NASA/TM-2001-211426

A Survey of Complex Object Technologies for Digital Libraries

Michael L. Nelson
Langley Research Center, Hampton, Virginia

Brad Argue, Miles Efron, Sheila Denn, and Maria Cristina Pattuelli
University of North Carolina, Chapel Hill, North Carolina

December 2001

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2001-211426

A Survey of Complex Object Technologies for Digital Libraries

Michael L. Nelson
Langley Research Center, Hampton, Virginia

Brad Argue, Miles Efron, Sheila Denn, and Maria Cristina Pattuelli
University of North Carolina, Chapel Hill, North Carolina

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

December 2001

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

A Survey of Complex Object Technologies for Digital Libraries

Michael L. Nelson
NASA Langley Research Center
MS 124
Hampton VA 23681
m.l.nelson@larc.nasa.gov

Brad Argue, Miles Efron, Sheila Denn and Maria Cristina Pattuelli
University of North Carolina
School of Information and Library Science
Chapel Hill NC 27599
{argub, efrom, denns, pattm}@ils.unc.edu

Abstract

Many early web−based digital libraries (DLs) had implicit assumptions reflected in their architecture that the unit of focus in the DL (frequently "reports" or "e−prints") would only be manifested in a single, or at most a few, common file formats such as PDF or PostScript. DLs have now matured to the point where their contents are commonly no longer simple files. Complex objects in DLs have emerged in response to various requirements, including: simple aggregation of formats and supporting files, bundling additional information to aid digital preservation, creating opaque digital objects for e−commerce applications, and the incorporation of dynamic services with the traditional data files. We examine a representative (but not necessarily exhaustive) number of current and recent historical web−based complex object technologies and projects that are applicable to DLs: Aurora, Buckets, ComMentor, Cryptolopes, Digibox, Document Management Alliance, FEDORA, Kahn−Wilensky Framework Digital Objects, Metadata Encoding & Transmission Standard, Multivalent Documents, Open eBooks, VERS Encapsulated Objects, and the Warwick Framework.

1.0 Background

Many web−based digital libraries (DLs) evolved from departmental report servers or pre−print collections, where compressed PostScript files (and later PDF files) were often the complete manifestation of the logical "document". Even as DLs became progressively more sophisticated in acquiring their holdings and the services provided on them, they still used the raw file format as the focus object. Consider the early DL, UCSTRI (http://www.cs.indiana.edu/ucstri/info.html; Van Heyningen, 1994). UCSTRI would harvest metadata from a list of ftp servers and use heuristics to match the README and abstract files with the file listings of the ftp server. Search services were provided on the metadata harvested from various sites, with the pointers to individual ".ps.Z" files. Though far more advanced, ResearchIndex (http://www.researchindex.com/; Lawrence, Giles & Bollacker, 1999) is similar in to UCSTRI in that it crawls a known list of web pages to extract PostScript and PDF files. It provides an impressive list of services based on extracted metadata and citation information from those files, but the requirement remains that ResearchIndex have direct access to the PostScript and PDF files themselves.

Of course, there are notable exceptions. The physics e−print archive (http://www.arxiv.org; Ginsparg, 1994) handles aggregation through dynamic conversion from the author−supplied TeX source files. The Dienst protocol (Davis & Lagoze, 2000) has a complex document object model built into the protocol to handle the many files resulting from scanned images (TIFFs, OCR, composite GIFs), in addition to the standard PostScript and PDF. While admittedly a subtle distinction, these DLs can be characterized as providing sophisticated services on specific file formats, and not deferring the complexity into the object itself.

Similarly, some DLs, such as software focused DLs like Netlib (http://www.netlib.org/; Browne et al., 1995), can ship Unix tar files (."tar"), PC zip files (".zip"), Linux RPM files (."rpm") or similar aggregation / compression formats, but the usage model for these are slightly different. These formats are generally used for transport, and the user typically interacts with these files using additional applications leaving the user's experience with the files outside of the DL environment.

Of course, creating complex objects for aggregation and preservation pre−dates web−based DLs. For example, a number of filesystem projects and scientific data set projects have implemented to achieve some of the same goals. The Extensible File System (ELFS) (Grimshaw & Loyot, 1991; Karpovich, Grimshaw & French, 1994; Karpovich, French & Grimshaw, 1994) implemented object−oriented (OO) technologies such as inheritance and encapsulation on "file objects" in Unix filesystems. ELFS allowed various access methods for the object (i.e., row−major or column−major access), providing a layer of abstraction around data storage and data access in a filesystem environment. Slightly less OO but still offering unique aggregation capabilities, Nebula / Synopsis (Bowman et al., 1994; Bowman & Camargo, 1998) and the Semantic File System (Gifford et al., 1991) allowed user−definable aggregation of file objects through "directories" created by queries on automatically extracted metadata from file objects.

In scientific data, NetCDF (Unidata NetCDF, 2001), HDF (Folk, 1998) and SmartFiles (Haines, Mehrotra & Van Rosendale, 1995) all offer varying implementations that provide roughly the same "self−describing" functionalities of encapsulating the data with the necessary metadata, units, accessing functions to assist in long term preservation of the data semantics as well as the structural syntax to allow for conversion and translation. Bundling the metadata with the data itself is especially important for scientific data because unlike text documents, metadata for scientific data frequently cannot be regenerated if lost. Bundling also offers the advantage of being accessible to other filesystem commands ("cp", "rm", etc.).

The motivation for studying digital objects is not limited to just understanding the implementations of previous DL projects. Digital objects are expected to grow in sophistication and assume greater roles and responsibilities. For example, Kahn & Lyons (2001) discuss the use of digital objects for "representing value", including concepts generally associated with canonical physical manifestations such as financial records, monies, and deeds. Erickson (2001a; 2001b) argues that advances in digital rights management can be made by embedding rights information within digital objects themselves and defining a "digital object infrastructure" to handle open, fine−grained negotiation and enforcement of individual intellectual property rights policies. Additionally, Kang & Wilensky (2001) draw from the peer−to−peer model and define how digital objects can be "self−administering" through internalizing declarative descriptions of their service requirements and matching these descriptions with the availability of a wide−spread infrastructure of "self−administering data handlers". The technologies surveyed in this paper can be considered the vanguard to these more ambitious applications of digital objects.

1.1 Technologies Surveyed

We studied a number of current and historical projects and models that employed complex objects suitable for DL use. This list should not be considered exhaustive, but we have tried to review all the technologies of which we are aware. To restrict the size and scope, generalized technologies were not considered, such as portal systems (e.g., Metaphoria/Smartcode (Shklar et al., 1998)) component software environments (e.g., JavaBeans, OLE/COM, OpenDoc, etc. (Ibrahim, 1998)) or object−oriented databases (e.g., Gemstone, Objectivity, etc. (Zand, Collins & Caviness, 1995)). While not specifically designed for DL applications, these technologies could be deployed in DL environments, and some are the base technologies used to implement some of the DL−specific systems described below.

It should stressed that this is not a review against performance criteria, but rather a survey to raise awareness of the various projects and outline their unique capabilities and features. Each of the technologies were created to address different problems, so readers will have to judge which technology best addresses requirements similar to theirs. However, a rough grouping of technologies does emerge: e−commerce/security (Cryptolopes, Digibox, eBooks), aggregation of content and services (Aurora, Buckets, FEDORA, Kahn−Wilensky Digital Objects, Metadata Encoding & Transmission Standard, Warwick Framework), annotation and collaboration (ComMentor, Document Management Alliance, Multivalent Documents) and preservation (VERS Encapsulated Objects).

1.2 Survey Methodology & Structure

The survey was conducted by reading the literature that was available, the availability of which varied greatly from project to project. Where possible, we also downloaded and ran demonstration software (this was obviously not possible for recently defunct projects and projects that presented only theoretical models). We have also borrowed screenshots and figures from reports and webpages where we thought them to be useful. The results of our analysis are presented in the following format:

- Overview
    - Developers
    - Motivation
- Analysis
    - Architecture
    - API
    - System Requirements
    - Example Application
- Summary
    - Strengths
    - Weaknesses
    - Future Directions

The format will occasionally deviate where applicable, but generally follows this structure. While we strived to be thorough and accurate, any errors that may be present are our responsibility, and not that of the systems developers.

2.0 Aurora

2.1 Overview

Aurora is an architecture for enabling dynamic composition of distributed, autonomous web services in an open environment (Marazakis, Papadakis & Nikolaou, 1997; Marazakis, Papadakis & Nikolaou, 1998). Aurora's architecture is based on a container framework that allows for unified access to disparate services. Aurora provides a run–time environment for HERMES, a specification language that defines resources and business rules or "service flows" (Nikolaou et al., 1997). Aurora complements middle–ware tools such as CORBA and Java applets as it addresses the collaboration and coordination needs of emerging network–centric applications such as digital libraries, e–commerce, and scientific collaborative work environments.

2.1.1 Developers

Aurora's primary architect is Christos Nikolaou, the head of the Parallel and Distributed Systems Division (PLEIADES) at the Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH), in Crete. Manolis Marazakis and Dimitris Papadakis, PhD students at the University of Crete Computer Science Department assisted with development as well. Most of the documentation related to the project was published in late 1997 and early 1998.

2.1.2 Motivation

Aurora addresses the need for a scripting environment that can tie together distributed objects. Its developers set out to extend the capabilities of CORBA and Java applets by providing coordination and collaboration services in a dynamic environment. Object frameworks such as CORBA have typically supported object creation and management but have not provided flexible ways to dynamically combine objects into applications via scripting. Aurora seeks to provide a generic infrastructure that enables flexible composition of distributed objects into "network–centric applications."

2.2 Analysis

The Aurora application model is based on a "service flow paradigm" in which composite services are unified in the context of a work session. Work sessions are comprised of series of simple and compound "tasks." Simple tasks are activities that use one resource/service while compound tasks require multiple resources. The Aurora architecture also supports the notion of an "event," which includes service request, state transition, and other application–specific messages. Resources, tasks, and events are described using metadata container constructs provided by the HERMES specification language.

2.2.1 Architecture

The Aurora architecture can be thought of in terms of five interrelated components: the container framework, metadata framework, repository service, session manager, and monitoring infrastructure.

In the Aurora architecture (Figure 2.1), application components are encapsulated by uniform containers. Containers are extensible shells that consist of a uniform management interface and component–specific metadata and code. The uniform management interface mediates all access to the container in addition to facilitating monitoring and control operations involving the component. The container framework also provides input and output ports as well as event detection and handling services.

Aurora uses a metadata framework to describe resources owned by external providers in order to determine which services may be useful for a particular task and to define access methods. An extensible, self–describing framework is specified using HERMES. The metadata framework describes only generic concepts such as resources, tasks, and events, enabling developers and providers of services to create application domain–specific sets of attribute–value pairs. The repository service manages this metadata in addition to providing directory and binding services. The directory service allows providers to advertise their services and applications to find the components that satisfy their requirements. The binding service dynamically binds tasks to available resources.

The Aurora session manager manages the session establishment and communication between container components. It acts as the run–time environment for containers, assuming on the responsibility of parsing the service flow script and determining which resources to use via the directory service. The session manager's distinct components include a task scheduler, container run–time environment, and logging system. This logging system provides the basis for a distributed monitoring infrastructure. The monitoring infrastructure tracks the progress and current state of service flows and maintains of all component interactions. It also provides an interface for querying system logs.

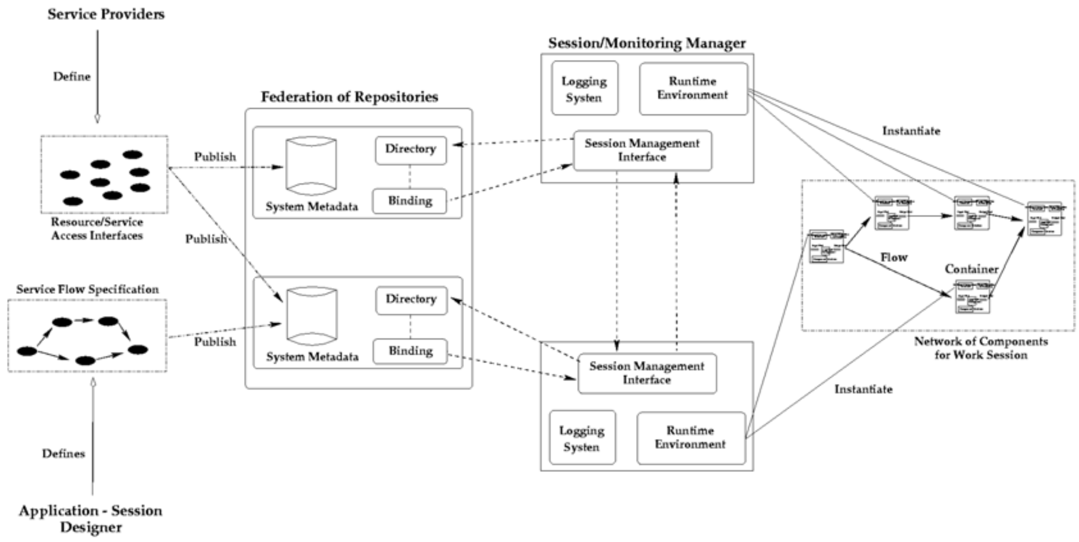

Service Providers
Define
Publish
Resource/Service Access Interfaces
Publish
Service Flow Specification
Publish
Defines
Application - Session Designer
Federation of Repositories
Directory
Binding
System Metadata
Directory
Binding
System Metadata
Session/Monitoring Manager
Logging Systen
Runtime Environment
Session Management Interface
Session Management Interface
Logging Systen
Runtime Environment
Instantiate
Flow
Container
Network of Components for Work Session
Instantiate

Figure 2.1. The Aurora Architecture (from http://www.ics.forth.gr/pleiades/projects/Aurora/aurora.html)

2.2.2 API

The Aurora documentation provides interface specifications for a number of the architecture components, including the directory service, uniform management interface, task scheduler, component interface, session manager, and communication interface. See Marazakis, Papadakis & Nikolaou (1997) for the complete API documentation.

2.2.3 Example Applications

The following HERMES code excerpt (Figure 2.2) defines a workflow (a.k.a. service flow) in which four services are involved: a collector, evaluator, broker, and provider. This application exemplifies a work session in which a consumer is searching for product information on the web. In this scenario, the user submits a string of keywords to a collector agent. The collector passes the query to a broker agent that returns some number of potential providers of this information. The broker service then communicates with the group of potential providers and an evaluator agent that filters the information returned by the providers. At run–time the session manager would process the code and manage coordination among the various encapsulated components.

```
DEFINE WORKFLOW SearchScenario {
  TYPE: AuroraProcess;
  # TYPE defines which "attribute model" to use
  INPUT: SearchKeywords: STREAM { Keyword: STRING; Weight: INTEGER; };
  OUTPUT: STREAM { FilteredResult: OBJECT; };
  PLAN: {
    # Upon session activation workflow input becomes Collector's input
    WHEN (SearchScenario.START) {
      Collector.START [ SearchScenario.Input BECOMES Collector.Input ];
    }
    WHEN (Collector.START) {
      Collector.START_PHASE("Broker Interaction");
      # Collector interacts with Broker, provides it with input,
      # and receives its output
      BrokerTask.START [ Collector.Output BECOMES BrokerTask.Input;
      BrokerTask.Output BECOMES Collector.Input ];
    }
  }
  WHEN (BrokerTask.DONE) { # Actions after Broker has finished its work
    Collector.END_PHASE("Broker Interaction");
    Collector.START_PHASE("Access Providers") [
    # Collector provides input to group members,
    # and receives their output
    Collector.Output BECOMES ProviderTaskGroup.Input;
    ProviderTaskGroup.Output BECOMES Collector.Input ];
    # Evaluator receives the same input as Collector, and its output
    # is the output of the workflow
    Evaluator.START [ Collector.Input BECOMES Evaluator.Input;
    Evaluator.Output BECOMES SearchScenario.Output];
  }
  WHEN (Collector.END_PHASE("Access Providers")) {
    ProviderTaskGroup.DONE;
    Evaluator.DONE;
    Collector.DONE;
    SearchScenario.DONE;
  }
  ATTRIBUTES:
  Description: "Workflow corresponding to the search scenario";
  Participants: {"Collector", "Evaluator", "Broker",
  "ProviderTaskGroup"};
}
```

Figure 2.2. HERMES Service Flow

2.3 Summary

Aurora addresses a common need among emerging network−centric applications such as digital libraries, eCommerce, and scientific collaborative work environments: namely, the ability to coordinate distributed web services and collaborate in a dynamically constructed, shared workspace. The Aurora architecture supports coordination and collaboration through flexible run−time composition of diverse components encapsulated in general−purpose containers. The session manager provides a run−time environment for containers and is supervised by a monitor. The repository service manages the required metadata, publishes a directory of resources, and handles the dynamic binding of tasks to available resources.

2.3.1 Strengths

Aurora is well suited for a digital library or other web application setting in which multiple distributed services need to be utilized within a unified workspace. Its architecture is designed to flexibly facilitate coordination of services as well as collaboration between users. Another strength of the Aurora framework is its reliance on open technologies and dynamic composition of objects at run−time.

2.3.2 Weaknesses

It is unclear from the available information whether Aurora was ever fully implemented. In any case, the work on the project does not appear to be proceeding and functioning software (if it exists) is not widely available. Development tools, especially for building HERMES scripts, are not readily available and would have to be developed as well to encourage its adoption.

2.3.3 Future Directions

The most recent Aurora articles found were published in 1998−1999. They indicated a number of future directions including incorporation of XML−based metadata and messaging as well as a potential user interface based on standard HTML, forms, and Java applets. However, the lack of more recent publications and updates to the Aurora site (http://www.ics.forth.gr/pleiades/projects/Aurora/aurora.html) leaves the current status of the project unknown.

3.0 Buckets

3.1 Overview

Buckets are an aggregative, intelligent construct for publishing in DLs. Buckets allow the decoupling of information content from information storage and retrieval. Buckets exist within the Smart Objects and Dumb Archives model for DLs (Maly, Nelson & Zubair, 1999) in that many of the functionalities and responsibilities traditionally associated with archives are "pushed down" (making the archives "dumber") into the buckets (making them "smarter"). Some of the responsibilities imbued to buckets are the enforcement of their terms and conditions, and maintenance and display of their contents. These additional responsibilities come at the cost of storage overhead and increased complexity for the archived objects. A bucket is a self−contained storage unit that has data and metadata, as well as the methods for accessing both.

3.1.1 Developers

Buckets were developed by Michael Nelson (NASA Langley Research Center) and others at Old Dominion University. They were first described in Nelson, Maly & Shen (1997).

3.1.2 Motivation

The development of buckets is guided by a number of design goals. As suggested by the SODA model, buckets have unique requirements due to their emphasis on minimizing dependence on specific DL implementations. The design goals are: aggregation, intelligence, self−sufficiency, mobility, heterogeneity and archive independence.

It is difficult to overstress the importance of the aggregation design goal. In previous experience with NASA DLs, data was often partitioned by its semantic or syntactic type: metadata in one location, PostScript files in another location, PDF files in still another location, etc. Over time, different forms of metadata were introduced for different purposes, the number of available file formats increased, the services defined on the data increased, new information types (software, multimedia) were introduced, the logging of actions performed on the objects became more difficult. The result of a report being "in the DL" eventually represented so much DL jetsam − bits and pieces physically and logically strewn across the system. The architecture of buckets reflects the reaction to this situation.

3.2 Analysis

Based on previous NASA DL experience, buckets have a two−level structure:

- buckets contain 0 or more *packages*
- packages contain 0 or more *elements*

Actual data objects are stored as elements, and elements are grouped together in packages within a bucket. A two−level architecture was considered sufficient for most applications, and thus employed as a simplifying assumption during bucket implementation. Current work involves implementing the semantics for describing arbitrarily complex, multi−level data objects.

An element can be a "pointer" to another object: another bucket, or any other arbitrary network object. By having an element "point" to other buckets, buckets can logically contain other buckets. Although buckets provide the mechanism for both internal and external storage, buckets have less control over elements that lie physically outside the bucket. However, it is left as a policy decision to the user as to the appropriateness of including pointers in an archival unit such as a bucket. Buckets have no predefined size limitation, either in terms of storage capacity, or in terms of number of packages or elements. Buckets are accessed through 1 or more URLs. For an example of how a single bucket can be accessed through multiple URLs, consider two hosts that share a file system:

http://host1.foo.edu/bar/bucket−27/
http://host2.foo.edu/bar/bucket−27/

Both of these URLs point to the same bucket, even though they are accessed through different hosts. Also, consider a host that runs multiple http servers:

http://host1.foo.edu/bar/bucket−27/
http://host1.foo.edu:8080/bucket−27/

If the http server running on port 8080 defines its document root to be the directory "bar", then the two URLs point to the same bucket.

Elements and packages have no predefined semantics associated with them. Authors can model whatever application domain they desire using the basic structures of packages and elements. One possible model for bucket, package, and element definition is based on NASA DL experiences. In Figure 3.1, packages represent semantic types (manuscript, software, test data, etc.) and elements represent syntactic representations of the packages (a `.ps` version, `.pdf` version, `.dvi` version, etc.). Other bucket models using elements and packages are possible.

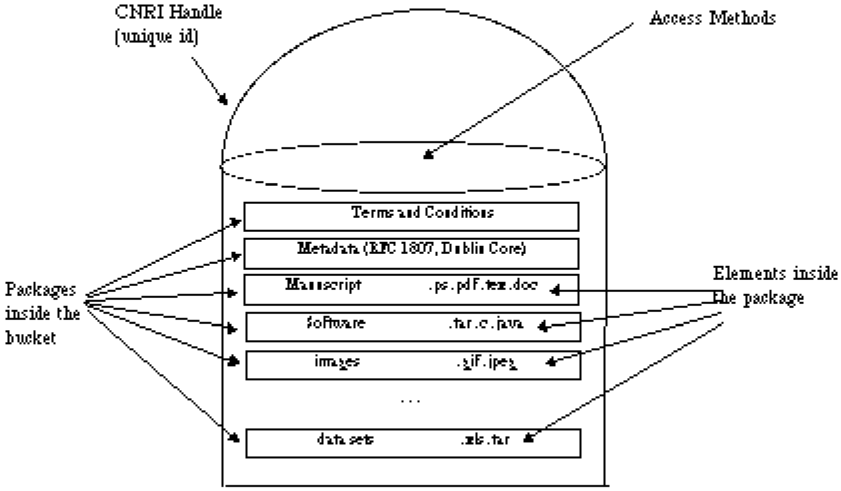


Figure 3.1. A model of a typical NASA bucket.

3.2.1 Architecture

The current implementation of buckets are written in Perl 5, and are accessed by an http server. Other non−Perl based implementations are being explored, and this description does not apply to them. Buckets take advantage of the package/element construct for their internal configuration. In addition to the user data entered as packages and elements, the bucket keeps its own files as elements in certain reserved packages. Thus, methods such as "add_element", "delete_element" and so forth can be used to update the source code for the bucket, update the password files, etc. Table 3.1 lists the predefined packages and some of the elements they contain. By convention, these packages begin with an underscore ("_") character. Figure 3.2 provides a model representation of the structure of a typical bucket, with internal packages and elements on the left and user–supplied data packages on the right.

Package	Elements Within the Package
_http.pkg	cgi–lib.pl – Steven Brenner's CGI library encoding.e – a list of MIME encoding types mime.e – a list of MIME types
_log.pkg	access.log – messages received by the bucket
_md.pkg	[handle].bib – a RFC–1807 bibliographic file other metadata formats can be stored here, but the .bib file is canonical
_methods.pkg	1 file per public method
_state.pkg	1 file per stored state variable
_tc.pkg	1 file per .tc (terms and condition) file password file & .htaccess file

Table 3.1. Reserved packages in buckets.

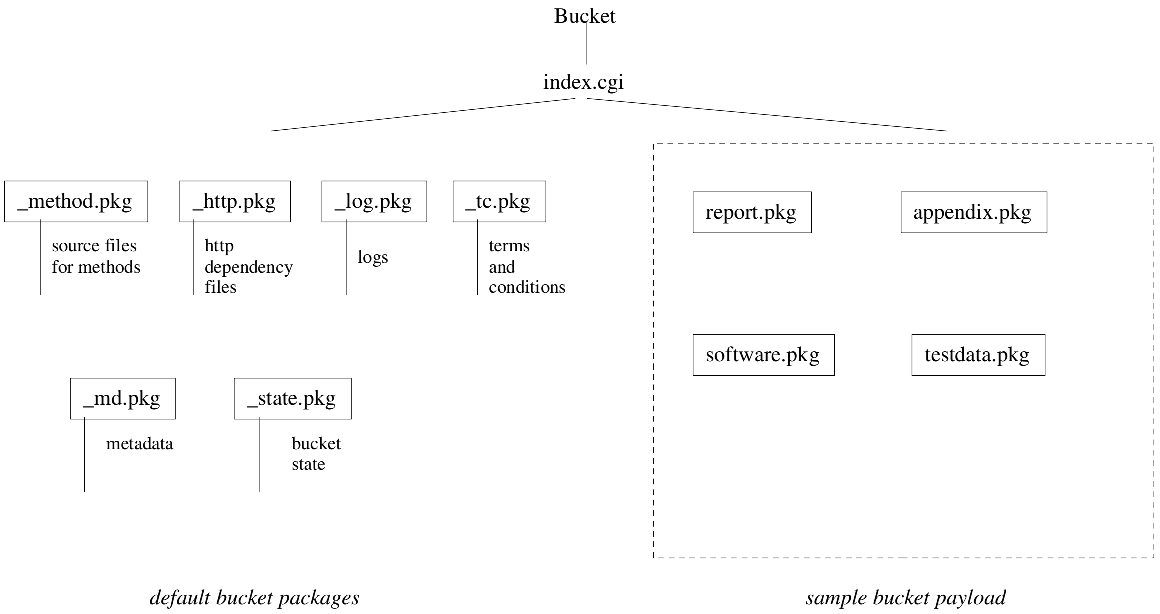


Figure 3.2. Internal bucket structure.

3.2.2 API

Regardless of bucket implementation, the bucket API is encoded using http as the transport protocol. There are currently approximately 30 methods define for bucket interaction and maintenance. The bucket API is fully defined in Nelson (2000).

3.2.3 System Requirements

From a server side, all that is needed to keep the current implementation of buckets functioning is Perl 5 (or better) and a http server that allows the ".cgi" extension. No other assumptions about the run−time environment or package installation are made. Alternate implementations of buckets are possible, but will continue to follow the minimalist server environment philosophy.

3.2.4 Example Applications

Readers are referred to the appendices in Nelson & Maly (2001) for QuickTime videos of buckets in operation.

3.3 Summary

3.3.1 Strengths

Buckets are well suited for aggregating heterogeneous content and remaining functional in low−fidelity environments. Since they are self−contained, independent and mobile, they should be resilient to changing server environments. Buckets can be adapted to a variety of data types and data formats.

3.3.2 Weaknesses

Bucket functionality introduces both storage overhead and management complexity overhead. Storage issues are only significant in very large numbers of buckets (> 100,000) and can be addressed. More significant is the issue of complexity management. Since buckets are designed to be independent, they contain much redundant information, introducing the possibility of the buckets becoming out of synchronization with each other. Tools and models for the management of large numbers of buckets exist, but remain largely untested.

3.3.3 Future Directions

Buckets are still being actively developed, with a focus on increasing their intelligence for digital preservation applications and consuming less resources than the current implementation. Various bucket versions and their notes can be accessed at http://dlib.cs.odu.edu/bucket−shop/.

4.0 ComMentor

4.1 Overview

ComMentor was a system designed to facilitate creating, sharing, and accessing annotations to web documents. ComMentor included a client−server protocol, a customized metadata description language called PRDM (Partial Redundant Descriptive Meta−Language), a server based on NCSA http 1.3, and a modified xMosaic 2.4 browser (Röscheisen, Mogensen & Winograd, 1997).

4.1.1 Developers

ComMentor was developed by members of Stanford University's Digital Library research group. The first references to ComMentor appear in 1994 (Röscheisen, Mogensen, & Winograd, 1994), and it appears that development was suspended toward the end of 1995. The most recent reference to ComMentor in the literature appears in 1997 (Röscheisen, Winograd, & Paepcke, 1997).

4.1.2 Motivation

There are a number of situations in which people would like to be able to communicate with one another about the documents they encounter in a networked environment. In many cases it would be convenient to have this kind of communication take place by being able to append information or annotations directly to a document so that others who are interested (such as other members of a workgroup, for example) would have a way to view those comments precisely at the point within a document to which they are relevant. In addition, it would be advantageous to be able to use such annotations as a way of providing multiple trails through a group of documents, or to be able to provide people with a way to evaluate and sift through documents based on collaborative filtering.

These functionalities are not supported by the native HTML format, and they all fall under the umbrella of providing ways to access certain kinds of meta−information associated with a document or group of documents. ComMentor was developed to address these issues and to devise a general architecture for capturing and managing this kind of meta−information.

4.2 Analysis

Objects in the PRDM language can be passed to browsers as the contents of a special MIME type message, or by embedding the information in HTML META tags. PRDM are interpreted on the client end through the use of a modified Mosaic 2.4 browser and CGI scripts written in Perl and C++.

4.2.1 Architecture

The ComMentor system architecture includes a client−server protocol, the PRDM metadata description language, a server built on NCSA http 1.3, and the modified Mosaic 2.4 browser.

Annotation objects are specified in PRDM, and they can be aggregated into sets. Each annotation refers to a specific location within a particular document. Each annotation has a particular server location and a URL−like identity for locating the annotation on that server. Control over access to annotations is handled at the set level − so there can be private, group, and public annotation sets. Any authentication procedures must be handled outside of the ComMentor architecture.

The browser used by ComMentor consists of the interactive renderer built into Mosaic, a document synthesis module, and a context control application. The document synthesis module is responsible for querying the appropriate servers for documents and any annotation sets associated with those documents, and includes functionality for synthesizing the document and its annotation set into one virtual document dynamically. The context control application is the module that mediates communication between the renderer and the document synthesis module and manages state information based on the user's option choices.

On the server side, the basic information unit is a PRDMitem. These items can include text content as well as links to other documents. The server manages an index of all of the annotation items that constitute a set. The system is designed so that the server that houses the annotation set for a document can be completely separate from the server that houses the original document.

The PRDM language that is used to create meta−information items is a typed, declarative object language. It is designed to be self−identifying in terms of the name and version of the language without having to know about the language's structure. Descriptions in PRDM can be partial, distributed and redundant, meaning that PRDMitems on a given server might provide only a partial description of all of the meta−information available about a given document. There may be overlap in the descriptive metadata that is stored about a document as part of different annotation sets on different servers, so in this way the objects may be redundant.

The procedures for performing the actual synthesis of documents and meta−information are contained in the merge library. The merge library takes as input a document and a list of annotation items and outputs the document with the annotations rendered in−line. For HTML and plain text, the in−line rendering is accomplished through the use of string position trees applied to a canonical representation of the document. Each annotation item has an associated highlighted text string and a position identifier string. The position identifier strings are designed to be robust in the face of document modifications. Any annotations whose position cannot be reconstructed are appended to the end of the document with a notation that they are unassigned.

4.2.2 API

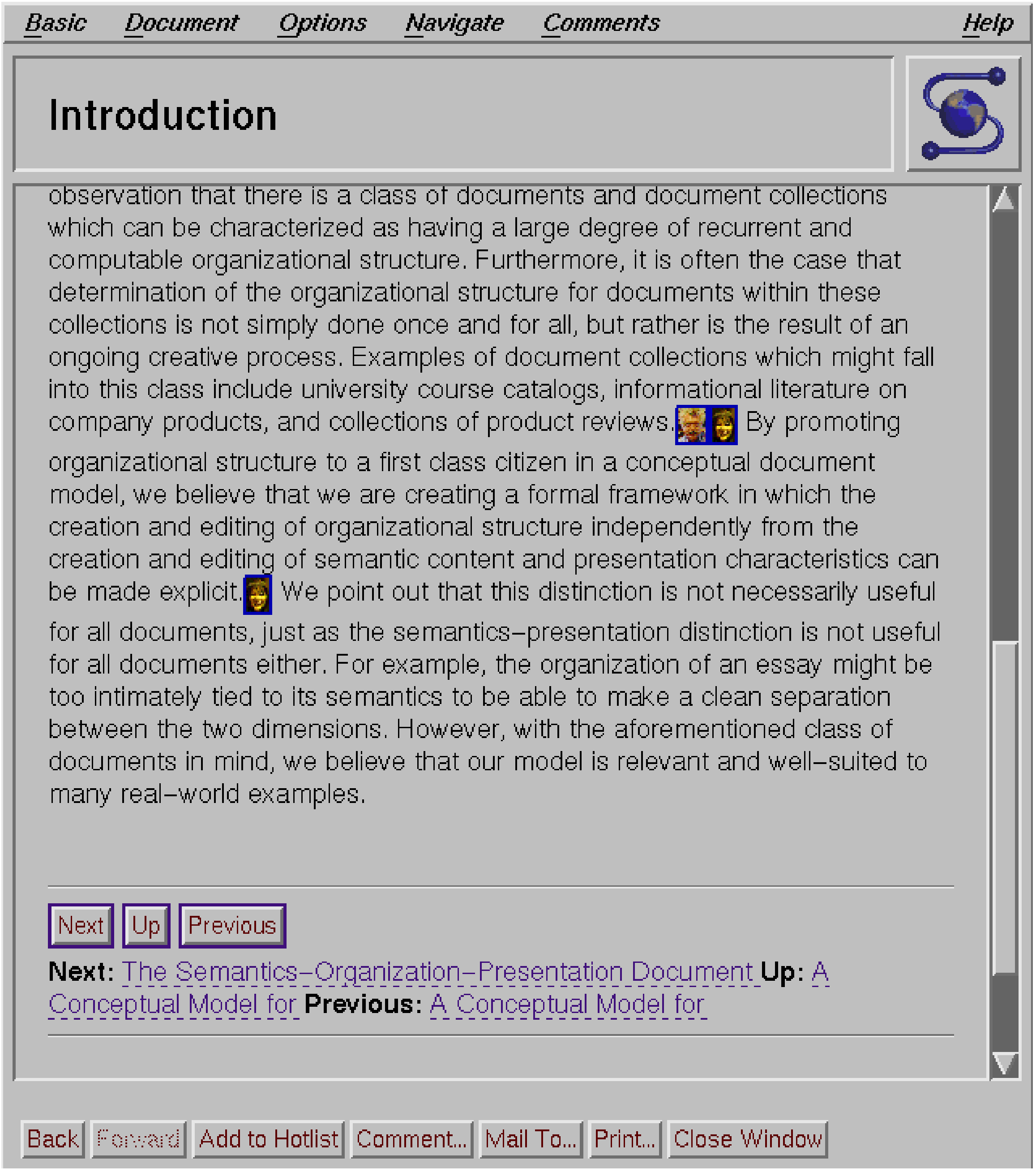
There is not an API per se (and the source code is no longer available from the Stanford Digital Library website) but there is a description of the PRDM language and the object interfaces available in Röscheisen, Mogensen & Winograd (1994) in Appendices B and C.

4.2.3 System Requirements

ComMentor does not run under any current operating systems or browsers, and is no longer available.

4.2.4 Sample Operation

In Figure 4.1, the modified HTML document is presented to the user with annotations marked as images in the text. Figure 4.2 illustrates a pop–up annotation by mousing over the facial image of the annotator. Figure 4.3 illustrates additional configurability showing text highlighting and the identify of the annotator diminished. (All screenshots taken from: http://www–diglib.stanford.edu/rmr/TR/shots/)



Inlined annotations are shown using in-line markers placed after the text they are commenting on.
Comments can be viewed using a quick viewer or viewed as a page.

Introduction

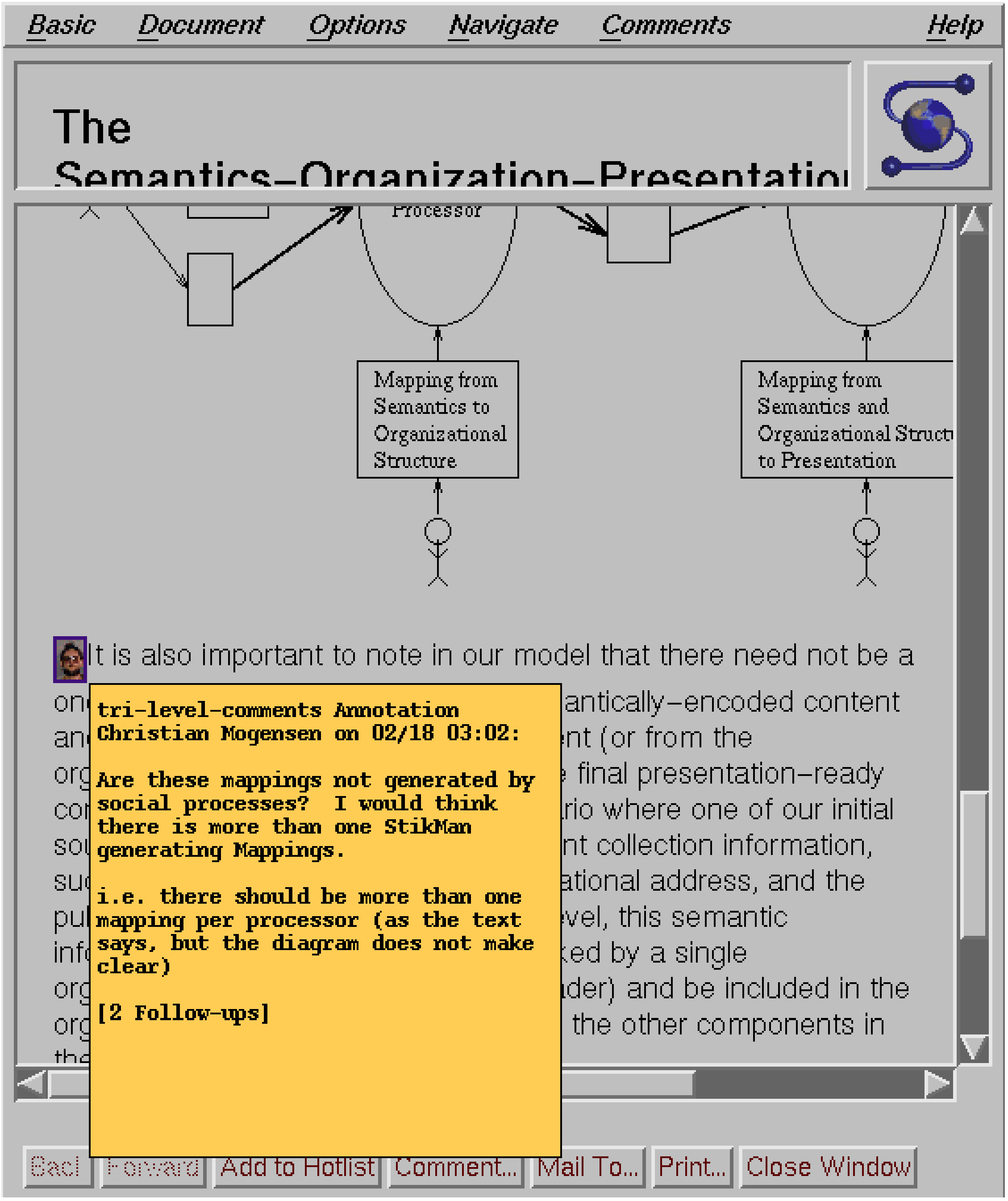
observation that there is a class of documents and document collections which can be characterized as having a large degree of recurrent and computable organizational structure. Furthermore, it is often the case that determination of the organizational structure for documents within these collections is not simply done once and for all, but rather is the result of an ongoing creative process. Examples of document collections which might fall into this class include university course catalogs, informational literature on company products, and collections of product reviews. By promoting organizational structure to a first class citizen in a conceptual document model, we believe that we are creating a formal framework in which the creation and editing of organizational structure independently from the creation and editing of semantic content and presentation characteristics can be made explicit. We point out that this distinction is not necessarily useful for all documents, just as the semantics–presentation distinction is not useful for all documents either. For example, the organization of an essay might be too intimately tied to its semantics to be able to make a clean separation between the two dimensions. However, with the aforementioned class of documents in mind, we believe that our model is relevant and well–suited to many real–world examples.

Next: The Semantics–Organization–Presentation Document **Up:** A Conceptual Model for **Previous:** A Conceptual Model for

Figure 4.1. Annotations in–lined in the document returned to the user.

The popup viewer used for quick inspection of comment

Note that any text associated with annotation is highlighted


Basic
Document
Options
Navigate
Comments
Help
The
Semantics–Organization–Presentatio
Processor
Mapping from Semantics to Organizational Structure
Mapping from Semantics and Organizational Structu to Presentation
Bacl
Forward
Add to Hotlist
Comment...
Mail To...
Print...
Close Window

It is also important to note in our model that there need not be a
on antically–encoded content
an nt (or from the
org final presentation–ready
co rio where one of our initial
so nt collection information,
su ational address, and the
pu vel, this semantic
inf ked by a single
org der) and be included in the
org the other components in
th

tri–level–comments Annotation
Christian Mogensen on 02/18 03:02:

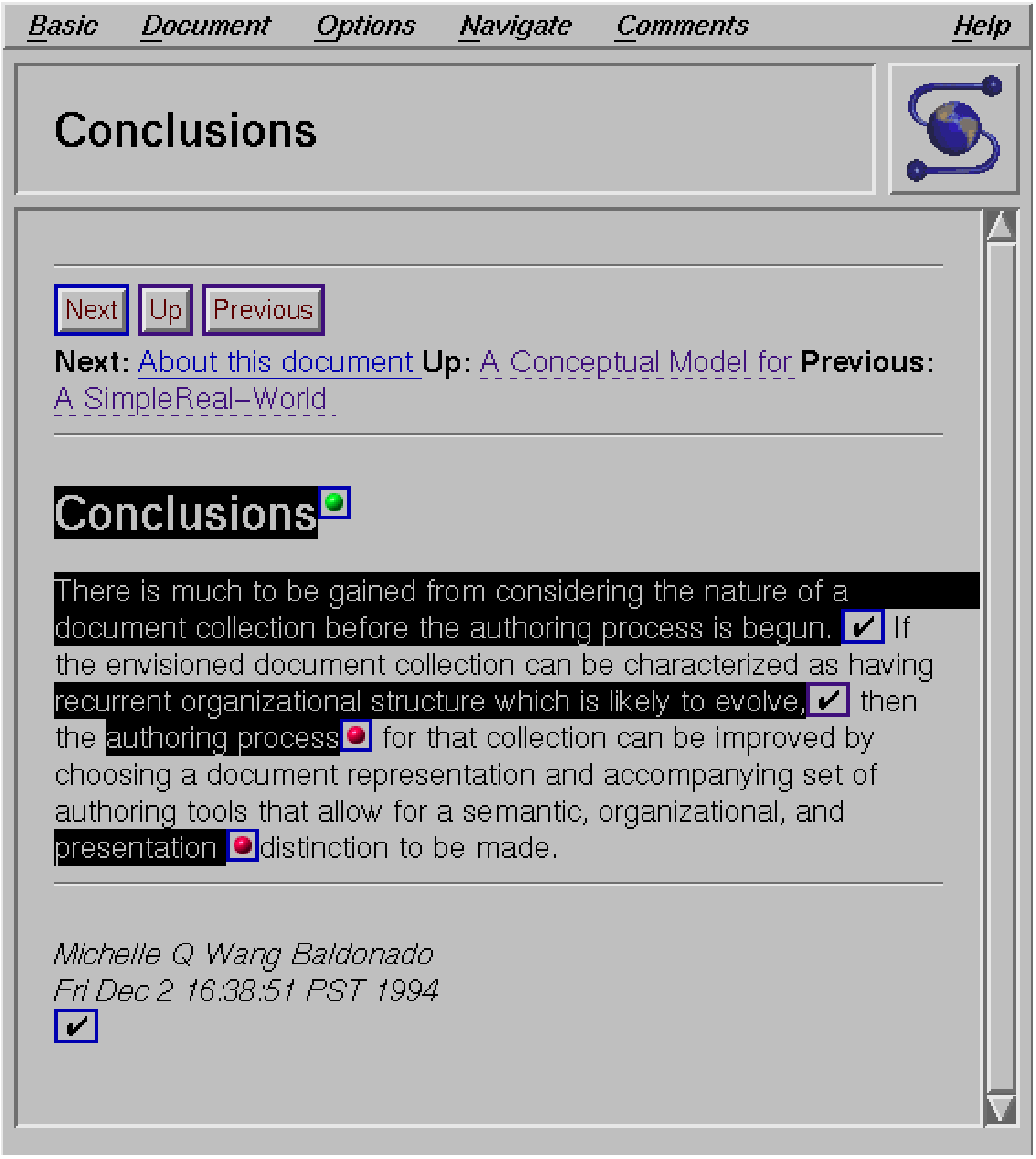
Are these mappings not generated by social processes? I would think there is more than one StikMan generating Mappings.

i.e. there should be more than one mapping per processor (as the text says, but the diagram does not make clear)

[2 Follow–ups]

Figure 4.2. Pop–up Annotation in ComMentor.

Comments can have highlighting turned on and off.
Here we can see the sets but not the individuals.
🟢:SampleSet ✓ :Tri-level-Comments and 🔴:PaperComments



Conclusions

Next Up Previous

Next: About this document **Up:** A Conceptual Model for **Previous:** A SimpleReal-World

Conclusions 🟢

There is much to be gained from considering the nature of a document collection before the authoring process is begun. ✓ If the envisioned document collection can be characterized as having recurrent organizational structure which is likely to evolve, ✓ then the authoring process 🔴 for that collection can be improved by choosing a document representation and accompanying set of authoring tools that allow for a semantic, organizational, and presentation 🔴 distinction to be made.

Michelle Q Wang Baldonado
Fri Dec 2 16:38:51 PST 1994
✓

Figure 4.3. Highlighted Text in ComMentor.

4.3 Summary

4.3.1 Strengths

The ComMentor system was especially well suited to digital libraries containing materials that might also want to capture annotations that are not part of the original document, or archives that want to provide more than one logical path through the collection (such as in the case of providing specialized tours through an art museum's digital collections based on topic, medium, artistic school, etc.) The fact that the object specifications could be passed either as MIME types or through HTML meta tags meant that overhead could be kept fairly minimal. Allowing for documents and annotations to reside on different servers without explicit knowledge of the creators of those materials allows for one document to be annotated by different groups according to their specific needs.

4.3.2 Weaknesses

The fact that the document synthesis module was coupled with the browser would have required the user to obtain the specially modified Mosaic browser in order to view ComMentor annotations. However, this implementation method represented the state of the art in the pre−Java web and illustrates many of the concepts that would later show up in Multivalent Documents. ComMentor was also not designed to handle any kind of security issues, such as terms and conditions or authentication.

4.3.3 Future Directions

Although ComMentor is not currently under development, the website can still be accessed at http://hci.stanford.edu/commentor/.

5.0 Cryptolopes

5.1 Overview

Cryptolopes (cryptographic envelopes) are self−contained structures that combine digital content, descriptive and administrative metadata, and authentication materials. Implemented as part of a 3−tiered architecture, cryptolopes offer secure distribution of digital content over non−secure channels. Along with clearinghouses and publishers, cryptolopes comprise an architecture that permits superdistribution of digital content without loss of control over materials. By protecting materials at the document level, the cryptolope system attempts to obviate the problems and cumbersome procedures common to digital content secured at the channel level−difficult distribution, repeated user authentication, labor−intensive rights management. And by integrating digital signatures and digital watermarks, cryptolope developers offer strong authentication and rights management. Although IBM developed cryptolopes in the context of e−commerce applications, its developers have subsequently integrated the cryptolope system into the IBM Digital Library package.

5.1.1 Development History

Created at IBM, cryptolopes were proposed by Jeff Crigler and Marc Kaplan. In the context of IBM's InfoMart, Crigler and Kaplan recognized the need for automated rights management of digital content. InfoMart acted as a clearing house for digital content. Organizations submitted content to InfoMart, which IBM managed, collecting fees for content use and negotiating access. Crigler and Kaplan worried that managing such a clearing house manually would not scale well. Cryptolopes were proposed as a remedy to this dilemma.

An initial prototype of cryptolopes was implemented by Kaplan, Josh Auerbach and Chee−Seng Chow in 1995. Kohl, Lotspiech, and Kaplan (1997) continued to develop the system. In 1996 IBM licensed Xerox PARC's Digital Property Rights Language (DPRL) (Ramanujapuram & Ram, 1998) for use in Cryptolopes, enhancing the system's ability to automate rights management transactions. A 1997−98 series of articles in D−Lib Magazine described the completed cryptolope architecture to the digital library community in a variety of applications (Gladney, 1997; Gladney, Mintzer & Schiattarella, 1997; Gladney & Lotspiech, 1997; Herzberg, 1998; Gladney, 1998; Gladney & Lotspiech, 1998).

5.1.2 Motivation

The goals of the cryptolope system are described in Kohl, Lotspiech & Kaplan (1997), who articulate four primary concerns:

- Authentication of the publisher and the reader
- Authentication of the content
- Reader privacy
- Superdistribution

Cryptolopes attempt to automate the negotiation between publishers and readers of digital content. Its developers argue that automating rights management improves content distribution by scaling well and by removing the administrative burdens incurred under manual systems. Under the rubric of automatic rights management cryptolopes implement *superdistribution* of digital content. Superdistribution, defined in Mori & Kawahara (1990), holds that given an abundance of cheap media (CDROM, DVD, the Internet) creators of digital content should have the ability to distribute their materials liberally without a concomitant loss of control over them. A company that permits users to sample their software by offering a downloadable free version with limited functionality, for instance, engages in superdistribution. To make superdistribution viable, distributors must decouple distribution and rights management. That is, superdistribution depends on a mechanism for enforcing terms and conditions of use throughout the life−cycle of a digital object.

Cryptolopes enable superdistribution by protecting content at the document level. Many security architectures (e.g. SSL and HTTPS) provide a secure channel through which protected information is passed in an encrypted state. However, on both the client and server sides, this information is decrypted. Thus once information is acquired over the channel, its owner may copy and redistribute it without permission of the distributor. Cryptolopes, on the other hand, hide digital content within an opaque container. Access to information inside this container is controlled cryptographically. Cryptolope creators specify terms and conditions of use, which users must satisfy in order to gain access to the Cryptolope's content. Once a user does satisfy the creator's terms and conditions, he or she is able to use the content in a controlled environment (described below), thus ensuring that the content remains under protection.

5.2 Analysis

A Cryptolope consists of several distinct parts, bundled into a single container (Cryptolopes are implemented as Java .jar files). Each Cryptolope contains:

- Encrypted content
- Non–encrypted (clear) metadata
- Authentication materials

The creator of a Cryptolope may include content in any format. A typical Cryptolope might contain the same information stored in several formats, or several different versions of a document, each of which is subject to unique terms and conditions. This content is encrypted. When a user attempts to open a Cryptolope, he is presented with the non–encrypted metadata. This provides information about the contents of the package and about the terms and conditions that he must satisfy to gain access to them.

If the user decides to "purchase" some or all of the Cryptolope's content, the Cryptolope system connects to a so–called clearinghouse. Once connected with a clearinghouse, the user may satisfy the terms and conditions expressed in the Cryptolope. Having fulfilled these, the clearinghouse delivers a cryptographic key to the user's system. This key is then used to decrypt the appropriate subset of the Cryptolope's content.

5.2.1 Cryptolope System Architecture

Kohl, Lotspeich & Kaplan (1997) describe a three–tiered Cryptolope architecture. The elements of this system are:

- Builder software
- Clearinghouse servers
- Client player/opener

Software for creating Cryptolopes is currently included in the IBM Digital Library package. This allows content distributors to store information in Cryptolopes, and to associate content with terms and conditions of use, expressed in machine–readable format via the Digital Property Rights Language.

Clearinghouses enable superdistribution by providing a platform on which to negotiate intellectual property transactions. Having secured content inside Cryptolopes, clearinghouses act as gate–keepers, providing appropriate access to protected information. IBM's InfoMart constitutes such a clearinghouse. Gladney et al. (1997) imagine digital libraries fulfilling the clearinghouse role.

Finally, the client application fulfills two roles in the Cryptolope architecture. First, it allows the end–user to interact with a clearinghouse, prompting him to respond appropriately during any transactions. Second, the client actually displays any content requested by the user. Since content may ostensibly take any format, the role of the client is non–trivial. IBM's literature describes the Cryptolope client as a plug–in for web browsers. However, the versions sampled for this study (see below) ran as applications in their own right.

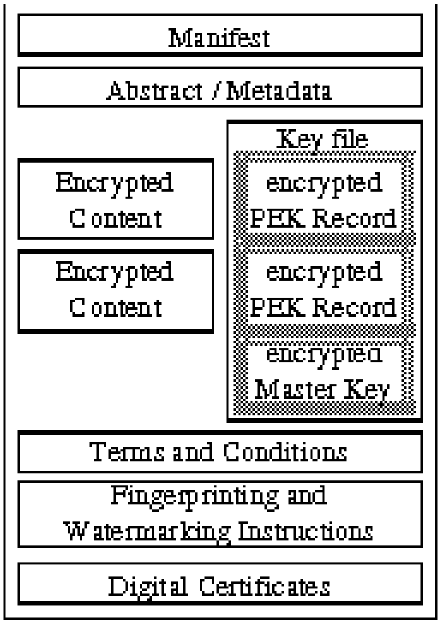


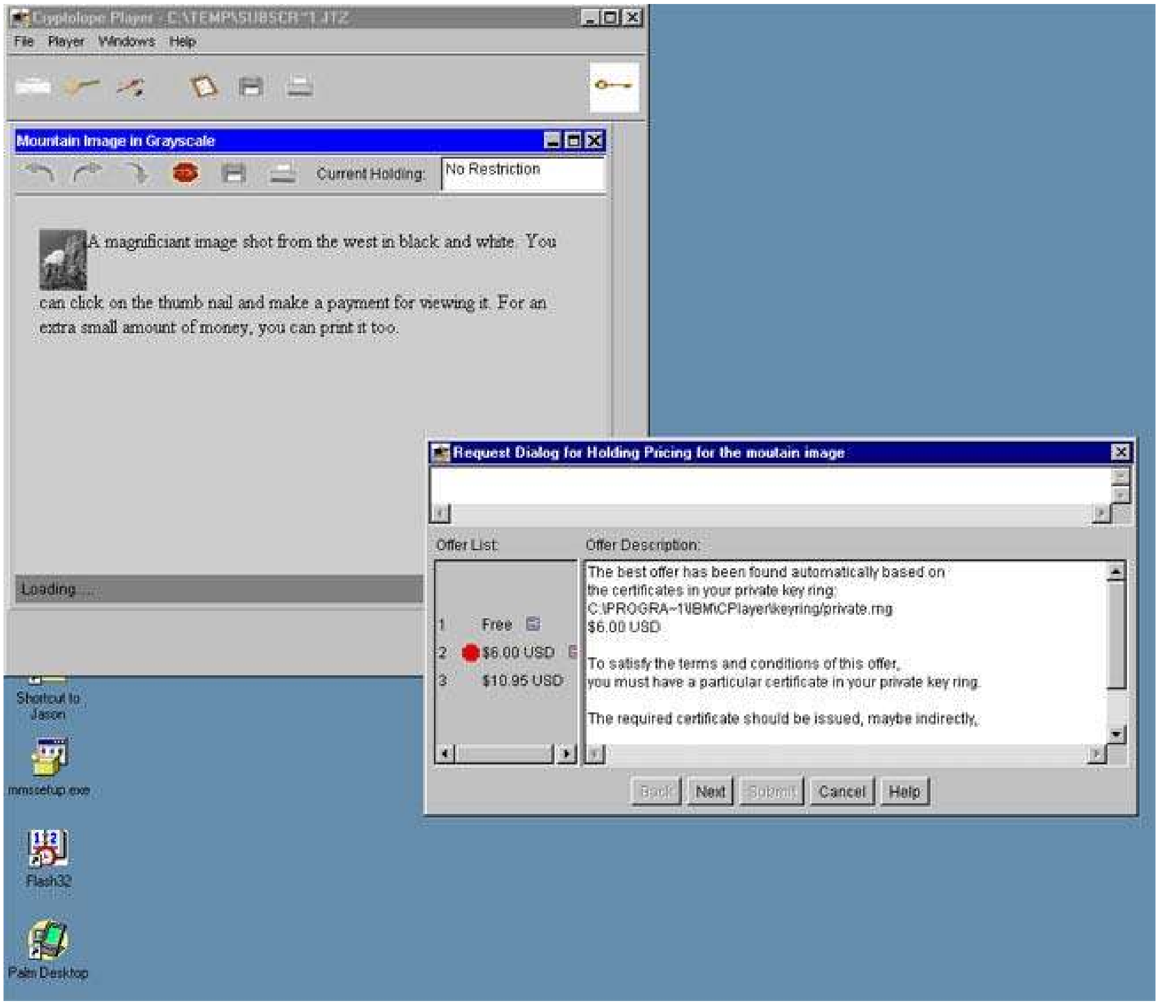
Figure 5.1. Cryptolope Architecture (from Kohl, Lotspiech & Kaplan (1997)).

Figure 5.1 shows the logical structure of a Cryptolope. Each Cryptolope contains a variety of non–encrypted information. The Manifest describes the contents of the Cryptolope. The section labeled Abstract/Metadata describes the Cryptolope in human–readable form. This section might include "teasers," portions of non–encrypted content, combined with instructions for acquiring access to the encrypted data. A Cryptolope also contains a variety of decryption keys. Each content block possesses its own decryption key. To prevent unauthorized use, these keys are themselves encrypted with a Master key. When the user fulfills terms and conditions of use, the clearinghouse delivers the client a key that is used to decrypt the Master key. Once this is accomplished, the other keys can be decrypted using information native to the individual Cryptolope. This is important because it bears on the deployment of clearinghouses. Because the clearinghouse need only decrypt Master keys, the number of keys stored by clearinghouses is low.

The other elements of a Cryptolope pertain to the system's verification and authentication functions. These functions are of two types, one of which protects end–users, and one of which protects content publishers. Each Cryptolope contains a series of checksums that are used to ensure the completeness and authenticity of the attached content. Thus when a user purchases information, the system verifies that he is in fact getting what he believes he is getting. To protect publishers, Cryptolopes also contain digital watermarking and fingerprinting instructions. When content blocks are decrypted, they are stamped with watermarks and fingerprints. Digital watermarks describe the provenance of a digital object. Thus if a user managed to create illicit copies of decrypted data, they would contain proof of their origin with the publisher. Digital fingerprints are unique to a given client's copy of Cryptolope content. Thus illegal copies of information will contain identifying information about their source.

5.2.3 Sample Operation

To sample the Cryptolope system we installed the Cryptolope Player version 1.1 (http://www–4.ibm.com/software/security/cryptolope/downloads.html). IBM's documentation indicates that the player runs under Win32 platforms. However, our installation was successful only under Windows NT (Windows 98 and Windows 2000 Professional failed). Screen captures from subsequent trials are shown in Figures 5.2 and 5.3. These figures derive from two example scenarios posted on IBM's Cryptolope website (http://www–4.ibm.com/software/security/cryptolope/casestudy.html). One treats authentication and verification, while the other highlight's Cryptolopes' rights management capabilities.


Cryptolope Player - C:\TEMP\SUBSCR~1.JTZ
File
Player
Windows
Help
Mountain Image in Grayscale
Request Dialog for Holding Pricing for the moutain image
Loading....
Shortcut to Jason
mmsetup.exe
Flash32
Palm Desktop
Back
Next
Submit
Cancel
Help

Current Holding: No Restriction

A magnificiant image shot from the west in black and white. You can click on the thumb nail and make a payment for viewing it. For an extra small amount of money, you can print it too.

Offer List:

1 Free
2 ● $6.00 USD
3 $10.95 USD

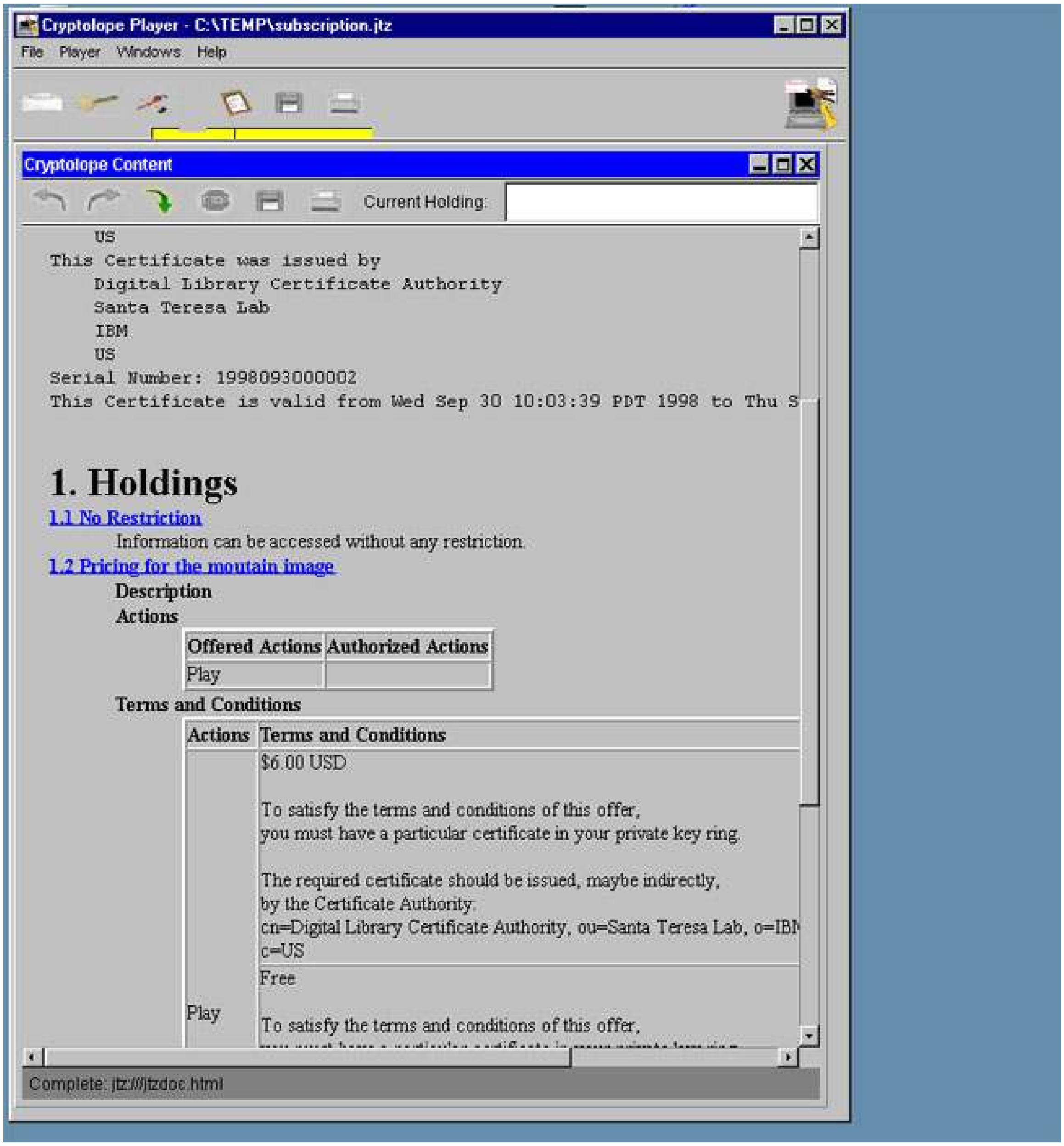
Offer Description:

The best offer has been found automatically based on
the certificates in your private key ring:
C:\PROGRA~1\IBM\CPlayer\keyring/private.rng
$6.00 USD

To satisfy the terms and conditions of this offer,
you must have a particular certificate in your private key ring.

The required certificate should be issued, maybe indirectly,

Figure 5.2. Requesting payment before viewing a large image.


Cryptolope Player - C:\TEMP\subscription.jtz
File Player Windows Help
Cryptolope Content
Current Holding:
Complete: jtz:///jtzdoc.html

```
    US
This Certificate was issued by
    Digital Library Certificate Authority
    Santa Teresa Lab
    IBM
    US
Serial Number: 1998093000002
This Certificate is valid from Wed Sep 30 10:03:39 PDT 1998 to Thu S
```

1. Holdings

1.1 No Restriction

Information can be accessed without any restriction.

1.2 Pricing for the moutain image

Description

Actions

Offered Actions	Authorized Actions
Play	

Terms and Conditions

Actions	Terms and Conditions
Play	$6.00 USD To satisfy the terms and conditions of this offer, you must have a particular certificate in your private key ring. The required certificate should be issued, maybe indirectly, by the Certificate Authority: cn=Digital Library Certificate Authority, ou=Santa Teresa Lab, o=IBI c=US
	Free To satisfy the terms and conditions of this offer,

Figure 5.3. Examining the manifest of a Cryptolope.

5.3 Summary

5.3.1 Strengths

Cryptolopes are potentially useful for administrators of digital libraries in several regards. Their ability to provide persistent rights management for digital objects is highly desirable for many digital libraries where intellectual property concerns are serious. Not only do Cryptolopes enable superdistribution; they also permit administrators to automate rights management transactions, and they enforce terms and conditions at any level of granularity.

5.3.2 Weaknesses

The proprietary viewing software that readers must use to purchase and make use of Cryptolope content is troublesome. In addition to its failure to run on several supported platforms, this software runs the risk of obsolescence. Providers of digital content might be uncomfortable delivering their information in a closed format that is not supported by major standards bodies. Recent lack of activity on the Cryptolope website, and the slow pace of recent Cryptolope publications serve to reinforce such misgivings.

5.3.3 Future Directions

We are unsure of the current status and future direction of the Cryptolope project. There has been little activity at http://www−4.ibm.com/software/security/cryptolope/ and it is possible that the technology has been folded into other IBM DL products.

6.0 Digibox

6.1 Overview

Like IBM’s Cryptolopes, InterTrust’s Digibox technology is intended to allow secure distribution of digital content over open channels. To accomplish this, Digiboxes combine encrypted content and rights management "controls" in an opaque container. Part of InterTrust’s larger MetaTrust framework, Digiboxes allow peer−to−peer enforcement of "distributed electronic contracts" (Sibert, Bernstein & Van Wie, 1995), permitting content providers and distributors to maintain control over terms and conditions of use throughout the lifecycle of a digital object. InterTrust has succeeded in licensing it’s MetaTrust technologies (including Digiboxes) to such companies as Adobe, Nokia, and America Online. Support from major software vendors, and InterTrust’s continued influence with standards bodies such as MPEG and SDMI suggests that Digiboxes are gaining sufficient market saturation to enhance their appeal to administrators of digital collections.

6.1.1 Developers

The Digibox architecture was first described by Sibert, Bernstein & Van Wie (1995). At that time, Sibert and his co−authors were at Electronic Publishing Resources, Inc. During subsequent years research continued within InterTrust’s STAR Labs. In 1997, researchers at STAR Labs published Sibert, Horning & Owicki (1997), which elaborates InterTrust’s digital rights management (DRM) architecture, in which Digiboxes play a key part.

6.1.2 Motivation

In the 1995 description of Digiboxes, the stated goal was to support "information commerce." By "information commerce" they refer to trade in electronic information, such as data from databases, articles, music, video, etc. They define three goals for any software purporting to enable such trade. Using this software:

1. Information providers can be assured that their content is used only in authorized ways;
2. Privacy rights of users of content are preserved;
3. Diverse business models related to content can be electronically implemented

Due to the openness of distributed environments, issues of copyright and other rights management matters plagued early information commerce architectures. Moreover, these early systems limited the viable business models available for information vendors.

Like the developers of Cryptolopes, the Digibox developers hoped to enable superdistribution of digital content. The same openness that made distributed environments difficult for information trade also made the distribution of information cheap and easy. The goal, then, was to decouple the information itself from the rights to the information. Digiboxes accomplish this by making content usable only in accordance with control directives, specified by the Digibox's creator. Thus by giving content−rich Digiboxes to a wide market, vendors do not correspondingly give away control of their intellectual property. By charging customers for the rights to use information rather than charging for possession of the information itself, and by affording providers flexibility in expressing rights management rules, Digiboxes permit a wide assortment of information commerce business models.

6.2 Analysis

Digibox is "implemented in a set of platform−independent class libraries that provide access to objects in the container and extensions to OpenDoc and OLE object technologies" (Sibert, Bernstein & Van Wie, 1995) Encryption is available via triple DES or RSA algorithms. For creating Digiboxes, InterTrust sells several products (http://www.intertrust.com/main/products/) with a variety of features. Of particular interest for digital library administrators are two products: DocBox, a tool for secure publishing using PDF format; and Flying Media, a suite of tools for delivering multimedia content. Both DocBox and Flying Media run under 32−bit Windows. They require Intel architecture. Since their recent agreement with Adobe Systems, InterTrust's Digibox software is also available to users of Adobe Acrobat. Content publishers can create secure PDF files using Acrobat. And end−users can use Acrobat Reader to access Digibox protected content through the MetaTrust system.

6.2.1 Architecture

6.2.1.1 The MetaTrust Utility

Digiboxes are delivered within the larger framework that comprises InterTrust's MetaTrust utility. Designed to be a trusted platform on which business can reliably conduct information commerce, MetaTrust consists of a network of "InterTrust points." An InterTrust point is a computer system running one of the InterTrust products. Thus content creators use MetaTrust–compliant authoring software to create Digiboxes. Next, distributors aggregate Digiboxes and disseminate them. Using MetaTrust viewers and players, customers can choose to acquire Digibox content by agreeing to the creator's and distributor's terms and conditions. Users may also forward Digiboxes to other users, who will be automatically subject to the requisite terms and conditions of use, due to the integrity of the Digibox. At periodic intervals each end–user's MetaTrust client communicates with an available MetaTrust clearinghouse, transmitting his or her recent MetaTrust transactions for processing. Finally, the clearinghouses process user transactions and deliver output (such as payment or user data) to the appropriate parties, including InterTrust itself.

Each party in this process–creators, distributors, users, and clearinghouses–uses InterTrust software. Together they comprise the MetaTrust utility, a network of so–called InterTrust points.

Points in the MetaTrust architecture communicate one another in a peer–to–peer relationship. For instance, if user *A* forwards Digibox *D* to user *B,* user *B* does not need to communicate with any third party to gain access to *D* (provided he already has Digibox–compliant viewing software). If user *B* fulfills the terms and conditions of *D,* the record of his having done so will be transmitted to another InterTrust point in a batch process at a later date. Thus the fulfillment of terms and conditions, and the user's access to digital content are handled in a completely automatic fashion, with no intermediaries.

6.2.1.2 Digibox Structure

Each Digibox contains 0 or more of the following:

- *Content Properties.* A Digibox may contain content in a variety of formats (e.g. PDF, HTML, OLE). However, due to the opacity of the Digibox container these content element are useless without the user's system being told explicitly how to access them. To gain such access, the user must comply with the Digibox creator's terms and conditions.
- *Controls.* Controls express terms and conditions of use in a machine–readable format. They may refer to content properties contained within a shared Digibox, or they may refer to content located in another Digibox.

Controls provide flexible control over digital content. They may express a wide variety of necessary conditions of use, or they may define detailed types of use that correspond to the fulfillment of conditions. Likewise, they are flexible with regard to scope. Digibox controls may govern aggregated content of various media. On the other hand, they may control portions of a single file (Digibox controls can specify directives at the byte–level).

Because controls dictate which content properties a client sees, and in what manner they see it, owners of content can distribute information freely, charging for or otherwise restricting only the rights to use the information.

6.2.1.3 Protected Local Storage

Although not strictly part of the Digibox itself, each machine capable of viewing cryptolopes must run InterTrust software in an area of "protected local storage." This is an opaque application that negotiates the transactions initiated by the user's interaction with Digibox controls. The protected local storage unit contains a variety of cryptographic keys that are used to decrypt data and metadata according to the directives specified by the control directives (see discussion of cryptographic protection below).

Because protected local storage contains the keys necessary to gain access to Digibox content, it is emphasized that Digibox security is "only as strong as the tamper–resistance of the local processing environment" (Sibert, Bernstein & Van Wie, 1995). For highly valuable content, where the risk and cost of theft or fraud is high, InterTrust recommends that the software be run on secure hardware (a secure processing unit, or SPU). However, for settings where the aim of encryption is deterrence of fraud rather than categorical prevention, a traditional CPU may suffice.

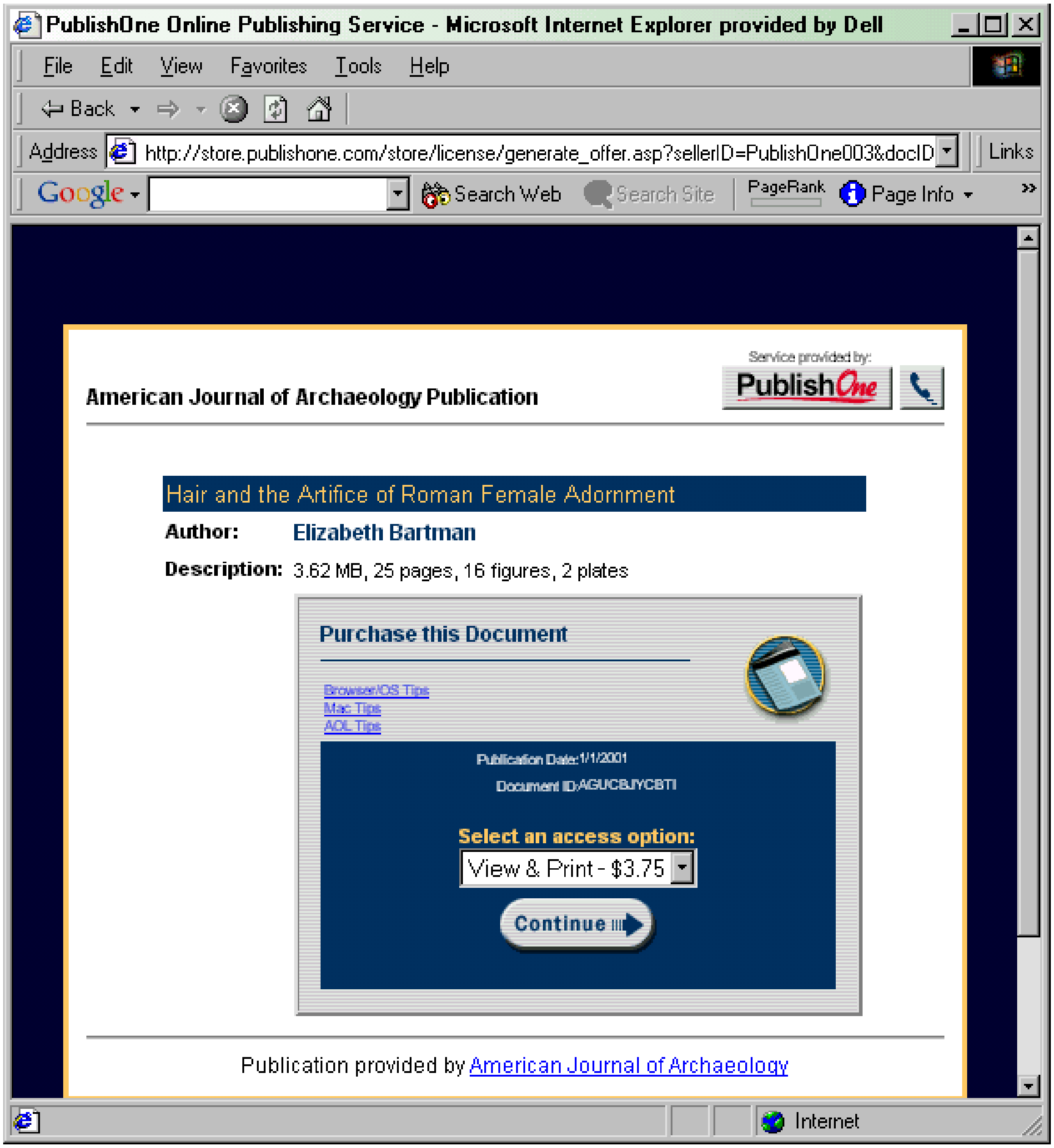
6.2.1.4 Cryptographic Protection

Depending on the level of security desired by its creators any combination of a Digibox's content properties and controls can be encrypted. For those delivering highly valuable content, perhaps all Digibox components merit protection. Other applications might call for more modest, lightweight encryption. But regardless of additional protection, each Digibox is relatively secure by virtue of its high–level information, which is encrypted. Part of the Digibox's organizational header, all of the Digibox controls, and any "content keys" needed to decrypt encrypted content properties are delivered in encrypted form. This encrypted information is unlocked by means of a "transport key block," which is derived by a combination of keys stored in the Digibox itself and in the local machine's protected storage area. Thus all access to a Digibox takes place in communication with the InterTrust protected software.

While this architecture provides robust enforcement of publisher's terms and conditions of use, it also prevents the delivery of altered or otherwise inauthentic content. In addition to supervising decryption of controls and content, a Digibox's encrypted high–level information includes checksums used to verify the integrity of content. Thus purchasers of Digibox information can be assured of the provenance of their information.

6.2.2 Sample Operation

To test the Digibox system, we purchased an article from an online publisher that uses InterTrust to deliver protected information. The American Journal of Archaeology (http://www.ajaonline.org) sells access to its publications using InterTrust software. Users may purchase PDF versions of recent AJA articles through PublishOne (http://www.publishone.com), a subsidiary of InterTrust. Once he or she has created a PubishOne account, users may easily purchase, view, and print articles using Adobe Acrobat Reader (version 4.05). AJA articles cost $0.15 per page. This purchase entitles the customer to view an article on three different workstations. Customers who wish to email an article to colleagues are free to do so. When recipients attempt to read an attached Digibox, they will be prompted to purchase appropriate viewing rights. The policies derive from AJA, not PublishOne. They are described at http://www.ajaonline.org/shared/s_info_ecommerce.html. Figures 6.1 and 6.2 show steps in the process of purchasing an article from the AJA website.



http://store.publishone.com/store/license/generate_offer.asp?sellerID=PublishOne003&docID

American Journal of Archaeology Publication

Hair and the Artifice of Roman Female Adornment

Author: Elizabeth Bartman

Description: 3.62 MB, 25 pages, 16 figures, 2 plates

Purchase this Document

Publication Date:1/1/2001

Document ID:AGUCBJYCBTI

Select an access option:

View & Print - $3.75

Publication provided by American Journal of Archaeology

Figure 6.1. Attempting to open a Digibox PDF file.

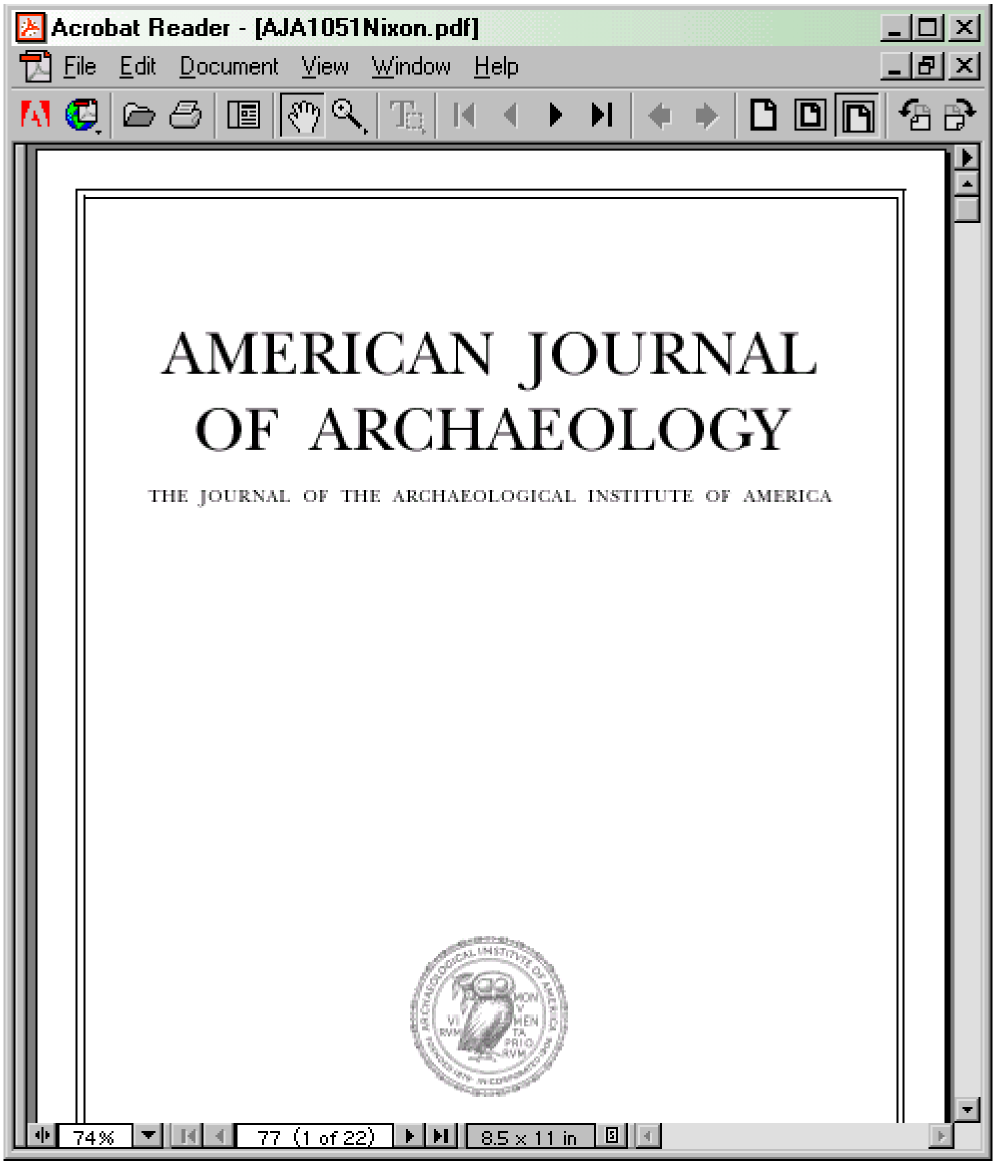

Acrobat Reader - [AJA1051Nixon.pdf]
File Edit Document View Window Help
AMERICAN JOURNAL
OF ARCHAEOLOGY
THE JOURNAL OF THE ARCHAEOLOGICAL INSTITUTE OF AMERICA
74%
77 (1 of 22)
8.5 x 11 in

Figure 6.2. The protected PDF file downloaded to the personal computer.

6.3 Summary

6.3.1 Strengths

Protecting objects at the document level, rather than at the channel level, affords a great deal of flexibility in the delivery of digital content. This flexibility is enhanced by the adjustable granularity of Digibox controls. Rights management directives in a Digibox may apply to groups of content properties, or to individual bytes within a single property.

Also attractive is InterTrust's successful integration of Digibox software into mainstream software such as Adobe Acrobat. By reducing dependence on proprietary software, InterTrust has provided a flexible, convenient way to manage digital content for superdistribution.

6.3.2 Weaknesses

While avoiding proprietary viewing/playing applications (such as those used in Cryptolopes) increases the chance of market acceptance, creating PDFs, MPEGs and other file with built–in protections raises serious issues regarding the long–term preservation of documents. The preservation of the AJA articles, for example, is now fundamentally bound to the economic well–being of InterTrust.

6.3.3 Future Directions

With a variety of business partners and alliances and frequent activity on their website (http://www.intertrust.com/), InterTrust appears committed to the continued development of their Digibox technologies.

7.0 Document Management Alliance

7.1 Overview

The Document Management Alliance (DMA) is a coalition of users and developers of document management (DM) software. Organized by the Association for Information and Image Management (AIIM), DMA's goal is to improve interoperability between DM systems. Towards this, DMA members defined DMA specification 1.0 (DMA Technical Committee, 2001), a standard API for document management. With its roots in issues of workflow management, DMA 1.0 aims to permit easy integration of DM systems, allowing distributed collaboration, cross–repository search and retrieval of documents, and complex document protection. In addition to enabling DM software interoperability, DMA 1.0 comprises a flexible, rich document model.

DMA is similar to other projects, notably ODMA and WebDAV. The Open Document Management API (ODMA) is the outcome of another AIIM task force. ODMA is similar to DMA insofar as it enables some interoperability between document management software. However, the reach of ODMA is more modest than that proposed by DMA. Web Distributed Authoring and Versioning (WebDAV) is an ongoing project from the IETF WebDAV working group (http://www.webdav.org). WebDAV is an extension to HTTP that permits collaboration and versioning of documents via web servers.

7.1.1 Developers

AIIM (http://www.aiim.org) is a trade organization in the document management and workflow systems industry. The DMA is an AIIM task force. This task force is composed of DM software users and developers, whose motivations represent a wide variety of document management tasks. A sampling of DMA member organizations includes IBM, Xerox, Novell, Documentum, Interleaf, Oracle and many other significant companies with an interest in DM systems.

7.1.2 Motivation

According to the DMA 1.0 specification, the goals of DMA are:

- Uniform access to any document, anywhere in an enterprise
- Self−describing systems and documents for ease of setup and configuration
- Scalable document management solutions from legacy systems to fully−featured, state of the art document management systems
- Expanded collaboration opportunities
- High−level integration of services and applications

Many organizations use a variety of document management software. The DMA authors argue that in these circumstances, incompatibility between DM systems results in an "islands of information" problem; information in one document repository is not visible or accessible to users of another repository. DMA−compliant software would address this problem by defining self−describing objects. Such objects−documents, services, systems, etc.−would allow diverse DM systems to exchange information easily. Also crucial for modern DM software is the ability to manage collaboration in document editing. The DMA specification aims to improve not only inter−repository search and retrieval, but also supports locking and versioning across document spaces. Because document management software is used in a wide variety of settings, the DMA model is intended to be scalable: adhering to the specification would help systems address the needs of small organizations or large government agencies.

7.2 Analysis

Section 2.4.1 of DMA 1.0 defines a three−tiered architecture for DMA−compliant systems. These systems include:

- Client Applications
- DMA Middleware
- DMA Service Providers

Using client applications end−users create and edit documents, search document repositories, organize document collections, etc. DMA service providers are DMA−compliant document management servers. These systems provide a variety of functionality, such as file–system operations. The DMA specification likens each DMA service provider to a single library in a larger library system. That is, each service provider–or "document space"–handles access to its own materials. Finally, the DMA middleware negotiates between applications and document spaces. For instance, the middleware provides search and retrieval services across multiple document spaces. Thus the DMA authors compare the middleware to the administration of a library system.

Communication between DMA clients, middleware, and service providers is enabled by the DMA API, which is based on several data abstractions defined in DMA 1.0 section 3. These abstractions are:

- DMA object model
- DMA interface and process model
- DMA integration model
- DMA distribution model
- DMA query model
- DMA containment model
- DMA content model
- DMA versioning model
- DMA internationalization and localization model

The DMA object model provides uniform access to data and services in DM systems. Exposing data and services via standard methods, the object model shields users and system designers from implementation details of individual DM systems. DMA's interface is "a strict, portable subset of Microsoft's COM (Component Object Model)" [DMA 1.0 3.2.1], which lends the system its encapsulation of objects. According to section 3.3.1, the "Integration Model defines how DMA components are assembled." Queries across document spaces are handled by the DMA query model. This component of the DMA system allows administrators of individual document spaces to define the data and methods of search available for their collection. The DMA Content Model allows compliant applications to access and modify document content. Particularly important to the Content Model is DMA's ability to support complex documents. These are documents composed of multiple versions or "renditions." The DMA Content Model allows DM software to manage collaborative editing and display of evolving or otherwise multifaceted documents. Documents that are being revised constitute a special class of complex documents. The management of these documents is supported by the DMA Versioning Model, which supports so–called "linear versioning." Thus a conceptual entity, the Document, is assumed to be represented by a consecutive series of individual document renditions.

7.2.1 DMA Classes

The DMA object model enables communication between document spaces of a DMA system by imposing a unified structured on the system's data and services. Described at length in DMA 1.0, Section 3.1, the majority of data in a DMA object is stored as a series of *properties.* DMA classes expose their current state through a variety of *getProp* methods. Likewise, they permit manipulation through *putProp* methods. These properties may be of standard data–types, such as integers, floating point numbers, or strings. Or they may be object–valued, pointing to another DMA object. In the interest of creating self–describing objects, DMA classes contain recursive metadata structures that allow other DMA classes to probe their structure and functionality. This relationship is illustrated in Figure 7.1.

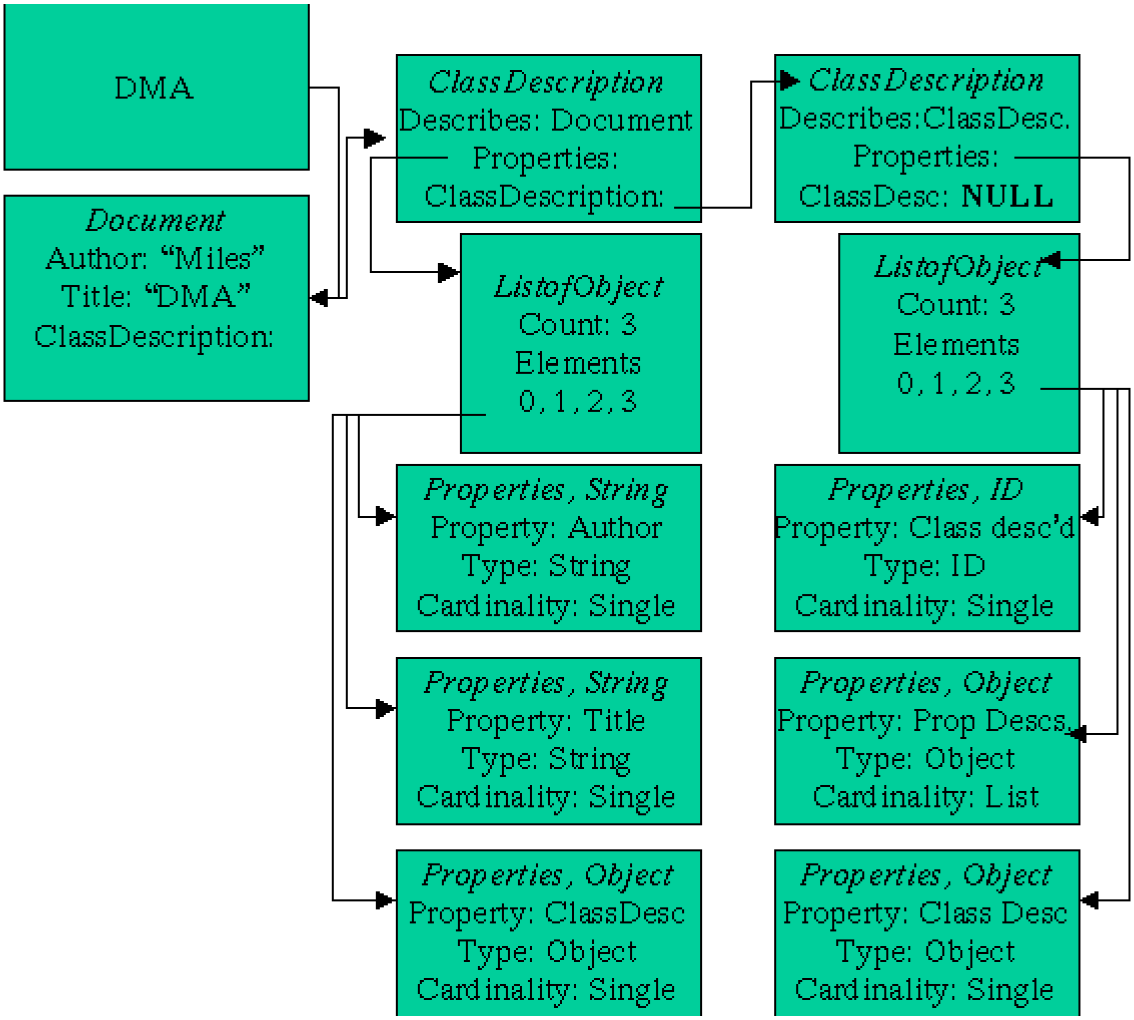


Figure 7.1. DMA Object Hierarchy

Figure 7.1 depicts an imaginary DMA class, *Document.* The *document* class has three properties: title, author, and classdescription. Title and author are both string–valued properties. However, classDescription is object–valued. Its value is itself an object of type *classDescription.* All objects in the DMA class hierarchy contain a classDescription property, which points to a *classDescription* object. This provides the requisite information for DMA objects to identify their own properties. Self–identification occurs by a recursive traversal of the object's classDescription property hierarchy. Thus in the example shown in Figure 7.1, by querying the document's classDescription property, the

DMA system can learn that the document object contains 3 properties. It can learn the datatype of each property and its cardinality. Note that the *classDescription* object does itself contain a classDescription property. However, the *classDescription* object pointed to by another classDescription has a NULL−valued classDescription property, thus averting an infinite recursion.

7.3 Summary

7.3.1 Strengths

Users of document management software incur many benefits under a DMA−compliant document management system. Most obviously, the ability to search across document spaces and the prospect of robust collaboration and versioning tools argue for the importance of DMA to document management and workflow management tasks. More generally, the document model defined in the DMA specification is flexible and precise. DMA's ability to manage complex documents (documents consisting of multiple versions) is particularly useful. For purposes of document editing, complex document support is an obvious boon. But complex documents are also useful in the common challenge of distributing a document in multiple formats (e.g. HTML, PDF, ASCII).

7.3.2 Weaknesses

Document management software vendors have little incentive to comply with the DMA specification. The proprietary protocols and formats that DMA views as a liability often secure the influence of a software vendor in an organization. As a consequence, implementation of DMA−compliant DM systems has been slow. According to the DMA website (as of 20 April 2001), "There are no commercial, fully−compliant products that can be assessed and with which interoperability can be verified" (http://www.infonuovo.com/NuovoDoc/analysis/DMA−adoption−risk.htm).

7.3.3 Future Directions

DMA Specification 1.0, Section 1.4 lists three future goals for the task force:

- Interoperability with other proposed standards (e.g. ODMA, wfMC, WebDAV)
- Additional work on DMA/Internet interoperability (e.g. Java)
- Additional support for compound documents, content−based search, security, directory service integration, transactions, and/or audit trails.

The website can be accessed at http://www.infonuovo.com/dma/.

8.0 Flexible and Extensible Digital Object and Repository Architecture (FEDORA)

8.1 Overview

The Flexible and Extensible Digital Object and Repository Architecture (FEDORA) can be defined as a new model of repository service able to store and make accessible distributed digital information of a complex, heterogeneous, and evolving nature (Payette & Lagoze, 1998; Payette & Lagoze, 2000a; Payette & Lagoze, 2000b). FEDORA is meant to serve as a fundamental component within a broader design of an open and flexible digital library architecture. The architecture model of FEDORA provides an effective mechanism for uniform access to digital objects distributed between multiple repositories. It also allows association of external rights management services with the object content. The key concepts in FEDORA represent an evolution from those in the Kahn−Wilensky Framework and the Warwick Framework (Table 8.1). It is also based on the Distributed Active Relationship (DAR) mechanism (Daniel & Lagoze,1997a; Daniel & Lagoze, 1997b; Daniel, Lagoze & Payette, 1998).

Kahn Wilensky Framework	***Extended Warwick Framework***	***FEDORA***
Data & metadata	Package	DataStream
Digital Object	Container	DigitalObject
Dissemination	DAR	Interface
Terms & Conditions	DAR	Enforcer
Repository	Container	Repository

Table 8.1. Evolution of Concepts to FEDORA (from Daniel & Lagoze (1997b)).

8.1.1 Developers

FEDORA is a DARPA funded project jointly developed by the Digital Library Research Group at Cornell University and the Corporation for National Research Initiatives (CNRI), and later with collaborators at Los Alamos National Laboratory and University of Virginia.

8.1.2 Motivation

The FEDORA project addressed the need for an advanced repository module intended to provide a reliable system for depositing, storing, and accessing digital objects. The concept of a digital object is continuously evolving toward multiple and dynamic formats and requires suitable digital library infrastructures for its management. The repository forms only the basic layer of a digital library model described as a multi−layered structure supporting a number of other services, including information searching and discovery, registration, user interface, and rights management services.

8.2 Analysis

To understand FEDORA, one must first understand the Repository Access Protocol (RAP). RAP was first defined in the Kahn−Wilensky Framework (Kahn & Wilensky, 1995), and then later refined and developed through a series of publications (Lagoze & Ely, 1995; Lagoze et al., 1995) and software implementations. FEDORA was built from the lessons learned from previous implementations and prototype deployments of systems that implemented RAP for the Library of Congress (Arms, Blanchi & Overly, 1997). It is important to note that non−FEDORA implementations of RAP are possible. CNRI distributes their own RAP−compliant software package (Digital Object Store, 2001) and interoperability between the two implementations was verified by CNRI and Cornell staff members (Payette et al., 1999).

The architecture model in FEDORA performs a series of repository functions for a digital object type. The developers of the project intend for FEDORA to provide uniform access to digital content regardless of the data format, the underlying structure, and the physical distribution of the object. Digital Objects and Repositories are the key components of the project. The Digital Objects managed in FEDORA have dynamic characteristics. They are the result of an aggregation of "multiple and compound" digital content. FEDORA is also expected to accommodate both current complex digital objects, such as multimedia objects, as well as future emerging object types.

8.2.1 Architecture

The basic architectural components of FEDORA are DigitalObjects, Disseminators, and Repositories. Extending the basic concepts of "content" and "package" from the Warwick Framework, the DigitalObject is defined as a "container abstraction" that encapsulates multiple content in packages called DataStreams. Figure 8.1 illustrates a simple DigitalObject with an access control list (ACL), MARC metadata, and a PostScript data component. The DataStreams are composed of MIME−type sequences of bytes (e.g., digital images, metadata , XML files) and operate as the basic elements of a content aggregation process. The DigitalStreams can be physically associated with the DigitalObject or they can be logically associated but stored in another DigitalObject. The DigitalObject is the result of distributed content. The mechanism that provides access to DigitalStreams is a set of service requests called PrimitiveDisseminators, identified by unique names, e.g., URN. The PrimitiveDisseminator is logically associated with every DigitalObjects and provides an interaction with the DigitalObject at the internal structural level. Other types of Disseminators, specifying behaviors for particular types of content, can be associated with DigitalObjects. They are designed for user interaction, providing recognizable views of content formats such as books, journals, or videos as a result of the client request. Figure 8.2 illustrates content−specific Disseminators for the metadata (MARC, Dublin Core derived from the MARC, and per−chapter access to the PostScript file).

The project developers use the metaphor of the cell to describe the structure of a DigitalObject. The core is composed of a nucleus containing data packages, surrounded by an interface layer containing sets of behaviors that transform the row data into information entities as requested by the user.

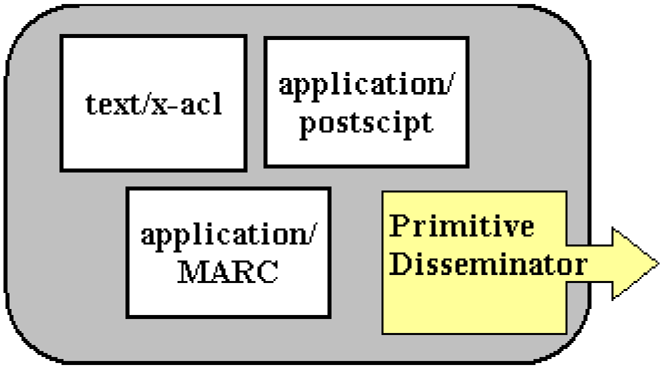


Figure 8.1 A simple DigitalObject (from Payette & Lagoze (1998)).

The Repository provides deposit, storing and access services to these DigitalObjects. These containers are opaque entities for the Repository that has no knowledge about their internal structure and content and manage them only through the unique identifiers (URNs) of the DigitalObjects. Each DigitalObject has a set of "native operations" to access and manipulate the content. They are defined by the Distributed Active Relationship (DAR). DAR provides an open interface that enables the functions of listing, accessing, deleting and creating DataStreams. DARs are the means for implementing components called "Interfaces" and "Enforcers" that are linked to DigitalObjects. "Interfaces" define behaviors to enable DigitalObjects to produce "disseminations" of their content. "Enforcers" are special types of interfaces that provide a mechanism for protecting intellectual content. The rights management security is directly applied to the DigitalObject behaviors instead of to the content, since the content, such as the internal structure, are opaque to the repository. As noted before, the DigitalObjects are identifiable only by their URNs.

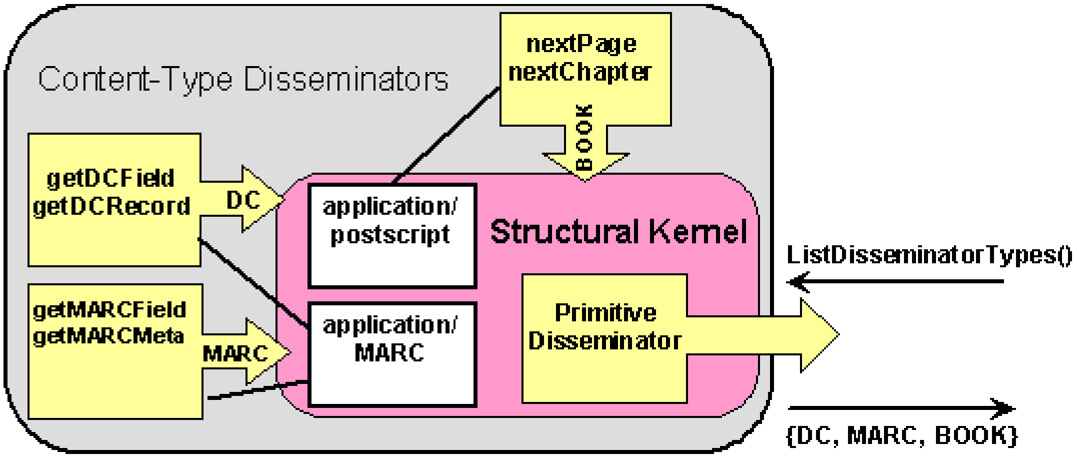


Figure 8.2. A DigitalObject with three content–specific Disseminators (from Payette & Lagoze (1998)).

8.2.2 System Requirements

The RAP protocol is defined using the Interface Description Language (IDL), a technology –independent definition language. Although non–CORBA versions of RAP have been developed, FEDORA is being implemented using Iona's OrbisWeb for Java and Visigenic's VisiBroker.

8.3 Summary

8.3.1 Strengths

FEDORA has provided valuable contributions to achieve interoperability and extensibility in digital libraries. At the structural level, the aggregation of "distributed and heterogeneous digital content" to compound complex objects is an important aspect of interoperability. At the architectural level, the DAR abstraction provides an effective tool for a uniform access to objects distributed among multiple repositories. The extensibility is ensured by the separation of the digital object structure from both the interfaces and the related mechanisms. The way of accessing the complex objects through open interfaces producing multiple outputs of content promotes both interoperability and extensibility.

The flexibility and modularity of the FEDORA architecture has been proved to be suitable to handle a variety of complex multi–level objects. In particular, FEDORA has been implemented and customized by the library research and development group at the University of Virginia for supporting the specific functionalities required by the electronic text collections of the UVa electronic text center (Staples & Wayland, 2000).

8.3.2 Weaknesses

While providing great power and flexibility, the acceptance of deploying CORBA for DL projects remains unknown. It is worth noting that the UVa implementation of FEDORA uses a relational database management system, and not CORBA.

8.3.3 Future Directions

FEDORA can be considered a key phase toward the development of open architecture digital libraries. Current research by Cornell/CNRI is focused on integrating Cornell's FEDORA with CNRI's DigitalObject and Repository Architecture in order to define a more powerful infrastructure able to achieve a higher level of interoperability and extensibility. Another area of further research is the security and access management. FEDORA has been recently chosen as "mediation architecture" for supporting "Value−Added Surrogates" for distributed content within Prism, a research project at Cornell University focused on developing access control and preservation mechanisms for distributed information systems (Payette & Lagoze, 2000b). The latest version of the RAP IDL can be found at: http://www.cs.cornell.edu/cdlrg/fedora/IDL/.

9.0 Kahn−Wilensky Framework Digital Objects

9.1 Overview

The Khan−Wilensky Framework is a theoretical infrastructure that defines a set of elements and requirements for supporting a variety of distributed information services (Kahn & Wilensky, 1995). This framework defines a few basic abstractions for an open and universal information system that have strongly influenced the development of subsequent information architectures. Kahn−Wilensky Framework addresses only structural aspects of the system, in order to avoid constraints to a future technological evolution and to insure the full interoperability of the services.

9.1.1 Developers

The Kahn−Wilensky Framework was developed in 1995 by Robert Kahn from the Corporation for National Research Initiatives (CNRI) and Robert Wilensky from the University of California at Berkeley. Their research was supported by the Advanced Research Projects Agency (ARPA) for use in the Computer Science Technical Report Project (CSTR). The initial architecture they provided was fleshed out in Arms (1995), and then partially implemented in Lagoze & Ely (1995) and Lagoze et al. (1995).

9.1.2 Motivation

Robert Kahn and Robert Wilensky addressed the need generalized system for managing a wide and extensible class of distributed information services, of which DLs represented only an example of these services. They provided the theoretical foundation of a new model of information architecture described in *A Framework for Distributed Digital Objects Services,* which became a milestone reference in the DL literature.

9.2 Analysis & Architecture

The Kahn−Wilensky framework does not provide a defined, structured architecture, but a more general, conceptual infrastructure for managing complex digital objects. They made no assumptions about implementation details. Kahn and Wilensky founded their information model on three essential components: Digital Objects, Handles and the Repository Access Protocol (RAP). All other notions are derived from these three. The Digital Object (DO) is a structured instance of an abstract data type made up of two components: data and key−metadata. The key−metadata includes the Handle, which is a mandatory and primary global identifier for the Digital Object. RAP is a technology that must be supported by all the repositories in order to accomplish the deposit and access processes of the Digital Objects. This relationship is illustrated in Figure 9.1.

The system provides a strategy for registering, depositing and accessing Digital Objects. A Digital Object is made available to the System by an Originator, individual or organization, which can deposit or access it. Every Digital Object has an Originator that defines also the terms and conditions for its use. In the registration process, the Originator requests the Handle from an authorized Handle generator. Then the Originator deposits the Digital Object in one or more Repositories. The Repositories too have unique names assigned through specific naming conventions.

Upon depositing, the Handle and the Repository name or IP address are registered with a globally available system of handle servers. Each Repository keeps a properties record for each Digital Object stored. The properties records include all the metadata of the Digital Object. RAP enables the processes of accessing and depositing Digital Objects within Repositories. Access to a Digital Object is achieved by specifying the Handle and a service request type. The data output of the access service request is a "dissemination". Although RAP is now more fully developed and expressed in an IDL (see section 8), the high−level operations of RAP, as expressed in Arms, Blanchi & Overly, are listed in Table 9.1.

RAP Commands	***Description***
VerifyHandle	Confirm that a handle has been registered in the handle system
AccessRepoMeta	Access the repository metadata
Verify_DO	Confirm that a repository stores a digital object with a specified handle
AccessMeta	Access the metadata for a specified digital object
Access_DO	Access the digital object
Deposit_DO	Deposit a digital object in a repository
Delete_DO	Deletes a digital object from a repository
MutateMeta	Edit the metadata for a digital object
Mutate_DO	Edit a digital object

Table 9.1. RAP commands, as listed in Arms, Blanchi & Overly (1997).

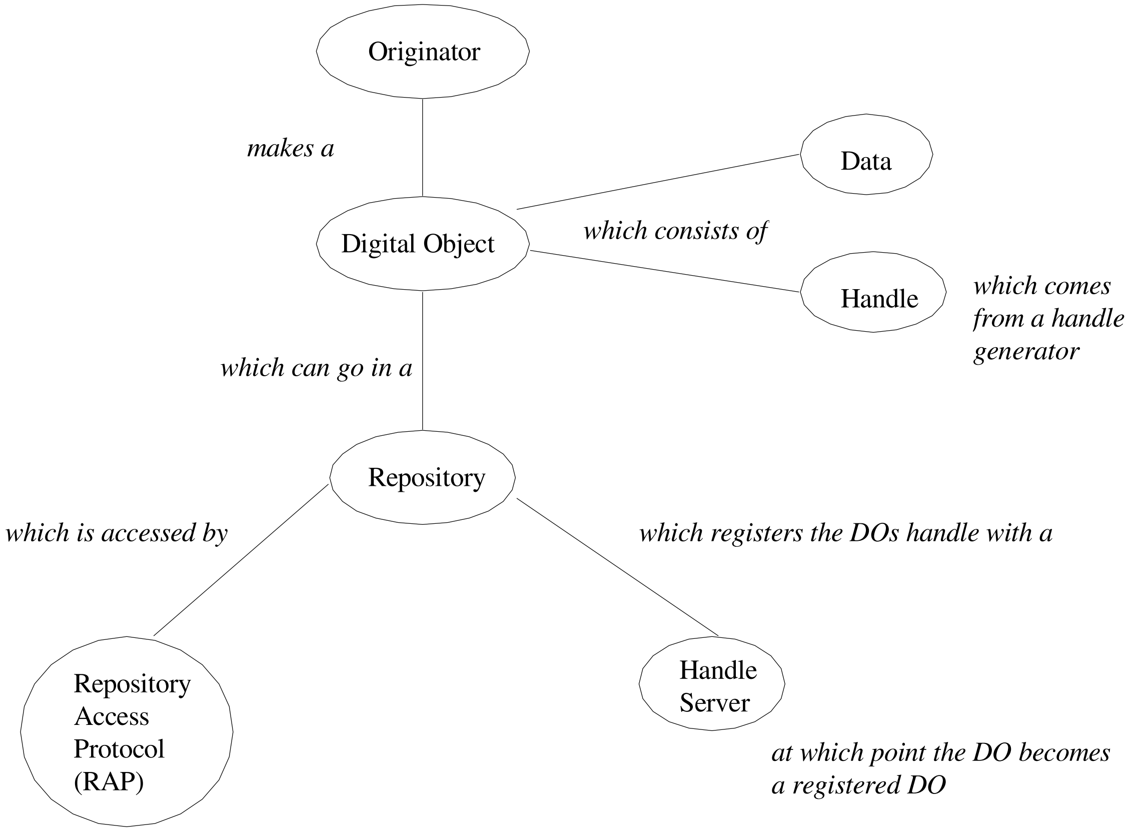

Originator
makes a
Data
Digital Object
which consists of
Handle
which comes from a handle generator
which can go in a
Repository
which is accessed by
which registers the DOs handle with a
Repository Access Protocol (RAP)
Handle Server
at which point the DO becomes a registered DO

Figure 9.1. Highlights of the Kahn–Wilensky Framework.

9.3 Summary

The Kahn−Wilensky Framework was never directly implemented, so the normal assessment of "strengths" and "weaknesses" does not seem applicable. However, it provided a new vision of a distributed information system and revised the fundamental concepts of "digital object" and "repository" in a way that strongly influenced subsequent digital libraries−related projects. Many of the projects discussed in this paper are either direct or indirect descendants of this architecture.

10.0 Metadata Encoding & Transmission Standard

10.1 Overview

The Metadata Encoding and Transmission Standard (METS) is project based on for encoding descriptive, administrative, and structural metadata regarding objects within a DL. The design of METS is heavily influenced by the participants in the Library of Congress' Making of America 2 (MOA2) project.

10.1.1 Developers

The development is sponsored Library of Congress (LOC) and Digital Library Federation (DLF), although the actual participants also include members of several universities.

10.1.2 Motivation

A workshop report (McDonough, Myrick & Stedfeld, 2001) documents the motivation for what would become known as METS, based on experiences with MOA2, anticipated requirements for MOA3, and evaluation of other technologies such as Resource Description Format (RDF) (Miller, 1998), MPEG−7 (Day & Martinez, 2001) and Synchronized Multimedia Integration Language (SMIL) (SMIL, 2001), as well as suitability for Open Archival Information Systems (OAIS) reference model (CCSDS, 2001).

10.2 Analysis

10.2.1 Architecture

A METS document consists of 4 sections:

- Descriptive metadata −bibliographic metadata, either encoded internally (textually or binary), links to external descriptive metadata, or both internal and external metadata.
- Administrative metadata − metadata concerning how the object was created, the terms & conditions for its use, translation / transformation history, etc. Administrative

metadata can also be internal, external or a combination.

- File Groups – a manifesto of all digital non–metadata content that comprises the digital object.
- Structural Map – provides the hierarchical structure of the digital object, as well as linking in the metadata and data as appropriate.

10.2.2 Example Applications

A full sample METS file can be found at http://www.loc.gov/standards/mets/sfquad.xml. METS documents are not difficult, but they can be lengthy, so we will examine only some smaller pieces drawn from http://www.loc.gov/standards/mets/METSOverview.html below. Figure 10.1 shows both external and internal Dublin Core and binary (Base64 encoded) descriptive metadata.

```
<dmdSec ID="dmd001">
      <mdRef LOCTYPE="URN" MIMETYPE="application/xml" MDTYPE="EAD"
       LABEL="Berol Collection Finding Aid">urn:x-nyu:fales1735</mdRef>

</dmdSec>

<dmdSec ID="dmd002">
      <mdWrap MIMETYPE="text/xml" MDTYPE="DC" LABEL="Dublin Core
Metadata">
            <dc:title>Alice’s Adventures in Wonderland</dc:title>
            <dc:creator>Lewis Carroll</dc:creator>
            <dc:date>between 1872 and 1890</dc:date>
            <dc:publisher>McCloughlin Brothers</dc:publisher>
            <dc:type>text</dc:type>
      </mdWrap>
</dmdSec>

<dmdSec ID="dmd003">
      <mdWrap MIMETYPE="application/marc" MDTYPE="MARC" LABEL="OPAC
Record">
            <binData>MDI0ODdjam0gIDIyMDA1ODkgYSA0NU0wMDAxMDA...(etc.)
             </binData>
      </mdWrap>
</dmdSec>
```

Figure 10.1 External and internal descriptive metadata in METS.

Figure 10.2 illustrates the administrative metadata and filegroup concepts, and the linkage between the two.

```
<amdSec ID="AMD001">
      <mdWrap MIMETYPE="text/xml" MDTYPE="NISOIMG" LABEL="NISO Img.
Data">
            <niso:MIMEtype>image/tiff</niso:MIMEtype>
            <niso:Compression>LZW</niso:Compression>
            <niso:PhotometricInterpretation>8</niso:PhotometricInterpret
ation>
            <niso:Orientation>1</niso:Orientation>
            <niso:ScanningAgency>NYU Press</niso:ScanningAgency>
      </mdWrap>
</amdSec>

<fileGrp>
      <fileGrp ID="FG001">
            <file ID="FILE001" ADMID="AMD001">
            <FLocat LOCTYPE="URL">http://dlib.nyu.edu/press/testimg.tif
            </FLocat>
            </file>
            <file ID="FILE002" ADMID="AMD001">
            <FLocat LOCTYPE="URL">http://dlib.nyu.edu/press/test2.tif
            </FLocat>
            </file>
      </fileGrp>
</fileGrp>
```

Figure 10.2. Administrative metadata and filegroups.

10.3 Summary

10.3.1 Strengths

The experience base behind MOA2 is significant. METS through the filegroup mechanism, has the advantage of containing links to the content itself and not being a metadata–only project. It enjoys the advantage of perspective and experience, but METS appears more ambitious than the Warwick Framework, and perhaps less so than VERS.

10.3.2 Weaknesses

Perhaps the biggest weakness associated with METS is its relatively new status. It is too soon to tell if tools will become available to support creation and management of METS files. It is also too soon to know if METS will be embraced over similar standards.

10.3.3 Future Directions

The METS project has just begun, so there is a great deal of activity still occurring. At the time of the writing, documents describing registries, extension schemas and tools & utilities are promised but not yet linked. The activity can be followed at http://www.loc.gov/standards/mets/.

11.0 Multivalent Documents

11.1 Overview

Multivalent documents are designed to be "anytime, anywhere, any type, every way user−improvable digital documents and systems" (Phelps, 1998). Multivalent documents are constructed as layers, allowing heterogeneous file types to be brought together into one semantically connected document. Basic document operations are defined as a set of protocols to allow functionality components, called behaviors, to act as a seamless unit even if they have been authored independently. The idea is not to create a new document format that incorporates all of the myriad file types, but rather to create a small core that can associate pieces of content with different file types with one another semantically, and that can call upon specific behaviors dynamically in order to manipulate those pieces of content while maintaining their place within the document as a whole.

11.1.1 Developers

Multivalent documents are being developed by Thomas Phelps, Robert Wilensky, and others in the Division of Computer Science at the University of California at Berkeley. The first references to multivalent documents appeared in 1996 (Phelps & Wilensky, 1996) and they remain in active development (Phelps & Wilensky, 1997; Phelps & Wilensky, 2000).

11.1.2 Motivation

Perhaps the greatest motivation behind the development of multivalent documents was the sheer diversity of document format types that existed in the mid−1990s, and the almost complete inability to combine those different format types into a coherent, semantically related whole. Because of the way different systems for document manipulation evolved over time, the user was forced to treat the various components of a complex digital document completely independently, and had to use different software applications in order to be able to view or edit these components.

Another problem that is an outgrowth of the document diversity problem is that both the documents created and the systems used to create them are not easily adaptable as user requirements for document capabilities change over time.

11.2 Analysis

The multivalent documents described here are part of the Multivalent Browser Developers Release 1. The Multivalent Browser is currently written in Java 2.0.

11.2.1 Architecture

The architecture is centered around the document as the core unit. So all of the functionality built into the system either serves to build documents or to support them in some way.

The primary data structure for multivalent documents is the document tree. Each document consists of a document class, which is a tree node. All document content is also constructed from tree nodes, so that documents and their contents can be nested inside one another. In addition, the document can contain behavior layers, implemented as document−specific Java classes, a style sheet, and the URL for its underlying data. The graphical user interface is implemented at the document's top level as nodes that implement interface features such as buttons and menus, and each document has a root that communicates events and coordinates screen redrawing with the browser window, which is defined to the operating system as the container for a document, its graphical user interface, and any documents that may be nested within it.

The leaf nodes of the document tree are responsible for displaying content of some recognized type, such as images, text strings in UNICODE format, or Java applets. While documents in HTML or plain text might not require document−specific leaf nodes, other kinds of documents will define their own leaf types. An example of this would be a scanned paper image that defines a hybrid image−OCR text type.

11.2.2 API

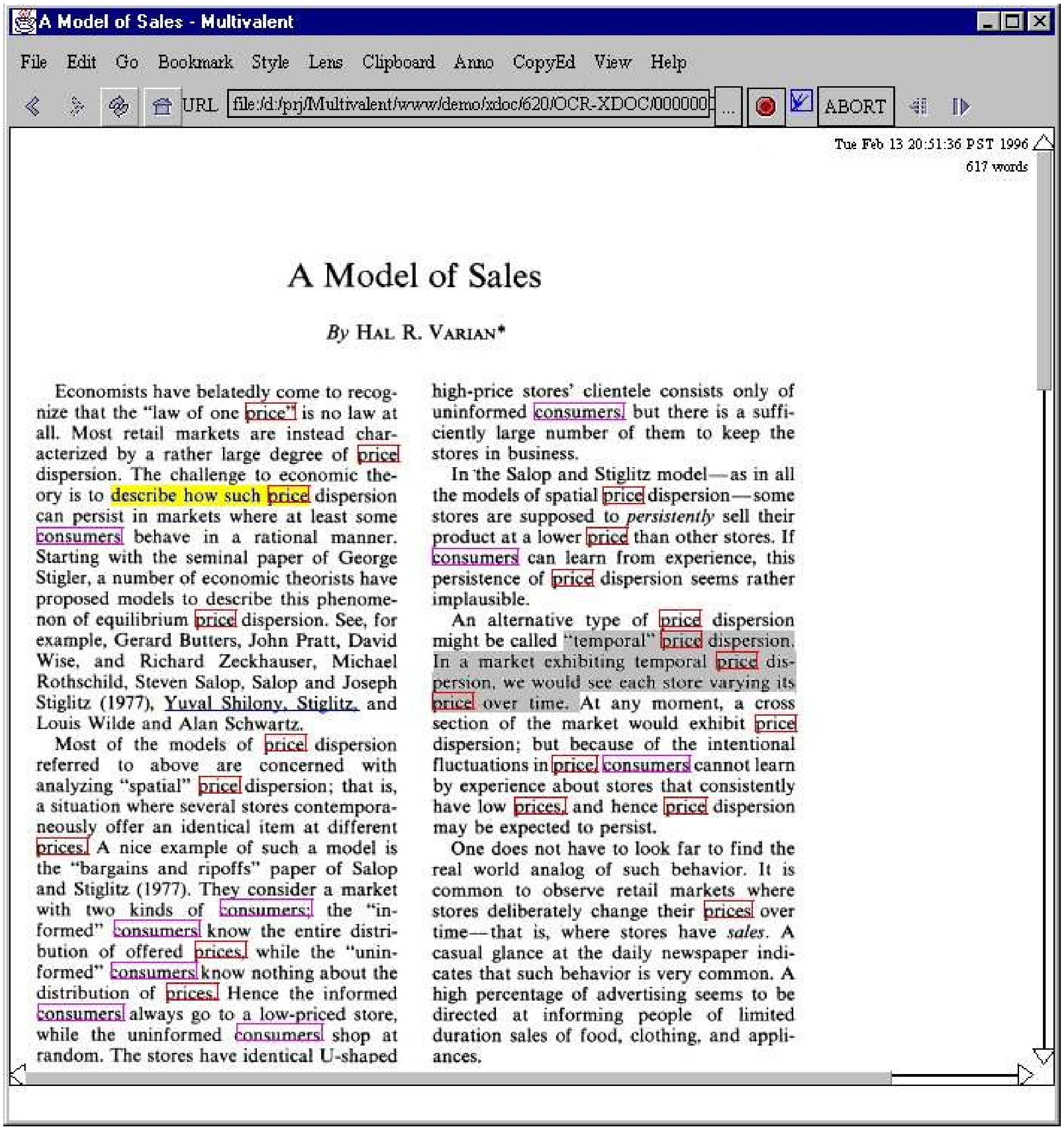
The API, along with other developer resources, is available for download at http://http.cs.berkeley.edu/~phelps/Multivalent/download.html.

11.2.3 System Requirements

The Multivalent Browser requires Java 2 v1.3 running under Solaris, Linux, Macintosh OS X, or Windows.

11.2.4 Sample Operation

The screenshots below show a sample document from the UC−Berkeley Digital Library (images are from http://http.cs.berkeley.edu/~phelps/Multivalent/doc/berkeley/adaptor/Xdoc.html). The document has been scanned using OCR, and then ScanSoft XDOC was used to align the OCR text with the page images (Figure 11.1). The grey highlighted area on the page image shows a segment of text whose OCR text can be cut and pasted into another document. The words that are highlighted are search results.


A Model of Sales - Multivalent
File
Edit
Go
Bookmark
Style
Lens
Clipboard
Anno
CopyEd
View
Help
URL
file:/d:/prj/Multivalent/www/demo/xdoc/620/OCR-XDOC/000000
...
ABORT
Tue Feb 13 20:51:36 PST 1996
617 words

A Model of Sales

By HAL R. VARIAN*

Economists have belatedly come to recognize that the "law of one price" is no law at all. Most retail markets are instead characterized by a rather large degree of price dispersion. The challenge to economic theory is to describe how such price dispersion can persist in markets where at least some consumers behave in a rational manner. Starting with the seminal paper of George Stigler, a number of economic theorists have proposed models to describe this phenomenon of equilibrium price dispersion. See, for example, Gerard Butters, John Pratt, David Wise, and Richard Zeckhauser, Michael Rothschild, Steven Salop, Salop and Joseph Stiglitz (1977), Yuval Shilony, Stiglitz, and Louis Wilde and Alan Schwartz.

Most of the models of price dispersion referred to above are concerned with analyzing "spatial" price dispersion; that is, a situation where several stores contemporaneously offer an identical item at different prices. A nice example of such a model is the "bargains and ripoffs" paper of Salop and Stiglitz (1977). They consider a market with two kinds of consumers: the "informed" consumers know the entire distribution of offered prices, while the "uninformed" consumers know nothing about the distribution of prices. Hence the informed consumers always go to a low-priced store, while the uninformed consumers shop at random. The stores have identical U-shaped high-price stores' clientele consists only of uninformed consumers, but there is a sufficiently large number of them to keep the stores in business.

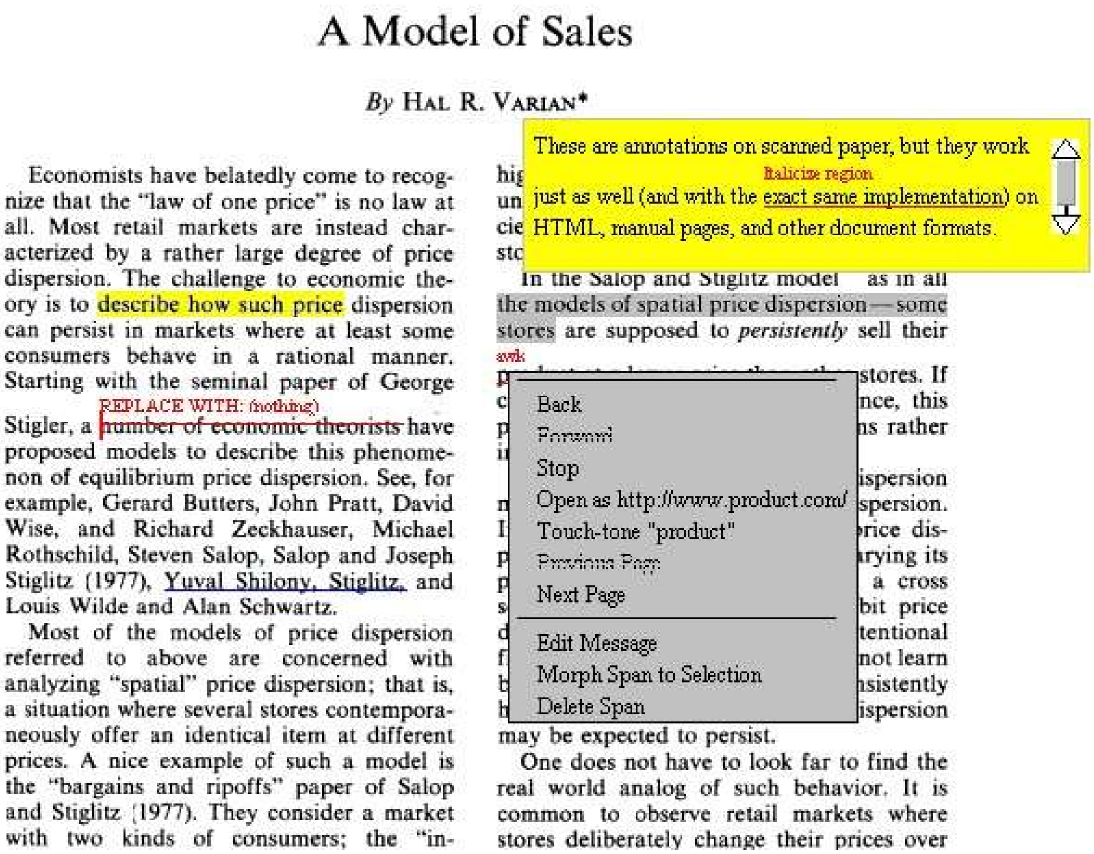
In the Salop and Stiglitz model—as in all the models of spatial price dispersion—some stores are supposed to *persistently* sell their product at a lower price than other stores. If consumers can learn from experience, this persistence of price dispersion seems rather implausible.

An alternative type of price dispersion might be called "temporal" price dispersion. In a market exhibiting temporal price dispersion, we would see each store varying its price over time. At any moment, a cross section of the market would exhibit price dispersion; but because of the intentional fluctuations in price, consumers cannot learn by experience about stores that consistently have low prices, and hence price dispersion may be expected to persist.

One does not have to look far to find the real world analog of such behavior. It is common to observe retail markets where stores deliberately change their prices over time—that is, where stores have *sales*. A casual glance at the daily newspaper indicates that such behavior is very common. A high percentage of advertising seems to be directed at informing people of limited duration sales of food, clothing, and appliances.

Figure 11.1. Performing text–based operations on a scanned document.

Figure 11.2 shows the same page image, this time with some annotations. Note that the REPLACE WITH annotation has actually altered the page image by creating more space between lines for the annotation. The yellow pop–up annotation is a window that can be moved around the screen (or closed), and it can also be annotated.



A Model of Sales

By Hal R. Varian*

Economists have belatedly come to recognize that the "law of one price" is no law at all. Most retail markets are instead characterized by a rather large degree of price dispersion. The challenge to economic theory is to describe how such price dispersion can persist in markets where at least some consumers behave in a rational manner. Starting with the seminal paper of George Stigler, a ~~number of economic theorists~~ have proposed models to describe this phenomenon of equilibrium price dispersion. See, for example, Gerard Butters, John Pratt, David Wise, and Richard Zeckhauser, Michael Rothschild, Steven Salop, Salop and Joseph Stiglitz (1977), Yuval Shilony, Stiglitz, and Louis Wilde and Alan Schwartz.

REPLACE WITH: (nothing)

Most of the models of price dispersion referred to above are concerned with analyzing "spatial" price dispersion; that is, a situation where several stores contemporaneously offer an identical item at different prices. A nice example of such a model is the "bargains and ripoffs" paper of Salop and Stiglitz (1977). They consider a market with two kinds of consumers; the "in-

These are annotations on scanned paper, but they work just as well (and with the exact same implementation) on HTML, manual pages, and other document formats.

Italicize region

In the Salop and Stiglitz model as in all the models of spatial price dispersion—some stores are supposed to *persistently* sell their

awk

stores. If

nce, this

ns rather

ispersion

spersion.

price dis-

rying its

a cross

bit price

tentional

not learn

nsistently

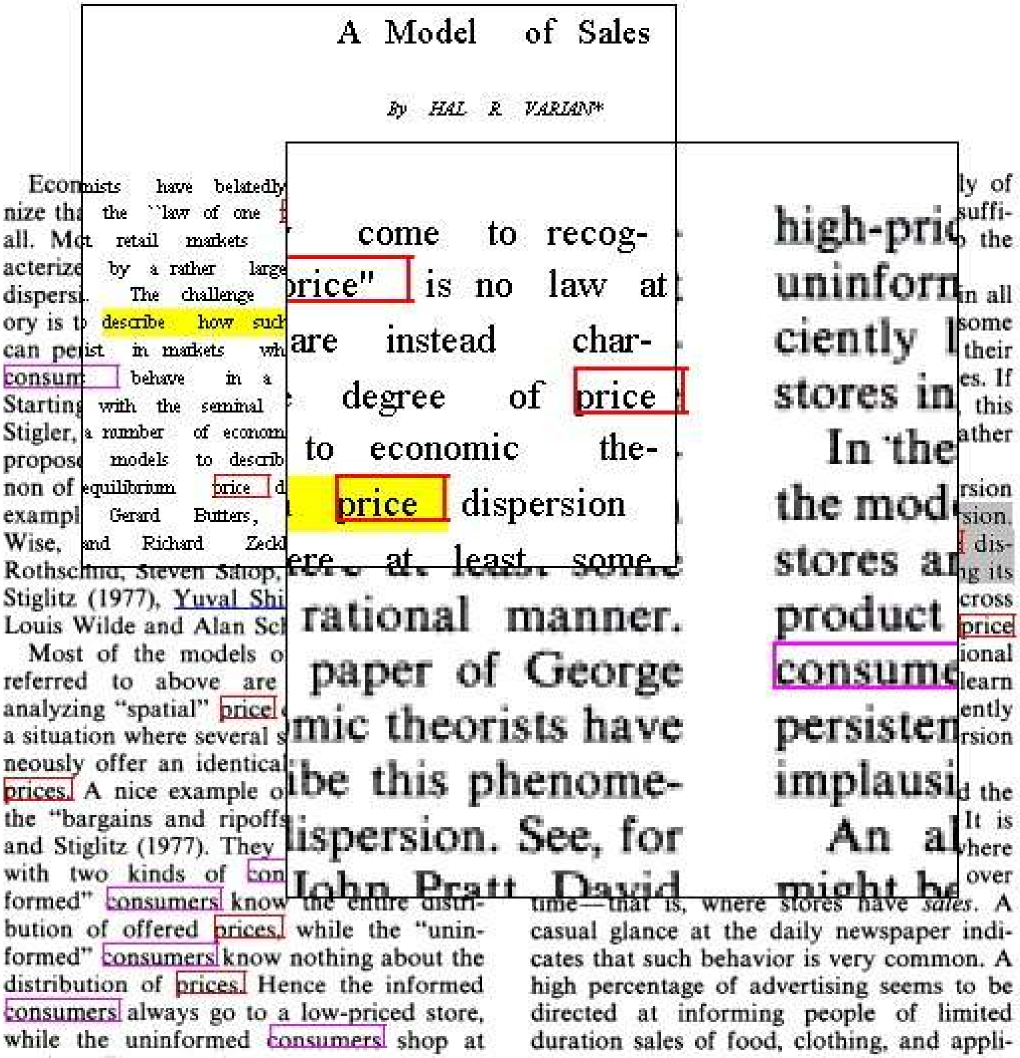
ispersion

may be expected to persist.

One does not have to look far to find the real world analog of such behavior. It is common to observe retail markets where stores deliberately change their prices over

Figure 11.2 Annotating the scanned document.

Figure 11.3 shows some of the lenses in operation on the same page. Here, a lens shows the OCR text that is linked to the page image, and there is also see a magnification lens. This lens magnifies whatever appears behind it, so notice that it magnifies not only the page image, but also the contents of the OCR scanned text lens.



A Model of Sales

By HAL R. VARIAN*

"formed" consumers know the entire distribution of offered prices, while the "uninformed" consumers know nothing about the distribution of prices. Hence the informed consumers always go to a low-priced store, while the uninformed consumers shop at

time—that is, where stores have *sales*. A casual glance at the daily newspaper indicates that such behavior is very common. A high percentage of advertising seems to be directed at informing people of limited duration sales of food, clothing, and appli-

Figure 11.3 An OCR and magnification lens on the document.

11.3 Summary

11.3.1 Strengths

Multivalent documents are well suited for use in situations where the documents lend themselves to having multiple semantic layers. This includes scanned documents (images + OCR text), language translations (e.g., English & French), annotations (cf. ComMentor), and geospatial data (maps with geographic and political layers). Having the system constructed in Java provides some platform independence, and since the emphasis has been on building an architecture that could support and relate existing file formats, it does not require conversion of existing documents into a new format. It is a relatively compact system, with a small core that loads individual behaviors on demand, so it does not incur a large amount of overhead on documents.

11.3.2 Weaknesses

Multivalent documents are not designed with any native security functionality (such as terms and conditions or authentication). The researchers emphasize that since much of the emphasis has been on creating a generalized framework that can accept modular pieces of software to handle particular behaviors for particular objects, that security is a place where functionality could be implemented through third party modules. The high functionality that multivalent documents provide comes at the expense of complexity in creation. However, there are significant developer resources and documentation available for those who wish to use multivalent documents.

11.3.3 Future Directions

Multivalent documents and the multivalent browser are under active development. In future versions, the researchers hope to improve the handling of distributed annotations, incorporate new media formats (the complete specification for HTML 4.0, CSS1, QuickTime, Flash, MP3, PDF, SVG, etc.), incorporate native XML display along with style sheets, and provide full−text searching of all document types. The project website is at http://http.cs.berkeley.edu/~phelps/Multivalent/.

12.0 Open eBook

12.1 Overview

The Open eBook (OEB) initiative defines a structure and terminology for the interoperation of "electronic books". In an attempt to tame the proliferation of terms to describe e−books, the OEB offers the following definitions (OEBPS, 2001): an *OEB document* is an XML document that conforms to the OEB specification; a *reader* is person that reads an OEB; a *reading device* is the hardware/software combination for rendering publications; and a *reading system* is a hardware/software combination for "accepting OEB documents, and directly or indirectly making them available to readers" (potentially allowing reading systems to convert OEBs into something that a reading device can natively understand).

12.1.1 Developers

Over 80 eBook, document, and e−commerce related companies are involved (http://www.openebook.org/who.asp has a full list), and the main contributors include Adobe, InterTrust and Microsoft. The initiative was first announced in 1998, with the first specification released in 1999.

12.1.2 Motivation

There has been considerable speculation as to the future of the "book" in a digital environment, and despite the publicity surrounding "electronic books", it has yet to materialize into a profitable market (Lynch, 2001). The various companies have an interest in defining some level of interoperability between the various encodings, software and hardware to accelerate the development of this market.

12.2 Analysis

The OEB Publication Structure (OEBPS) is the union of a number of other web–based technologies, including: XML, HTML, Cascading Style Sheets (CSS), JPEG, Portable Network Graphics (PNG), Dublin Core, Unicode, MIME, etc. Key, related concepts in the OEBPS include *extensibility* and *fallback*. Extensibility allows OEB document creators to extend the functionality of their documents with arbitrary media types. However, the fallback mechanism holds that anything in the OEB document that is not XML, CSS, JPEG or PNG needs to have a fallback version of the non–supported media type in one of the 4 supported media types (XML, CSS, JPEG, PNG). This allows an OEB document to take advantage of advanced media types when possible, while still allowing for interoperability through a "lowest common denominator" media type. The fallback mechanism is analogous to the "dumb–down principle" of Dublin Core qualifiers (DCMI, 2000).

12.2.1 Architecture

An OEB document is an XML file that conforms to the OEBPS specification. The OEB document contains exactly one OEB package file. The package file ends in ".opf" and has a MIME type of "text/xml". The package file contains:

- *Package Identity* – a URI for the OEB document.
- *Metadata* – Dublin Core metadata, with possible extensions.
- *Manifest* – a list of files (documents, images, style sheets, etc.) that comprise the OEB document. The manifest also includes fallback declarations to handle files not supported by OEBPS 1.0.1.
- *Spine* – an arrangement of items providing a linear reading order of the OEB document.
- *Tours* – a set of alternate reading sequences through the OEB document. This could include different tours for different reading purposes, reader expertise levels, etc.
- *Guide* – a set of references to fundamental structural features of the publication, such as table of contents, foreword, bibliography, index, acknowledgments, etc.

12.2.2 Example Applications

Drawing from some of the examples in OEBPS (2001), Figure 12.1 shows a package file with a unique id, Dublin Core metadata (with attributes defined by the OEBPS).

```
<package unique-identifier="xyz">
      <metadata>
            <dc-metadata xmlns:dc="http://purl.org/dc/elements/1.0/"
xmlns:oebpackage="http://openebook.org/namespaces/oeb-package/1.0/">
                        <dc:Title>Alice in Wonderland</dc:Title>
                  <dc:Type>Novel</dc:Type>
                  <dc:Identifier id="xyz"
                  scheme="ISBN">123456789X</dc:Identifier>
                  <dc:Creator role="aut">Lewis Carroll</dc:Creator>
            </dc-metadata>
      </metadata>
      ...
</package>
```

Figure 12.1. An OEB package file.

Figure 12.2 illustrates a manifest with several items, including a fall back for reading systems that do not support MPEGs, as well as a spine.

```
<manifest>
      <item id="toc" href="contents.html" media-type="text/x-oeb1-
document"/>
      <item id="c1" href="chap1.html" media-type="text/x-oeb1-
document"/>
      <item id="c2" href="chap2.html" media-type="text/x-oeb1-
document"/>
      <item id="c3" href="chap3.html" media-type="text/x-oeb1-
document"/>
      <item id="results" href="results.mpeg" media-type="video/mpeg"
fallback= "results-simple"/>
      <item id="results-simple" href="results.jpg" media-
type="image/jpeg"/>
      <item id="f1" href="fig1.jpg" media-type="image/jpeg"/>
      <item id="f2" href="fig2.jpg" media-type="image/jpeg"/>
      <item id="f3" href="fig3.jpg" media-type="image/jpeg"/>
</manifest>

<spine>
      <itemref idref="toc"/>
      <itemref idref="c1"/>
      <itemref idref="c2"/>
      <itemref idref="c3"/>
</spine>
```

Figure 12.2. An OEB manifest & spine.

Figure 12.3 includes both a list of tours appropriate for the culinary preferences of the readers, as well as a guide listing the pertinent parts of the OEB document.

```
<tours>
      <tour id="tour1" title="Chicken Recipes">
            <site title="Chicken Fingers" href="appetizers.html#r3" />
            <site title="Chicken a la King" href="entrees.html#r5" />
      </tour>
      <tour id="tour2" title="Vegan Recipes">
            <site title="Hummus" href ="appetizer.html#r6" />
            <site title="Lentil Casserole" href="lentils.html" />
      </tour>
</tours>

<guide>
      <reference type="toc" title="Table of Contents" href="toc.html" />
      <reference type="loi" title="List Of Illustrations"
href="toc.html#figures" />
      <reference type="other.intro" title="Introduction"
href="intro.html" />
</guide>
```

Figure 12.3. An OEB tour & guide.

12.3 Summary

12.3.1 Strengths

The low level of base interoperability is a significant advantage and the Dublin Core–like "fallback" rules are a good method of maintaining interoperability while allowing extended OEB documents. The composite technologies of OEB are solid and being driven by other communities. The OEB Forum promises backward compatibility between OEBPS versions.

12.3.2 Weaknesses

The OEB Publication Structure 1.0.1 currently does not address digital rights management, though this on the agenda and could be addressed in version 2.0. However, it remains to be seen how rights management and other social expectations for electronic books (Lynch, 2001) impact the OEB.

12.3.3 Future Directions

The OEB Forum has many significant commercial partners. The OEB Publication structure 1.0.1 came out in 2001, and a 2.0 version is currently being developed. The status can be monitored at http://www.openebook.org/.

13.0 VERS Encapsulated Objects

13.1 Overview

The Victorian Electronic Records Strategy (VERS) is a project developed to ensure a long–term preservation to the electronic records produced by the government agencies of the state of Victoria in Australia (Waugh et al., 2000). VERS is focused on the construction of a system able to capture records from common desktop applications and to convert them into a preservation format that ensures a long–term access and readability. Following a background investigation (1995–6) and a prototype implementation (1998), it is now in its third stage (1999–2001), which focuses on the implementation of the system on every desktop within the Victoria Government Department of Infrastructure and on testing the economic feasibility (Heazelewood, et al., 1999).

13.1.1 Developers

VERS was funded by Victorian State Government (Australia) where it is currently tested. It is sponsored and run by the Public Record Office Victoria (PROV), the state's archival authority that publishes standards for the management of public records, in conjunction with the Australian Commonwealth Scientific and Industrial Research Organization (CSIRO), and Ernst & Young.

13.1.2 Motivation

As a result of unsuccessful investigations on available systems for permanent preservation, the VERS project was intended to find a solution for preserving electronic government records produced by Victorian government agencies for the indefinite future.

13.2 Analysis

Among several possible approaches to digital preservation, VERS adopted the *encapsulation* strategy because it was considered the most technically appropriate to address the preservation needs of a government organization. Such an organization requires the information that is preserved also be continually used and updated with further inclusions.

Preservation by *encapsulation* consists on wrapping the information to be preserved within a human readable wrapper that contains information. This information is metadata that provides documentation about the preserved information and enables them to be identified and decoded in future. The metadata describes also the format of the *encapsulation* itself. This means that the preserved record is self–documented and self–sufficient, non depending on systems, data or documentation.

13.2.1 Architecture

The basic element of the VERS architecture is the VERS Encapsulated Object (VEO). VEO is the result of the conversion of document and metadata into a unit meant to be visible indefinitely. VEO can contain any type of object including images and audio/video files. The VEOs have a common record structure to allow a common management of all types of records.

The structure of a VERS Encapsulated Object is composed of four layers: *VEO Object, Content, Document, and Encoding* (Figure 13.1). It is called an *Onion* model in which the information contained in the record maintains its integrity without dispersion across systems. Each layer contains metadata with descriptive information about the specific layer. Metadata and content embedded in each VEO, instead of in different databases, makes the VEO self–sufficient. The VERS record also includes authentication information and digital signature technology that validates the object (Figure 13.2).

Changes to the record may be made, without interfering with the status of the record, by wrapping a new layer of encoding around the existing VEO. This process, called the *Onion record* approach, allows storing the record’s history within the record itself and preserves its integrity.

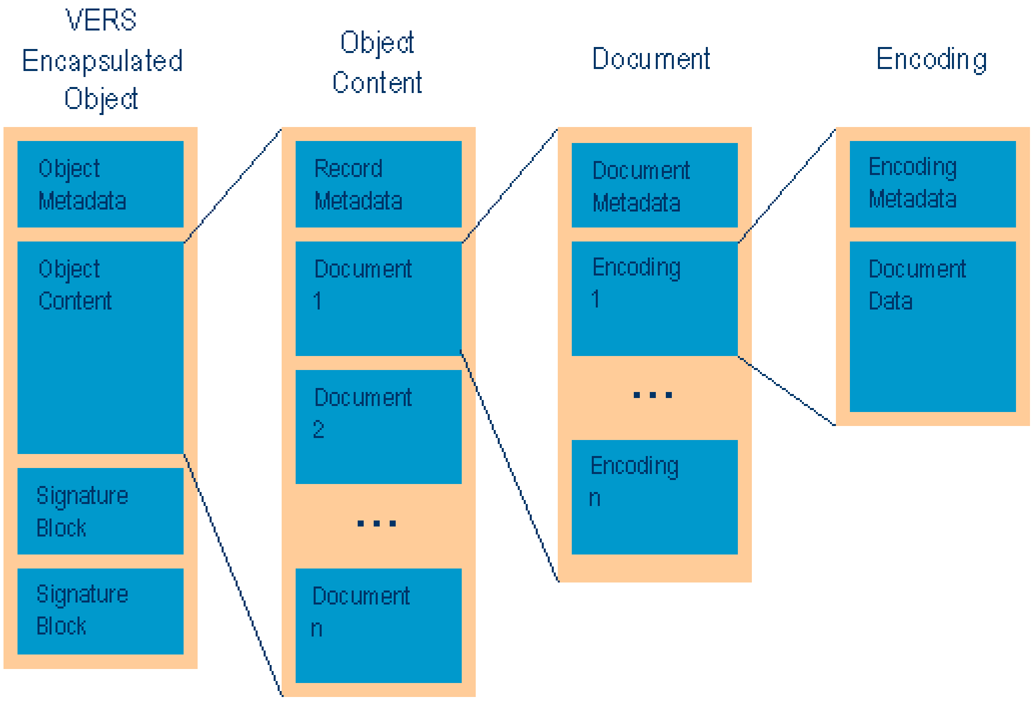


Figure 13.1. VEO structure (from http://www.prov.vic.gov.au/vers/standard/99–7–1s5.htm).

The VERS Final Report proposes the VERS Demonstrator System, a prototype of an electronic archive system within Victorian government agencies. This implementation is considered only a demonstrative architectural solution. The VERS Demonstrator System consists of three major components: *Record capture* for capturing records and converting then into a long–term format, *Repository* for managing the archival functions, and *Record Discovery* for searching and displaying archived records. These three components can be modeled in different ways. The archiving process can be synthesized through the following phases: records are captured from a range of diverse applications and in a variety of formats. A *record encapsulation module* converts records and their associated metadata into the *long life electronic record* format. The records are locked to avoid undetectable modifications and passed to the *Repository* that stores the archived record. Upon registering new records, the *Repository* sends a copy of the record to the *Discovery* system where it is indexed and cached. The *Discovery* system provides the interface for locating and retrieving the records.

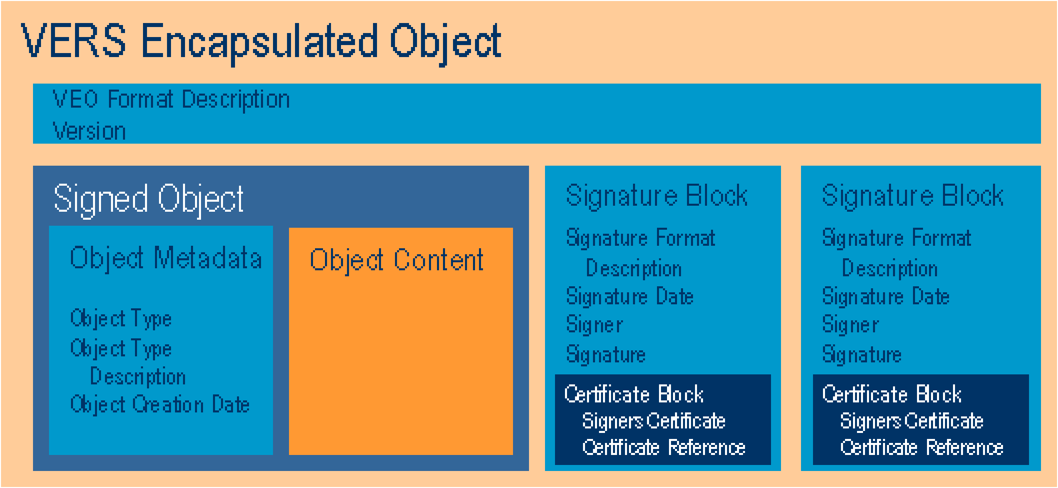


Figure 13.2. Generic VERS record (from http://www.prov.vic.gov.au/vers/standard/99–7–1s5.htm).

13.2.2 System Requirements

Extensible Markup Language (XML) is the long–term format recommended within the VERS project for encoding metadata wrapped around the record content. XML is preferred to a binary data format because XML does not depend upon the program that interprets the binary data to extract the content. The project has defined Standard encodings of three types of documents: documents, database tables, and records and has developed an XML DTD. For encoding the record content, VERS recommends the use of Adobe Portable Document Format (PDF) as the best, widely published, long–term format currently available.

13.3 Summary

13.3.1 Strengths

Instead of being only theoretical, as most work on digital preservation, VERS adopts a practical approach aimed to build actual systems where to test and implement its technology. The strategy proposed by VERS is developed from experimentation. The approach is basically data–driven in order to maintain independence from system constraints.

13.3.2 Weaknesses

At present it is still not clear how to best deal with objects more complex than flat files. Possible options are to "flatten" the objects and make them simple or to choose a data format that supports complex object and linking. VERS is proclaimed as the "100 year experiment" (Francis et al., 1998). It will be a number of years before the effectiveness of the preservation proposed by VERS be tested.

13.3.3 Future Directions

Australian government continues to fund the VERS project for an implementation at the Department of Infrastructure. It is under way the expansion of VERS to other Australian agencies with the inclusion of elements of VERS within their new systems. The project web page is at http://www.prov.vic.gov.au/vers/.

14.0 Warwick Framework

14.1 Overview

Diverse communities of users and stakeholders create and maintain metadata schemas for addressing different applications, domains, and functional needs. Different metadata models are related in many ways, with various levels of interaction, often overlapping and competing. The Warwick Framework (WF) offers a conceptual foundation for a high–level infrastructure for aggregation and exchange of metadata from multiple schemas associated with a common resource (Lagoze, Lynch & Daniel, 1996; Lagoze, 1996).

14.1.1 Developers

Among the many present at the initial workshop, the core developers of the WF concept were Carl Lagoze (Cornell University), Ron Daniel Jr. (Los Alamos National Laboratory) and Clifford Lynch (University of California – Office of the President).

14.1.2 Motivation

During the 2nd Dublin Core Metadata Workshop, held at Warwick University in April 1996 (Dempsey & Weibel, 1996), the representatives of several groups of metadata developers expressed the need for an integration of metadata from different schemas into a common information architecture, able to ensure interoperability among metadata of different types. The meeting also addressed the need to insure compatibility with further developments and extensions of the metadata schemas, and attempted to build on the success of the first meeting, which produced Dublin Core (Weibel, 1995). Consensus was reached on the idea of a container architecture for interchanging multiple metadata packages. The Warwick Framework was the first practical approach that provides the effective integration of metadata into a comprehensive information infrastructure.

14.2 Analysis & Architecture

The Warwick Framework is founded on a container–package approach where discrete packages of metadata can be aggregated by users or software agents in conceptual containers. The architecture of the Warwick Framework is essentially based on two main components: the *container* and the *metadata package* (Figure 14.1). A container is responsible for shipping metadata among repositories, clients, and agents. The container can be considered a first–class object and can be managed as any other resource, stored in servers and accessed using a URI.

The package contains actual metadata sets such as Dublin Core records, MARC records, encoded terms and conditions, etc. The metadata package can be physically included in the container, but can also be logically contained and accessed by references such as URLs. A container itself may be a package and can be embedded in another container. Both package and container have identifiers for cross–reference one another.

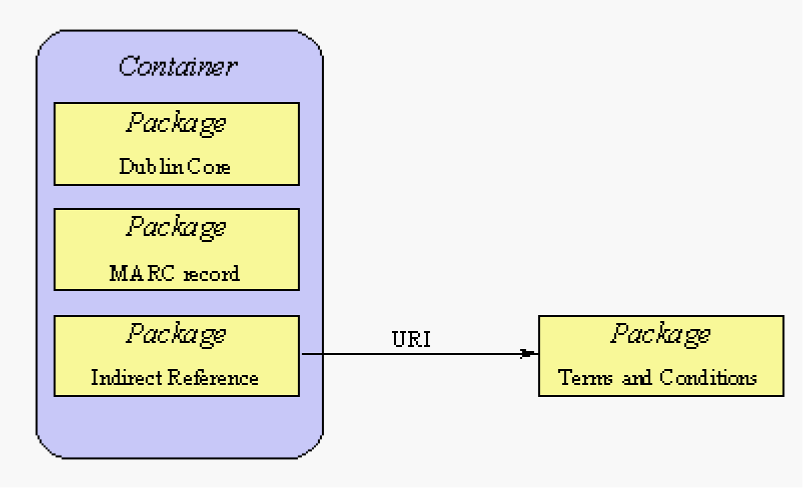


Figure 14.1 A simple Warwick Framework container (from Daniel & Lagoze (1997a)).

As a result of further research following the first phase of definition of the model, the Warwick Framework was extended to address issues concerning the relationships between the packages. Upon access a container of related packages, there was not an explicit understanding of the relationships between the data types of the packages. A mechanism called Warwick Framework Catalog (WFC) was defined in order to express these relationships. The WFC is a method for naming relationships between metadata packages and defining the semantics. WFC provides a list of simple names identifying the relationships. The names might be listed in a Registry or be simply URIs. This abstraction makes the relationships be considered first–class object, so that they can be retrieved and executed. A mechanism called Distributed Active Relationships (DAR) provides a method for explicitly expressing the relationships and allow them to be dynamically downloaded and executed. Through these abstractions, the Warwick Framework becomes a more general architecture and the container can be considered itself a framework for aggregating distinct datasets.

14.3 Summary

14.3.1 Strengths

The Warwick Framework's distributed architecture insures interoperability across resources and consistency in the aggregation and exchange of metadata. Its modular design supports the extensibility of the metadata, without overlapping and redundancy. The simplicity of the design enables the Warwick Framework to be expressed in the context of the current Web infrastructure. Some possible implementations proposed include MIME, SGML, CORBA, and HTML. MIME, in particular, seems to be suitable for implementing the framework since it is already a well–established protocol for transmitting and storing multiple object documents on the Internet and may support the container–package functionality.

14.3.2 Weaknesses

The primary shortcoming of the Warwick Framework could be considered its focus only on metadata, and not data itself. This was addressed as the Warwick Framework evolved into FEDORA.

14.3.3 Future Directions

Although a MIME–based implementation was prototyped (Knight & Hamilton, 1996), the Warwick Framework was never actually implemented. A combination of new technologies (e.g., XML) and the evolution of new projects caused the Warwick Framework to be overcome by events. However, the models served as a foundation for subsequent information system projects that incorporated some of its concepts and components. FEDORA generalized the containers to include data and services, and the Resource Description Framework (RDF) for integrating disparate metadata descriptions was greatly influenced by the design of the Warwick Framework (Daniel & Lagoze, 1997a; Miller, 1998).

15.0 Conclusions

We have introduced a number of technologies that address the need for complex objects in web−based DLs.   It is important to note that the various projects were designed to address a number of different requirements, including aggregation of metadata, content & services, e−commerce and security, annotation & collaboration, and digital preservation. While we did not evaluate the technologies against the requirements of a specific deployment profile, we hope that we have provided enough information to allow the readers to judge applicability to their specific scenarios (Table 15.1).

While we have tried to focus on current technologies, we theoretical and historical projects that had significant influence on other current projects. While we have strived to be accurate and fair, we accept responsibility for any errors or inaccuracies that may be present. We welcome comments and corrections about these and other complex object technologies.

Acknowledgments

The majority of this work was performed while the first author was visiting at the School of Information and Library Science, University of North Carolina at Chapel Hill. We thank Cristophe Blanchi (CNRI) for his comments on FEDORA and RAP.

Technology	***Approximate Active Dates***	***Creators & Collaborators***	***Focus***	***Synopsis***
Aurora	1997–1999	University of Crete	Aggregation	CORBA / Java based aggregation of content and services
Buckets	1997 – present	NASA LaRC & Old Dominion University	Aggregation	Independent, mobile Perl data objects
ComMentor	1994–1995	Stanford University	Annotation	Modified http server and client records, presents annotations
Cryptolopes	1995 – present	IBM	e–commerce	Superdistribution with opaque data objects
Digibox	1995 – present	Electronic Publishing Resources & InterTrust	e–commerce	Superdistribution with enhanced PDFs, MPEGs, etc.
DMA	1997 – present	Association for Information & Image Management	Document management systems	Middleware for interoperability between DM systems
FEDORA	1997 – present	Cornell University, Los Alamos National Laboratory, University of Virginia	Aggregation	CORBA based aggregation of content and services
Kahn–Wilensky Framework	1995	CNRI & University of California – Berkeley	Distributed digital object definitions	An unimplemented framework that influenced the design of many DL projects
METS	2001 – present	Library of Congress, Digital Library Federation	Metadata aggregation	XML schema for aggregating descriptive, administrative and structural metadata
Multivalent Documents	1996 – present	University of California – Berkeley	Annotation & collaboration	Java based tools for reading MVD files.
Open eBooks	1998 – present	Open eBook Forum	Electronic books	XML structure interoperability and extensibility in e–books
VEOs	1995 – present	Victorian State Government (Australia)	Digital preservation	XML files encapsulating metadata, data and formatting rules.
Warwick Framework	1996 – 1997	Cornell University & Los Alamos National Laboratory	Metadata aggregation	Framework that influenced FEDORA and RDF

Table 15.1. Summary of findings.

References

Arms, W. Y. (1995). Key Concepts in the Architecture of the Digital Library. D−Lib Magazine, 1(1). Available at: http://www.dlib.org/dlib/July95/07arms.html

Arms, W. Y., Blanchi, C. & Overly, E. A. (1997). An Architecture for Information in Digital Libraries. D−Lib Magazine, 3(2). Available at: http://www.dlib.org/dlib/february97/cnri/02arms1.html

Bowman, C. M., Dharap, C., Baruah, M., Camargo, B. & Potti, S. (1994). A File System for Information Management. Proceedings of the Conference on Intelligent Information Management Systems. Available at: http://citeseer.nj.nec.com/bowman94file.html

Bowman, M. & Camargo, B. (1998). Digital Libraries: The Next Generation in File System Technology. D−Lib Magazine, 4(2). Available at: http://www.dlib.org/dlib/february98/bowman/02bowman.html

Browne, S., Dongarra, J., Grosse, E. & Rowan, T. (1995). The Netlib Mathematical Software Repository. D−Lib Magazine, 1(3). Available at: http://www.dlib.org/dlib/september95/netlib/09browne.html

CCSDS (2001). Reference Model for an Open Archival Information System (OAIS). Available at: http://www.ccsds.org/documents/pdf/CCSDS−650.0−R−2.pdf

Daniel, R. Jr., & Lagoze, C. (1997a). Extending the Warwick Framework From Metadata Containers to Active Digital Objects. D−Lib Magazine, 3(11). Available at: http://www.dlib.org/dlib/november97/daniel/11daniel.html

Daniel, R. Jr. & Lagoze, C. (1997b). Distributed Active Relationships in the Warwick Framework. Proceedings of the Second IEEE Metadata Workshop. Available at: http://computer.org/conferen/proceed/meta97/papers/rdaniel/rdaniel.pdf

Daniel, R. Jr., Lagoze, C., & Payette, S. (1998). A Metadata Architecture for Digital Libraries. Proceedings of the IEEE Advances in Digital Libraries (pp. 276−288). Available at: http://www.cs.cornell.edu/lagoze/papers/ADL98/dar−adl.html

Day, N. & Martinez, J. M. (2001). Introduction to MPEG−7 (v3.0). Available at: http://www.darmstadt.gmd.de/mobile/MPEG7/Documents/W4325%20M7%20Intro.htm

Davis, J. R. & Lagoze, C. (2000). NCSTRL: Design and Deployment of a Globally Distributed Digital Library. Journal of the American Society for Information Science, 51(3), 273−280.

DCMI. (2000). Dublin Core Qualifiers. Available at: http://www.dublincore.org/documents/dcmes−qualifiers/

Dempsey, L. & Weibel, S. L. (1996). The Warwick Metadata Workshop: A Framework for the Deployment of Resource Description. D−Lib Magazine, 2(7). Available at: http://www.dlib.org/dlib/july96/07weibel.html

DMA Technical Committee. (2001). DMA 1.0 Specification. Available at: http://www.infonuovo.com/dma/dma1.0−7/

Digital Object Store. (2001). Available at: http://www.cnri.reston.va.us/digital_object_store.html

Erickson, J. S. (2001a). Information Objects and Rights Management: A Mediation−based Approach to DRM Interoperability. D−Lib Magazine, 7(4). Available at:: http://www.dlib.org/dlib/april01/erickson/04erickson.html

Erickson, J. S. (2001b). A Digital Object Approach to Interoperable Rights Management: Fine−grained Policy Enforcement Enabled by a Digital Object Infrastructure. D−Lib Magazine, 7(6). Available at:: http://www.dlib.org/dlib/june01/erickson/06erickson.html

Folk, M. (1998). HDF as an Archive Format: Issues and Recommendations. Available at: http://hdf.ncsa.uiuc.edu/archive/hdfasarchivefmt.htm

Francis, R., Gibbs, R., Harari, L., Heazlewood, J., Hills, B., Leask, N., Sefton, A., Waugh, A. & Wilkinson, R. (1998). Electronic Archiving a 100 Year Experiment. Proceedings of the Third Document Computing Symposium. Available at: http://www.prov.vic.gov.au/vers/adcs98.pdf

Gifford, D. K., Jouvelot, P., Sheldon, M. A. & O'Toole, J. W. Jr. (1991). Semantic File Systems. Proceedings of the 13th ACM Symposium on Operating Systems Principles. Available at: http://www.psrg.lcs.mit.edu/publications/Papers/sfsabs.htm

Ginsparg, P. (1994). First Steps Towards Electronic Research Communication. Computers in Physics, 8, 390−396. Available at: http://arXiv.org/blurb/blurb.ps.gz

Gladney, H. M. (1997). Safeguarding Digital Library Contents and Users Document Access Control. D−Lib Magazine, 3(6). Available at: http://www.dlib.org/dlib/june97/ibm/06gladney.html

Gladney, H. M. & Mintzer, F. & Schiattarella, F. (1997). Safeguarding Digital Library Contents and Users Digital Images of Treasured Antiquities. D−Lib Magazine, 3(7/8). Available at: http://www.dlib.org/dlib/july97/vatican/07gladney.html

Gladney, H. M. & Lotspiech, J. B. (1997). Safeguarding Digital Library Contents and Users: Assuring Convenient Security and Data Quality. D–Lib Magazine, 3(5). Available at: http://www.dlib.org/dlib/may97/ibm/05gladney.html

Gladney, H. M. (1998). Safeguarding Digital Library Contents and Users Interim Retrospect and Prospects. D–Lib Magazine, 4(7). Available at: http://www.dlib.org/dlib/july98/gladney/07gladney.html

Gladney, H. M. & Lotspiech, J. B. (1998). Safeguarding Digital Library Contents and Users Storing, Sending, Showing, and Honoring Usage Terms and Conditions. D–Lib Magazine, 4(5). Available at: http://www.dlib.org/dlib/may98/gladney/05gladney.html

Grimshaw, A. S. & Loyot, E. C., Jr. (1991). ELFS: Object–Oriented Extensible File Systems. UVA CS TR CS–91–14. Available at: ftp://ftp.cs.virginia.edu/pub/techreports/CS–91–14.ps.Z

Haines, M., Mehrotra, P. & Van Rosendale, J. (1995). SmartFiles: An OO Approach to Data File Interoperability. Proceedings of the 1995 Conference On Object–Oriented Programming, Systems, Languages and Applications (pp. 453–466). (Also available as: NASA CR–198187 and ICASE Report No. 95–56). Available at: ftp://ftp.icase.edu/pub/techreports/95/95–56.pdf

Heazlewood, J., Dell Oro, J., Harari, L., Hills, B., Leask, N., Sefton, A., Waugh, A. & Wilkinson, R. (1999). Electronic Records: Problem Solved? A Report on the Public Record Office Victoria's Electronic Records Strategy. Archives and Manuscripts: The Journal of the Australian Society of Archivists, 27(1). Available at: http://www.prov.vic.gov.au/vers/am.pdf

Herzberg, A. (1998). Safeguarding Digital Library Contents: Charging for Online Content. D–Lib Magazine, 4(1). Available at: http://www.dlib.org/dlib/january98/ibm/01herzberg.html

Ibrahim, R. (1998). Component–Based Systems: A Formal Approach. Proceedings of the Component Oriented Software Engineering (COSE'98) Workshop. Available at: http://www.fit.qut.edu.au/~ibrahim/cosez.ps.Z

Kahn, R. E. & Lyons, P. A. (2001). Representing Value as Digital Objects: A Discussion of Transferability and Anonymity. D–Lib Magazine, 7(5). Available at:: http://www.dlib.org/dlib/may01/kahn/05kahn.html

Kahn, R. & Wilensky, R. (1995) A Framework for Distributed Digital Object Services. cnri.dlib/tn95–01. Available at: http://www.cnri.reston.va.us/home/cstr/arch/k–w.html.

Kang, B. & Wilensky, R. (2001). Toward a Model of Self−Administering Data. Proceedings of the First ACM/IEEE Joint Conference on Digital Libraries (pp. 322−330). Available at:: http://www.cs.berkeley.edu/~hoon/published/jcdl2001.pdf

Karpovich, J. F., French, J. C. & Grimshaw, A. S. (1994). High Performance Access to Radio Astronomy Data: A Case Study. UVA CS TR CS−94−25. Available at: ftp://ftp.cs.virginia.edu/pub/techreports/CS−94−25.ps.Z

Karpovich, J. F., Grimshaw, A. S. & French, J. C. (1994). Extensible File Systems (ELFS): An Object−Oriented Approach to High Performance File I/O. Proceedings of the Ninth Annual Conference on Object−Oriented Programming Systems, Languages and Applications (pp. 191−204). (Also available as UVA CS Technical Report CS−94−28; ftp://ftp.cs.virginia.edu/pub/techreports/CS−94−28.ps.Z)

Knight, J. & Hamilton, M. (1996). MIME Implementation for the Warwick Framework. Available at: http://www.roads.lut.ac.uk/MIME−WF.html

Kohl, U., Lotspiech, J. & Kaplan, M. A. (1997). Safeguarding Digital Library Contents and Users. D−Lib Magazine, 3(9). Available at: http://www.dlib.org/dlib/septemeber97/ibm/lotspiech.html

Lagoze, C. & Ely, D. (1995). Implementation Issues in an Open Architectural Framework for Digital Object Services. Cornell University Computer Science Technical Report TR95−1540. Available at: http://ncstrl.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR95−1540.

Lagoze, C., McGrath, R., Overly, E. & Yeager, N. (1995). A Design for Inter−Operable Secure Object Stores (ISOS). Cornell University Computer Science Technical Report TR95−1558. Available at: http://cs−tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell/TR95−1558

Lagoze, C. (1996). The Warwick Framework: A Container Architecture for Diverse Sets of Metadata. D−Lib Magazine, 2(7). Available at: http://www.dlib.org/dlib/july96/lagoze/07lagoze.html

Lagoze, C., Lynch C. A., & Daniel, R. Jr. (1996). The Warwick Framework: A Container Architecture for Aggregating Sets of Metadata. Cornell University Computer Science Technical Report TR96−1593. Available at: http://ncstrl.cs.cornell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR96−1593

Lawrence, S., Giles, C. L. & Bollacker, K. (1999). Digital Libraries and Autonomous Citation Indexing. IEEE Computer, 32(6), 67−71. Available at: http://www.neci.nj.nec.com/homepages/lawrence/papers/aci−computer98/aci−computer99.html

Lynch, C. (2001). The Battle to Define the Future of the Book in the Digital World. First Monday, 6(6). Available at: http://www.firstmonday.dk/issues/issue6_6/lynch/

McDonough, J., Myrick, L. & Stedfeld, E. (2001). Report on the Making of America II DTD Digital Library Federation Workshop. Available at: http://www.diglib.org/standards/metssum.pdf

Maly, K., Nelson, M. L., & Zubair, M. (1999). Smart Objects, Dumb Archives: A User–Centric, Layered Digital Library Framework. D–Lib Magazine, 5(3). Available at: http://www.dlib.org/dlib/march99/maly/03maly.html

Marazakis, M., Papadakis, D. & Nikolaou, C. (1997). Developing Network–Centric Applications by Dynamic Composition of Components. Institute of Computer Science, FORTH Technical Report 213. Available at: http://www.ics.forth.gr/pleiades/projects/Aurora/publications/auroraTR.ps.gz

Marazakis, M., Papadakis, D. & Papadakis, S. A. (1998). A Framework for the Encapsulation of Value–Added Services in Digital Objects. Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries – ECDL '98 (pp. 75–94) Available at: http://www.ics.forth.gr/pleiades/projects/Aurora/publications/ecdl98–final.ps.gz

Miller, E. (1998). An Introduction to the Resource Description Framework. D–Lib Magazine, 4(5). Available at: http://www.dlib.org/dlib/may98/miller/05miller.html

Mori, R. & Kawahara, M. (1990). Superdistribution: The Concept and the Architecture. Transactions of the IEICE, E73(7). Available at: http://www.virtualschool.edu/mon/ElectronicProperty/MoriSuperdist.html

Nelson, M. L., Maly, K. & Shen, S. (1997). Buckets, Clusters and Dienst. NASA TM–112877. Available at: ftp://techreports.larc.nasa.gov/pub/techreports/larc/1997/tm/NASA–97–tm112877.ps.Z

Nelson, M. L. (2000). Buckets: Smart Objects for Digital Libraries. PhD Dissertation, Computer Science Department, Old Dominion University. Available at: http://mln.larc.nasa.gov/~mln/phd/

Nelson, M. L. & Maly, K. (2001). Smart Objects and Open Archives. D–Lib Magazine, 7(2). Available at: http://www.dlib.org/dlib/february01/nelson/02nelson.html

Nikolaou, C., Marazakis, M., Papadakis, D., Yeorgiannakis, Y. & Sairamesh, J. (1997). Towards a Common Infrastructure for Large–scale Distributed Applications. Proceedings of the 1st European Conference on Digital Libraries (pp. 173–193). Available at: http://www.ics.forth.gr/pleiades/projects/Aurora/publications/euroDL97.ps.gz

OEPS. (2001). Open eBook Publication Structure 1.0.1. Available at: http://www.openebook.org/oebps/oebps1.0.1/download/

Payette, S. & Lagoze, C. (1998). Flexible and Extensible Digital Object and Repository Architecture (FEDORA). Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries – ECDL '98 (pp. 41–60). Available at: http://www.cs.cornell.edu/payette/papers/ECDL98/FEDORA.html

Payette, S., Blanchi, C., Lagoze, C. & Overly, E. A. (1999). Interoperability for Digital Objects and Repositories: The Cornell/CNRI Experiments. D–Lib Magazine, 5(5). Available at: http://www.dlib.org/dlib/may99/payette/05payette.html

Payette, S. & Lagoze, C. (2000a). Policy–Carrying, Policy–Enforcing Digital Objects. Proceedings of the Fourth European Conference on Research and Advanced Technology for Digital Libraries – ECDL 2000 (pp. 144–157). Available at: http://www.cs.cornell.edu/payette/papers/ECDL2000/pcpe–draft.ps

Payette, S. & Lagoze, C. (2000b). Value–Added Surrogates for Distributed Content Establishing a Virtual Control Zone. D–Lib Magazine, 6(6). Available at: http://www.dlib.org/dlib/june00/payette/06payette.html

Phelps, T. A. (1998). Multivalent Documents: Anytime, Anywhere, Any Type, Every Way User–Improvable Digital Documents and Systems. PhD Dissertation, Computer Science Division, University of California – Berkeley. Available at: http://www.cs.berkeley.edu/~phelps/papers/dissertation–abstract.html

Phelps, T. A. & Wilensky, R. (1996). Multivalent Documents: Inducing Structure and Behaviors in Online Digital Documents. Proceedings of the First ACM International Conference on Digital Libraries (pp. 100–108). Available at: http://www.cs.berkeley.edu/~phelps/papers/dl96–abstract.html

Phelps, T. A. & Wilensky, R. (1997). Multivalent Annotations. Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries – ECDL '97, (pp. 287 – 303). Available at: http://www.cs.berkeley.edu/~phelps/papers/edl97–abstract.html

Phelps, T. A. & Wilensky, R. (2000). Multivalent documents. Communications of the ACM, 43(6), 83–90. Available at: http://www.acm.org/pubs/citations/journals/cacm/2000–43–6/p82–phelps/

Ramanujapuram, A. & Prasad, R. (1998). Digital Content and Intellectual Property Rights. Dr. Dobbs Journal, 292, 20–27.

Roscheisen, M., Mogensen, C. & Winograd, T. (1997). Shared Web Annotations as a Platform for Third–Party Value–Added, Information Providers: Architecture, Protocols, and Usage Examples. Stanford University Technical Report CS–TR–97–1582. Available at: ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/97/1582/CS–TR–97–1582.pdf

Roscheisen, M., Winograd, T. & Paepcke, A. (1997). Content Ratings, and Other Third–Party Value–Added Information: Defining an Enabling Platform. Stanford University Technical Report CS–TN–97–40. Available at: ftp://reports.stanford.edu/pub/cstr/reports/cs/tn/97/40/CS–TN–97–40.pdf

Shklar, L., Makower, D., Maloney, E. & Gurevich (1998). An Application Development Framework for the Virtual Web. Proceedings of the Fourth International Conference on Information Systems, Analysis, and Synthesis, Orlando, FL. Available at: http://www.cs.rutgers.edu/~shklar/isas98/

Sibert, O., Bernstein, D. & Van Wie, D. (1995). DigiBox: A Self–Protecting Container for Information Commerce. Proceedings of the First USENIX Workshop on Electronic Commerce, New York, NY. Available at: http://www.usenix.org/publications/library/proceedings/ec95/sibert.html

Sibert, O., Horning, J. & Owicki S. (1997). A Massively Distributed Trusted System. Work–in–Progress Session 16th ACM Symposium on Operating System Principles. Available at: http://www.star–lab.com/talks/massively–distributed.html

SMIL (2001). Synchronized Multimedia Integration Language (SMIL 2.0). W3C Recommendation. Available at: http://www.w3.org/TR/smil20/

Staples, T. & Wayland, R. (2000). Virginia Dons FEDORA: A Prototype for a Digital Object Repository, D–Lib Magazine, 6(7/8), Available at: http://www.dlib.org/dlib/july00/staples/07staples.html

Unidata NetCDF (2001). Unidata NetCDF. Available at: http://www.unidata.ucar.edu/packages/netcdf/

Van Heyningen, M. (1994). The Unified Computer Science Technical Report Index: Lessons in Indexing Diverse Resources. Proceedings of the Second International World Wide Web Conference (pp. 535 – 543). Available at: http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Day/vanheyningen/paper.html

Waugh, A., Wilkinson, R., Hills, B., & Dollar, J. (2000). Preserving Digital Information Forever. Proceedings of the Fifth ACM Conference on Digital Libraries (pp. 175−184). Available at: http://www.acm.org/pubs/contents/proceedings/dl/336597/

Weibel, S. (1995). Metadata: The Foundations of Resource Description. D−Lib Magazine, 1(1). Available at: http://www.dlib.org/dlib/July95/07weibel.html

Zand, M., Collins, V. & Caviness, D. (1995). A Survey of Current Object−Oriented Databases. DATA BASE 26(1), 14−29.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
December 2001

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
A Survey of Complex Object Technologies for Digital Libraries

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael L. Nelson, Brad Argue, Miles Efron, Sheila Denn, and Maria Cristina Pattuelli

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-18146

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA/TM-2001-211426

11. SUPPLEMENTARY NOTES
Nelson: Langley Research Center, Hampton, VA; Argue, Efron, Denn, and Pattuelli: University of North Carolina, Chapel Hill, NC.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 82 Distribution: Standard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many early web-based digital libraries (DLs) had implicit assumptions reflected in their architecture that the unit of focus in the DL (frequently "reports" or "e-prints") would only be manifested in a single, or at most a few, common file formats such as PDF or PostScript. DLs have now matured to the point where their contents are commonly no longer simple files. Complex objects in DLs have emerged from in response to various requirements, including: simple aggregation of formats and supporting files, bundling additional information to aid digital preservation, creating opaque digital objects for e-commerce applications, and the incorporation of dynamic services with the traditional data files. We examine a representative (but not necessarily exhaustive) number of current and recent historical web-based complex object technologies and projects that are applicable to DLs: Aurora, Buckets, ComMentor, Cryptolopes, Digibox, Document Management Alliance, FEDORA, Kahn-Wilensky Framework Digital Objects, Metadata Encoding & Transmission Standard, Multivalent Documents, Open eBooks, VERS Encapsulated Objects, and the Warwick Framework.

14. SUBJECT TERMS
Digital Libraries, Digital Objects, Complex Objects

15. NUMBER OF PAGES
77

16. PRICE CODE
A05

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT