items. The primary standard on which the architecture is based is the ADEPT bucket framework, which defines uniform client-level metadata query services that are compatible with heterogeneous underlying collections. ADEPT functionality strikes a balance between the simplicity of Web document delivery and the richness of Z39.50. The current ADEPT implementation runs as servlet-based middleware and supports collections housed in arbitrary relational databases.

Categories and Subject Descriptors

H.3.7 [**Digital Libraries**]: Systems Issues; D.2.12 [**Interoperability**]: Data mapping; H.3.5 [**Online Information Services**]: Web-based services.

General Terms

Design, Standardization.

Keywords

bucket framework; collection discovery; distribution; interoperability; metadata.

1. INTRODUCTION

The Alexandria Project [25] is a consortium of researchers, engineers, and educators, spanning the academic, public, and private sectors, working to develop distributed digital libraries for heterogeneous georeferenced information. *Distributed* means that a library's components may be spread across the Internet, as well as coexisting on a single desktop. *Heterogeneous* means that a library may contain multiple types of digital information, including non-traditional items such as remotely-sensed imagery, executable models, and multimedia instructional materials. *Georeferenced* means that, whenever possible, each item in a library is associated with one or more regions on the Earth's surface. (We refer to these regions as the item's *footprint*.)

The original motivation of the Alexandria Project was to create a digital library that could both reproduce and extend the content and functionality of a traditional research map library, specifically the Map and Imagery Laboratory (MIL) at the University of California, Santa Barbara (UCSB) Davidson Library. In pursuit of this objective the project developed three successive versions of the Alexandria Digital Library (ADL) architecture: a "rapid prototype" [7] system comprising a relational database of map and imagery metadata, accessed through a desktop geographic information system (GIS); a "web prototype" [10] system which replaced the stand-alone GIS with an HTTP server, generating an HTML forms-based user interface accessible via the World Wide Web; and the "ADL-3" [8] system which extended the HTTP server into full-fledged middleware, supporting HTTP interfaces to multiple clients, and connections to multiple catalog databases.

As our experience with digital libraries grew, the focus of the Alexandria Project broadened from a geospatial library to an integrated environment for managing, querying, and presenting geospatial information, especially for instructional applications [26]. We refer to these new goals by the umbrella term Alexandria Digital Earth ProtoType (ADEPT).

This paper presents the ADEPT digital library architecture. We introduce the architecture's fundamental concepts, and describe the ADEPT bucket framework, the organizing principle for ADEPT queries and metadata. We then detail the architecture's standard interfaces and describe its current (January 2002) implementation. We conclude with a description of the new collection discovery service.

2. ARCHITECTURE OVERVIEW

In this section we describe the ADEPT architecture in terms of its component objects and the information flows between them.

2.1 Objects

The ADEPT architecture is composed of three kinds of objects: items, collections, and libraries.

Items are the fundamental objects in ADEPT, and correspond directly to the "holdings" (e.g. books) in a traditional library. (We avoid the term "holding", with its connotations of physical custody, when referring to digital objects.) Items in ADEPT have identity, but no other innate properties -- all information or services available for an item must be accessed through higher-level objects. By not requiring items to do anything other than exist, ADEPT allows for the broadest possible definition of a digital library (e.g., a digital catalog of non-digital information).

Collections are sets of items. Information about individual items, as well as about the collection as a whole, is maintained at the collection level. It is thus possible for different information about the same item to appear in different collections. Collections may be heterogeneous (i.e., may "contain" different kinds of items); however, there are advantages, especially with respect to collection-level summarization, to having collections be homogeneous.

Libraries are sets of collections. In most cases, "using" ADEPT means accessing the client-level services provided by a library. Libraries expose a single standard set of interfaces to all their collections, making it possible to issue a single query against multiple collections. (By contrast, the interfaces to collections are not standardized; instead, a library has standard mechanisms for adapting itself to whatever interfaces the collection exposes.) Libraries may also form peer-to-peer relationships in which they grant each other remote access to their collections, by redirecting queries and responses to and from each other.

An ADEPT system is, minimally, a network of one or more autonomous libraries, each of which provides a uniform interface to one or more collections, each of which manages metadata for one or more items (Figure 1).

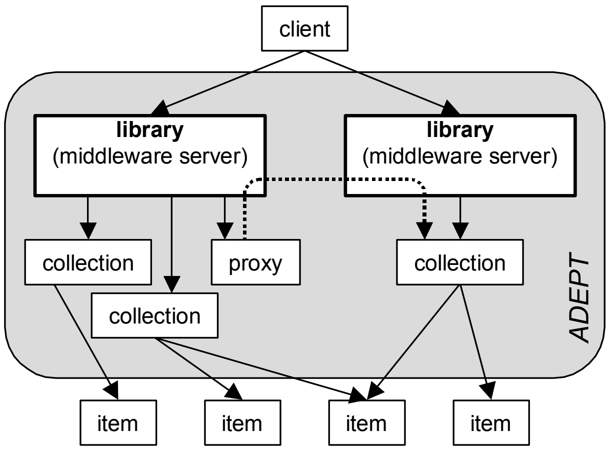

Figure 1: basic ADEPT architecture

As in the CRADDL architecture [17], a collection is the fundamental unit of organization, being responsible for the metadata for both its items and itself. A library is, in effect, a "collection broker," mediating standardized access to its collections. In terms of a standard three-tier architecture, collections correspond to the server tier, and libraries to the middleware tier. Any application that can access the middleware's client interfaces can be an ADEPT client -- there is no specific client defined (or required) by the architecture.

2.2 Information

The ADEPT architecture supports three kinds of externally-visible information: metadata, queries, and result sets.

2.2.1 Metadata

Metadata is information about items or collections. In the ADEPT architecture, metadata moves between objects as *reports*, which are documents pertaining to single items or collections.

The following kinds of metadata reports are defined by the ADEPT architecture, specifically as XML DTDs (i.e., each report is an XML document):

- *Collection* reports pertain to collections as a whole. They include descriptive information about the collection (e.g. its origin), as well as summary information derived from the collection's items (e.g., their geographic distribution; see Figure 3)
- A *bucket* report contains the item's metadata mapped (insofar as is possible) into the ADEPT bucket framework (described in the next section).
- A *scan* report contains a brief view of the item's metadata mapped (again, insofar as possible) to the ADEPT core buckets.
- A *full* report contains as much of the item's metadata as the collection is willing to provide. There are no constraints on the semantics of a full report; the DTD merely specifies a generic hierarchical encoding suitable for a wide variety of metadata content standards.
- A *browse* report contains references (e.g., URLs) to one or more reduced-resolution representations of the item's contents (e.g., a thumbnail image). These are typically used by graphical user interface clients as iconic representations of the item, and also to help human users qualitatively evaluate the item's suitability for a particular purpose.
- An *access* report contains references (e.g. URLs) to one or more representations of the item's content. Multiple representations allow an item's content to be made available in a variety of formats, packagings, compressions, etc.

Note that the access report is the only means by which an item's content is exposed by ADEPT. Furthermore, access reports are optional -- the ADEPT architecture does not require that an item's content be accessible; or, if it is, that it be accessible in any particular standard format. This allows an ADEPT collection to refer to an item that it does not own or cannot access.

2.2.2 Queries and result sets

An ADEPT *query* is a predicate directed at one or more of a library's collections. A *result set* is the (possibly empty) set of identifiers of the items that satisfy the query. ADEPT clients issue queries to find items of interest, and use the identifiers from result sets to obtain metadata reports about the items.

ADEPT queries have standard semantics, defined by the ADEPT bucket framework. This fundamental architectural underpinning of ADEPT is described in the next section.

3. ADEPT BUCKET FRAMEWORK

Our experience with various collection-providing groups (including MIL, the Earth System Science Workbench [6], and the UC Berkeley Digital Library Project [28]) is that collection development is difficult, laborious, understaffed, and underfunded. Constraints imposed by the group's mission, customer requirements, idiosyncrasies associated with the collection content, etc. typically require the group to focus most of its effort on providing customized, content-specific access to their collections. To succeed as an alternative, universal means of access, ADEPT must minimize the requirements it places on already-burdened collection providers.

It is also our experience that, despite the availability of many well-known metadata standards, collection-providing groups often work with (or are forced to work with) locally-defined, idiosyncratic metadata. Even government agencies such as the U.S. Geological Survey and its affiliates, which are nominally mandated to use the FGDC Content Standard for Digital Geospatial Metadata [5], use many different standards, ranging from the published and documented (e.g., [20]) to the entirely undocumented. ADEPT must therefore allow collection providers to use *source metadata* -- metadata unprocessed and untransformed from its native state as employed by the collection -- to the greatest extent possible. It is ADEPT's goal to provide uniform client services across heterogeneous collections, and in particular, to provide 1) common high-level search and description services, and 2) structured means of discovering and exploiting lower-level, collection- and/or item-specific search and description services.

These two considerations -- placing minimal requirements on collection providers, and providing uniform client services -- lead to the *ADEPT bucket framework*: a specification governing semantics, mapping, and representation of metadata for the specific purpose of providing certain search and description services. The framework is predicated on the following assumptions:

- Collection items have metadata.
- Item metadata is of relatively high quality, with known, well-defined semantics.
- Item metadata is mappable to "core" buckets (described below).

Although these assumptions clearly do not apply universally [16], in our experience enough collections satisfy these assumptions to make the ADEPT architecture of value.

The current bucket framework reflects extensive changes from the original specification [9], including unification of search and description functionality and semantics, completion of the semantic definition of the buckets and delineation of bucket types, and support for metadata source tracking and qualified searches.

3.1 Bucket definition

A *bucket* is an abstract, strongly typed metadata category with defined search semantics, to which source metadata is mapped. The ADEPT bucket framework specification defines the semantics, syntactic representation, and functional behavior of buckets, as well as the mapping of source metadata fields.

We use the term "metadata category", as opposed to "metadata field," to emphasize that the framework is *not* a metadata content standard. Collection providers need not use buckets directly or internally; rather, the framework includes support for explicitly representing the mapping of source metadata to buckets. Consider an item that has FGDC source metadata, by which we mean that the item directly associates values with FGDC metadata fields, for example:

```
Citation/Originator = "U.S.
Geological Survey"
```

In the ADEPT bucket framework, buckets are populated with tuples indicating the metadata value, the source of the metadata, and its appropriateness for discovery. (Metadata marked as inappropriate for discovery is ignored during searches but still returned as part of the item's description.) The mapping of the above example to the "Originator" bucket would be represented conceptually as:

```
Originator = {(FGDC, 1.1/8.1,
"Citation/Originator", "U.S.
Geological Survey", searchable)}
```

We say that buckets are *abstract*, or high-level, because they are intended to aggregate semantically similar metadata fields. Buckets elide small semantic and syntactic differences in favor of providing uniform, high-level search and description services. The aggregation may occur across the items in a collection: for example, one item in a collection may populate the `Originator` bucket with the FGDC `Citation/Originator` field, while another may populate it with the MARC `Main Entry--Personal Name` field. Or, a single item may populate a bucket with multiple values from the same or different metadata sources, as in this example:

```
Originator = {(FGDC, 1.1/8.1,
"Citation/Originator", "U.S.
Geological Survey", searchable),
(USGS DOQ, PRODUCER, "Producer",
"Photo Science, Inc.", searchable)}
```

Combining metadata mapping and aggregation allows a client to operate at two levels. A client can search for items, and a collection can describe items, by a high-level category like `Originator` without the client having to understand source metadata fields or their content standards. At the same time, a client can view the lower-level, source metadata if desired. The ADEPT bucket framework also supports "drilling down" into buckets; that is, searching by specific metadata fields. For example, a client can search a collection's `Originator` bucket, but match on only the FGDC `Citation/Originator` field where it has been mapped.

Buckets are strongly typed, and thus restrict data type and representation of the values they may be populated with. For example, the `Originator` bucket specifies just an unstructured `text` type, but the `Date` bucket declares that it can be populated only with calendar dates, and ranges of calendar dates, expressed in ISO 8601 [14] format.

A bucket also defines its search semantics by specifying its allowable query operands and operators, as well as the semantics of matching against items that have mapped multiple values to the bucket. For example, the `Date` bucket specifies that it can be searched by specifying a range of calendar dates and using one of the operators `intersects`, `contains`, or `within`. Multiple values mapped to the `Date` bucket are treated severally; i.e., the constraint is applied to each value independently and the results are then logically OR-ed together. Search semantics are typically left slightly under-specified to accommodate multiple implementations. For example, the `Originator` bucket defines a `contains-all-words` query operator, but the exact definition of what constitutes a "word" is left to the implementation.

The advantage of combining (semantically loose) metadata aggregation with (semantically rigorous) strong typing is that the ADEPT bucket framework achieves rich search functionality while remaining flexible enough so as to have broad applicability.

Buckets have simple names such as `Date` and `Originator`. Although there is no mechanism for qualifying names, we intend that there be relatively few commonly agreed-upon buckets, with collections defining only a limited number of more domain- and item-specific buckets, and so we do not anticipate name clashes.

Buckets are generally independent and orthogonal, but there are cases where two buckets have a relationship to each other. For example, the `Title` bucket is a sub-bucket of the `Subject-related text` bucket, meaning that the values mapped to the `Subject-related text` bucket must include the values mapped to the `Title` bucket, and therefore every match against the `Title` bucket is a match against the `Subject-related text` bucket. Such relationships are defined by convention only; the ADEPT bucket framework provides no formal means of expressing such relationships.

3.2 Bucket types

We have found it advantageous during software development to work with bucket types. A *bucket type* captures that portion of a bucket definition that has functional implications: the data type and syntactic representation of bucket values, and the allowable query operators and operands. The purely semantic portions of a bucket definition (the bucket's name and semantic definition) are then left to configuration files and such.

ADEPT has defined six bucket types (Table 1.)

Table 1: ADEPT bucket types

bucket type	value type	Query	
		term type	operators
spatial	any of several types of geometric regions defined in WGS84 latitude / longitude coordinates, expressed in an ADEPT-defined syntax	WGS84 box or polygon	`contains`, `overlaps`, `is-contained-in`
temporal	(range of) calendar date(s) in ISO 8601 syntax	same as value type	
hierarchical	term from controlled vocabulary or thesaurus		`is-a`
textual	text		`contains-all-words`, `contains-any-words`, `contains-phrase`
qualified textual	text with optional associated namespace		`matches`
numeric	real number in standard scientific notation		standard relational operators

Other bucket types are certainly conceivable, such as bucket types based on new metadata data types; bucket types providing domain-specific searches (e.g., that support finding satellite imagery by ubiquitous path/row identifiers); and bucket types that provide actual content searches (e.g., that support finding images by content-based characteristics). In the current framework, adding a bucket type requires modifying central XML specifications. A future version of the framework may make this process more distributed and extensible.

3.3 The ADEPT core buckets

The bucket *types* described above are defined architecturally, but the *buckets* actually in use are defined by collections and items. To mitigate the chaos that would result from each collection defining its own unique buckets, ADEPT has defined a set of standard buckets, to support cross-collection uniformity. Essentially, defining these standard buckets amounts to reserving certain bucket names.

The ten *core* buckets (Table 2) are intended to be simple, universally applicable, easily and broadly populated, and above all, useful for concise description and searchability. The buckets were originally [9] designed based on our experiences working with geospatial data and other types of scientific data, and after looking closely at similar efforts such as GILS and Dublin Core.

The ADEPT core buckets share many similarities with Dublin Core [3, 4] in particular. The differences between the two frameworks result from their different origins and purposes. Dublin Core was developed to facilitate description, while buckets have been developed to support searching. For example, the qualified Dublin Core field `DC.Coverage.Spatial` defines multiple means of specifying geographic locations (coordinate-defined points and boxes, country codes, place names, etc.), the primary intent being to increase specificity and decrease ambiguity. However, such a system is too general to support a spatial search service. The ADEPT `Geographic location` is more flexible than `DC.Coverage.Spatial` in some ways (e.g., it accepts more types of coordinate-defined geographic regions), but it is also more tightly constrained in other, key ways (e.g., it accepts *only* coordinate-defined regions) so as to support a spatial search service.

Table 2: ADEPT core buckets

Bucket name	Bucket type	Description	Approximate Dublin Core equivalent
`Subject-related text`	textual	text indicative of the subject of the item, not necessarily from controlled vocabularies.	`DC.Subject`
`Title`		the item's title. This bucket is a sub-bucket of Subject-related text.	`DC.Title`
`Assigned term`		subject-related terms from controlled vocabularies. This bucket is a sub-bucket of `Subject-related text`.	qualified `DC.Subject`
`Originator`		names of entities related to the origination of the item	`DC.Creator + DC.Publisher`
`Geographic location`	Spatial	the subset of the Earth' surface to which the item is relevant	`DC.Coverage.Spatial`
`Date`	Temporal	the calendar dates to which the item is relevant	`DC.Coverage.Temporal`
`Object type`	hierarchical	ADL Object Type Thesaurus (image, map, thesis, sound recording, etc.)	`DC.Type`
`Feature type`		ADL Feature Type Thesaurus (river, mountain, park, city, etc.)	none
`Format`		ADL Object Format Thesaurus (loosely based on MIME)	`DC.Format`
`Identifier`	Qualified textual	names and codes that function as unique identifiers	`DC.Identifier`

3.4 Role of buckets in the ADEPT architecture

The ADEPT bucket framework plays three specific roles in the ADEPT architecture of the ADEPT core system: describing items, searching for items, and characterizing collections.

The bucket framework provides an XML format for encoding an item's source metadata mappings. The format is capable of representing mappings to any buckets that adhere to the ADEPT bucket types. A collection returns a document in this format as the "bucket" view of the item-level metadata for an item.

The framework also provides an XML format for the ADEPT query language. The query language is capable representing Boolean combinations of constraints against arbitrary buckets, and arbitrary source fields mapped to buckets, again subject only to their adherence to the ADEPT bucket types. A query submitted to the ADEPT middleware is a document in this format.

Finally, the XML format for the ADEPT collection-level metadata includes structures for recording statistical overviews of the collection's metadata mappings. For example, a collection can record that it supports the `Originator` bucket, and moreover that a given number of its items have an FGDC `Citation/Originator` field and map that field to that bucket. In this way a client can gain substantial information regarding a collection's support for buckets and the nature of the underlying, source metadata.

3.5 Comparison with Other Approaches

In any system in which information flows from provider to consumer, it is instructive to look at what knowledge about the underlying information is captured and exploited by the system itself, for this is indicative of how much functionality the system can provide over that information. To take an extreme example, a system that treats all input as bit streams affords relatively little functionality (copy bits, compress stream, etc.), whereas a system that recognizes that inputs are word processing files can provide much richer functionality (e.g., repagination).

To look at two examples in the realm of digital libraries, consider first the World Wide Web, which for the purposes of this discussion we define as HTML documents delivered via the HTTP protocol. Although one can express all sorts of things in HTML, the actual *semantic* content of HTML tags and HTTP headers is extremely low. As a consequence, the functionality afforded by basic Web technology is limited to delivering documents and rendering them for direct human consumption.

By contrast, in a traditional library cataloging system, Machine-Readable Cataloging (MARC) [18] records prepared according to the Anglo-American Cataloging Rules (AACR2) [11] are queried and delivered via the Z39.50 [29] protocol. This system explicitly captures extremely fine-grained distinctions (e.g., "date of publication" versus "date of treaty signing"). So, in theory (though admittedly, not in practice) such a system affords much richer functionality.

These two examples represent extremes of the tradeoffs between structure and standardization (or, more negatively, complexity and "heaviness") on the one hand, and flexibility and generality (or, simplicity and "lightness") on the other. This tradeoff is graphically represented in Figure 2:

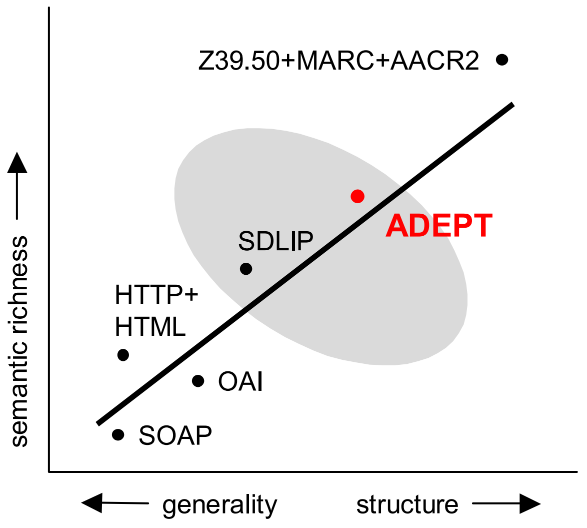

Figure 2: Interoperabilty tradeoffs

We believe there is a "sweet spot" in this figure: a set of architectural design tradeoffs that provide sufficient structure so as to afford rich functionality, and yet retain sufficient simplicity and generality so as to be flexible, broadly applicable, and easily implemented.

The Simple Digital Library Interoperability Protocol (SDLIP) [22] represents one effort to attain this sweet spot. The ADEPT architecture represents another, with ADEPT providing slightly greater structure so as to afford richer functionality. As a concrete example of the differences between SDLIP and ADEPT, consider how the two systems treat query languages. SDLIP does not specify a query language; it provides only a means of identifying query languages. Thus an SDLIP client wishing to query will get no support from SDLIP itself; query languages are effectively part of the underlying information flow, i.e., something to be interpreted and negotiated on a collection-by-collection basis.

By contrast, ADEPT provides a standard query language. The language is generic and extensible, and there is some flexibility in how collections may implement the language, but nevertheless, there are sufficient structures in the language, and there is sufficient commonality across collections, that it is possible for an ADEPT client to query arbitrary collections with no a priori analysis or agreements required.

4. IMPLEMENTATION ISSUES

4.1 Middleware

The ADEPT middleware server has been the basis for the Alexandria Digital Library's operational system since 1999, and as such it is relatively robust and reliable. Newer features such as intra-bucket searches, personal collections, and the collection discovery service are still under development at this writing (April 2002).

The current ADEPT middleware server is written in Java and Python and can be run as a web application inside a servlet container [2], as a standalone RMI server, or both. Personal collections are supported by an embedded Berkeley DB database [24]. External collections are supported by a generic relational database connection.

The middleware currently supports the following public client interfaces, each as an HTTP-addressable servlet:

- *Configuration*: Get a list of the collections the middleware is currently connected to.
- *Collection*: Get the collection metadata document for the specified collection.
- *Query*: Pass a query to one or more collections. Queries are asynchronous. Running queries may be terminated with the *Cancel* interface. Query result sets accumulated by the middleware are retrieved with the *Results* interface.
- *Metadata*: Get a standard metadata report for the specified item from the specified collection.

The middleware also supports maintenance interfaces that report its status, and manage its connections to local and remote collections.

4.2 Support for relational databases

To date, most ADEPT collections have been implemented with relational technology (i.e., metadata fields stored in tables in relational database management systems). We have therefore developed a generic software component, the "Bucket99 driver", which accesses a relational database over a JDBC connection and which translates bucket-level search and description requests into appropriate SQL statements. The Bucket99 driver makes an arbitrary relational database look like an ADEPT collection.

The crux of the driver is a generic query translator, implemented in the Python scripting language, which converts Boolean combinations of query constraints into an SQL query. The translator works from an abstract model of a relational bucket implementation (an extensive configuration file supplies the necessary specifics), comprising, for each bucket:

- A table *T* whose rows correspond to items in the collection that have value(s) for the bucket, and a column within *T* that holds item identifiers.
- Zero or more *auxiliary tables* that are related to *T* by given relationships and that are required to implement bucket semantics.
- A *table cardinality* that indicates how many table rows are potentially associated with each collection item. Possible values are: exactly one; zero or one; zero or more; one or more.
- An *implementation paradigm*: a set of rules that describes how an atomic constraint against the bucket is translated into a constraint against *T* and *T*'s auxiliary tables. Specifically, a paradigm is a coordinated set of functions that translates a constraint into an SQL `WHERE` clause fragment. ADEPT supplies over a dozen paradigms for commonly-encountered implementation strategies. For example, for textual buckets ADEPT supplies paradigms that support the Verity and Excalibur text engines as well as two paradigms that use `VARCHAR` columns. For spatial buckets ADEPT supplies paradigms for two Informix spatial DataBlades (MapInfo and Geodetic) as well as a paradigm implementing bounding boxes with numeric columns.

- An optional *field selection condition*. If drill-down capability for the bucket is implemented using multiple rows per item, one row per mapped source field, then this condition selects the appropriate row.

The real power of ADEPT's query translation system is that *any* relational structure matching the Bucket99 abstract model can be treated as a bucket. The model is semantically rich enough to enable the query translator to translate arbitrary boolean combinations of bucket query constraints into SQL.

4.3 Access Control

The ADEPT architecture provides *access control points* (ACPs) at each middleware service and at each interface between a middleware service and a collection's implementation of the service. At each ACP a *gatekeeper* may be installed. Functionally, a gatekeeper accepts the parameters of a proposed service request and returns a boolean allow/disallow response.

The current middleware implementation includes several "generic" gatekeepers, which examine only the generic attributes of a request (e.g., the client's IP address).

- *ConstantGatekeeper*: always allow or disallow.
- *IPAddressGatekeeper*: allow if client IP address matches pre-configured patterns.
- *BasicPasswordGatekeeper*: allow if HTTP "Basic" username/password authentication is satisfied.
- *CombinationGatekeeper*: allow if a Boolean combination of other gatekeepers is satisfied. (E.g., "Allow all requests from IP address set *S1*, or all requests from IP address set *S2* that also specify a valid username/password from database *D1*.")

Gatekeepers not yet developed (but fully supported by the ADEPT architecture) could support finer-grained access control. For example, a collection-level gatekeeper to the middleware's Metadata service might allow access based on both the client making the request and the specific report being requested.

Gatekeepers can be installed and independently configured at any ACP. Gatekeepers are installed by dynamic class loading, so new gatekeepers can be developed without modifying ADL in any way.

4.4 Collection statistics

We have also developed a software package for gathering, processing, and rendering the statistical portion of the collection-level metadata (the spatial and temporal histograms and the counts of items) for collections implemented in relational databases. The package requires only a command-line SQL interface to the database, and hence is largely independent of database vendor, but it is otherwise highly Unix-specific and relies on numerous third-party packages (NetPBM, Gnuplot, etc.). We plan to rewrite this package to be more in line with the middleware server, and to be more easily configured.

5. COLLECTION DISCOVERY SERVICE

A client wishing to access ADEPT must first know what libraries and collections exist. For this reason the ADEPT architecture has been extended with a central *collection discovery service*: a registry of known collections and a search capability that allows clients to find relevant collections.

5.1 Central Registry

Any distributed system must provide some kind of entrance or starting point for clients. Completely decentralized systems such as the World Wide Web and Gnutella [15] eschew any central architectural artifact in favor of non-architectural mechanisms (e.g., word-of-mouth). For ADEPT's purposes, we believe it is simpler, and sufficiently manageable and scalable, for there to be a central registry of known collections.

The ADEPT collection discovery service (CDS) provides such a registry. A middleware server coming online registers itself with the CDS, at a well-known address. Thereafter the CDS periodically (e.g., daily) retrieves the collection-level metadata for each of the middleware server's collections. (The "update frequency" field in the collection-level metadata might be used someday to suggest appropriate polling frequencies.) A middleware server that fails to respond after some interval (e.g., a week) is considered defunct.

There can be more than one CDS -- we anticipate collection discovery services oriented around domains or user communities such as the National Science Digital Library (NSDL) [19]. A middleware server may register itself with any number of collection discovery services.

The CDS registry is sufficient for finding collections, but as a complete solution it scales very poorly. Given little or no information about collections, a client wishing to find information in the system can do so only by querying every collection every time. The unworkability of this approach was demonstrated by an early version of the FGDC Clearinghouse [21]. Visitors to the Clearinghouse website in 1999-2000 were presented with a simple list of collections (*sites* in Clearinghouse parlance) with no context other than a name (e.g., "Southwest Data Center Clearinghouse"). Additional information about sites could only be obtained manually and individually. (The Clearinghouse has since improved its interface to include a preliminary collection identification phase.)

5.2 Collection Relevance

The CDS harvests, stores, and indexes collection-level metadata for each known collection. ADEPT has specified a content standard for this metadata [13]. The portions of the standard relevant to this discussion include:

- spatial histograms indicating the geographic coverage of the items in the collection. A spatial histogram is a two-dimensional histogram whose domain is the Earth's surface, and in which a cell value indicates the number of collection items that 1) have a value for the `Geographic location` bucket, and 2) whose value overlaps the cell. A graphic rendering of a spatial histogram is shown in Figure 3. The data structure computed and stored is an Euler histogram, which is a slight variation of a simple histogram that supports range searching [27].

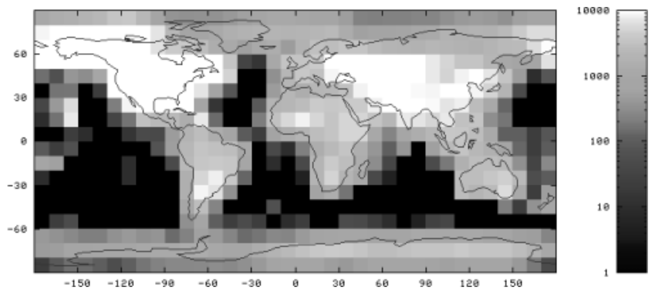

Figure 3: Spatial histogram example

- analogous to the spatial histogram, temporal histograms indicating the calendrical coverage of the collection. Specifically, a temporal histogram is a one-dimensional histogram whose domain is calendrical time, and in which a cell value indicates the number of collection items that 1) have a value for the `Date` bucket, and 2) whose value overlaps the cell. Again, the histogram is an Euler histogram.
- The controlled vocabularies used to categorize items in the collection, and, for each vocabulary term, the number of items in the collection relevant to the term. If a vocabulary is a thesaurus (that is, supports broader/narrower relationships between terms), the counts reflect the specificity of the terms. So, for example, if "images" is a broader term for "photographs", then the number of images includes the number of photographs.

The CDS allows clients to rank collections by each of the above three quantities. For example, a client specifies a spatial or temporal query in the form of a geographic region or time period, and the CDS returns a (possibly truncated) list of the known collections ranked by the number of items each collection has that spatially or temporally overlap the query region. A vocabulary query takes the form of a single vocabulary term, and ranking of results is by the number of items associated with the term. Conjunctions of queries are discussed under Limitations, below.

It should be noted that the CDS's query language is essentially a subset of middleware's query language, i.e., the query language used to query individual collections. Thus a client can formulate a single query that can be used to, in effectively one step, both identify relevant collections and subsequently search for items within those collections.

5.3 Scalability

Evaluating a query region against an Euler histogram takes constant time [1]. The histograms and item counts amount to only a few kilobytes per collection, and thus can all be stored in primary memory. We therefore anticipate that the CDS can easily accommodate tens of thousands of collections, and answer queries with reasonable speed, without having to resort to more sophisticated indexing and querying techniques.

5.4 Limitations

There are several limitations to our approach to collection discovery. First, the CDS, as described above, cannot perform joined queries. For example, consider the query "Find collections that are likely to contain aerial photographs of Southern California taken in the 1930s." The CDS can rank collections by the numbers of items relevant to Southern California; by the numbers of items relevant to the 1930s; and by the numbers of aerial photographs. But it cannot rank collections on the joint condition because the underlying statistics are gathered independently. Currently we approximate joint conditions by assuming that the statistics are linked. Under this heuristic, a collection with lots of aerial photographs and good coverage of the 1930s is presumed to contain relatively many aerial photographs taken in the 1930s. We are exploring new data structures that explicitly capture joint statistics and yet are still compact and easy to compute [23].

Second, the CDS relies on binary statistics: an item is either relevant or it is not. A world map drawn on the back of a napkin thus has the same spatial relevance as a high-resolution, multi-terabyte dataset with worldwide coverage. We are working on incorporating resolution (specifically, minimum feature size) as a way of approximating information density. (I.e., the dataset has much higher information density than the napkin, and thus would rank higher.) Ideally, one could constrain both minimum feature size *and* information density -- after all, there are times when one indeed wants the napkin and not the dataset.

Third, the CDS is not especially effective for collections that are not well discriminated by geographic space, calendrical time, or controlled vocabulary terms, or which contain items that lack the metadata supporting such discrimination. A possible solution may be to augment the CDS with an approach based on word frequency, such as that used in STARTS [12].

6. ACKNOWLEDGEMENTS

We thank Larry Carver, Linda Hill, Mary Larsgaard, Terry Smith, and David Valentine for their contributions to the design and implementation of the ADEPT architecture. The work described herein was supported by the National Science Foundation under cooperative agreement IIS-9817432.

7. REFERENCES

[1] Beigel, R. and Tanin, E., The geometry of browsing. in Latin American Symposium on Theoretical Informatics, (Brazil, 1998), 331-340.

[2] Coward, D. Java Servlet API Specification, Sun Microsystems, Inc., 2001.

[3] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1: Reference Description, Dublin Core Metadata Initiative, 1997.

[4] Dublin Core Metadata Initiative. Dublin Core Qualifiers, Dublin Core Metadata Initiative, 2000.

[5] Federal Geographic Data Committee. Content standard for digital geospatial metadata (FGDC-STD-001-1998), Federal Geographic Data Committee, Washington, DC, 1998.

[6] Frew, J. and Bose, R., Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products. in SSDBM 2001: Thirteenth International Conference on Scientific and Statistical Database Management, (George Mason University, Fairfax, VA, 2001), IEEE Computer Society, 180-189.

[7] Frew, J., Carver, L., Fischer, C., Goodchild, M., Larsgaard, M., Smith, T. and Zheng, Q., The Alexandria Rapid Prototype: building a digital library for spatial information. in 1995 ESRI International User Conference, (Palm Springs, CA, 1995), Environmental Systems Research Institute, Inc.

[8] Frew, J., Freeston, M., Freitas, N., Hill, L., Janee, G., Lovette, K., Nideffer, R., Smith, T. and Zheng, Q. The Alexandria Digital Library architecture. International Journal on Digital Libraries, 2 (4). 259-268.

[9] Frew, J., Freeston, M., Hill, L., Janee, G., Larsgaard, M. and Zheng, Q., Generic query metadata for geospatial digital libraries. in Third IEEE META-DATA Conference, (National Institutes of Health, Bethesda, Maryland, 1999).

[10] Frew, J., Freeston, M., Kemp, R., Simpson, J., Smith, T., Wells, A. and Zheng, Q. The Alexandria Digital Library testbed. D-Lib Magazine, 2 (7/8).

[11] Gorman, M. and Winkler, P.W. (eds.). Anglo-American Cataloguing Rules, 2nd ed. (AACR2). American Library Association, Chicago, IL, 1978.

[12] Gravano, L., Chang, C.-C.K., Garcia-Molina, H. and Paepcke, A., STARTS: Stanford Proposal for Internet Meta-Searching. in SIGMOD 1997: Proceedings of the ACM SIGMOD International Conference on Management of Data, (1997), ACM Press, 207-218.

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

[14] nternational Organization for Standardization. Data elements and interchange formats - information interchange - Representation of dates and times (ISO 8601:1988), International Organization for Standardization, 1988.

[15] Kan, G. Gnutella. in Oram, A. ed. Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology, O'Reilly and Associates, Inc., Sebastopol, CA, 2001, 94-122.

[16] Lagoze, C. Keeping Dublin Core Simple: Cross-Domain Discovery or Resource Description? D-Lib Magazine, 7 (1).

[17] Lagoze, C. and Fielding, D. Defining Collections in Distributed Digital Libraries. D-Lib Magazine, 4 (11).

[18] Library of Congress Cataloging Distribution Service MARC 21 Format for Bibliographic Data. Library of Congress, 1999.

[19] Manduca, C.A., McMartin, F.P. and Mogk, D.W. Pathways to Progress: Vision and Plans for Developing the NSDL, National Science, Mathematics, Engineering, and Technology (SMET) Education Digital Library (NSDL), 2001.

[20] National Mapping Program. Standards for Digital Orthophotos, U.S. Geological Survey, National Mapping Division, 1996.

[21] Nebert, D., Supporting Search for Spatial Data on the Internet: What it means to be a Clearinghouse Node. in 1996 ESRI International User Conference, (Palm Springs, CA, 1996), Environmental Systems Research Institute, Inc.

[22] Paepcke, A., Brandriff, R., Janee, G., Larson, R., Ludaescher, B., Melnik, S. and Raghavan, S. Search Middleware and the Simple Digital Library Interoperability Protocol. D-Lib Magazine, 6 (3).

[23] Riedewald, M., Agrawal, D. and El Abbadi, A., Flexible Data Cubes for Online Aggregation. in International Conference on Database Theory (ICDT), (2001), 159-173.

[24] Sleepycat Software Inc. Berkeley DB. New Riders Publishing, 2001.

[25] Smith, T. and Frew, J. Alexandria Digital Library. Communications of the ACM, 38 (4). 61-62.

[26] Smith, T.R., Janee, G., Frew, J. and Coleman, A., The Alexandria Digital Earth Prototype System. in JCDL 2001: First ACM/IEEE-CS Joint Conference on Digital Libraries, (Roanoke, VA, 2001), ACM Press, 118-119.

[27] Sun, C., Agrawal, D. and El Abbadi, A., Exploring spatial datasets with histograms. in International Conference on Data Engineering (ICDE 2002), (San Jose, CA, 2002), (to appear).

[28] Wilensky, R. UC Berkeley's Digital Library project. Communications of the ACM, 38 (4). 60.

[29] Z39.50 Maintenance Agency Information Retrieval (Z39.50): Application Service Definition and Protocol Specification (ANSI/NISO Z39.50-1995). NISO Press, Bethesda, MD, 1995.