Advanced Distributed Learning
Emerging and Enabling Technologies for the Design of Learning Object Repositories Report

Version 1.0

Advanced Distributed Learning Initiative.
1901 N. Beauregard Street
Alexandria, VA 22311

November 26, 2002

This page intentionally left blank.

Authors

Dr. Thelma Looms, Principal Learning Technologies Architect, ADL

Clark Christensen, Lead Software Engineer, ADL

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This page intentionally left blank.

Section 1: About this Document

The Department of Defense (DoD) established the Advanced Distributed Learning (ADL) Initiative in 1997 to develop a DoD-wide strategy for using learning and information technologies to transform education and training and to promote cooperation between government, academia and business to develop e-learning standardization. The ADL Initiative has defined high-level requirements ("-ilities") for learning content, such as content reusability, accessibility, durability and interoperability to leverage existing practices, promote the use of technology-based learning and provide a sound economic basis for investment.

The ADL Initiative is preparing for a world where communications networks and personal delivery devices are pervasive and inexpensive, as well as transparent to the users in terms of ease of use, bandwidth and portability. ADL development envisions the creation of learning "knowledge" libraries, or repositories where learning objects may be accumulated and cataloged for broad distribution and use. These objects must be readily accessible across the World Wide Web or whatever forms our global information network takes in the future.

This report focuses on major standards for networked repository architectures and other important infrastructure technologies that may be useful for managing SCORM conformant content. It does not provide an authoritative view, but rather focuses on those elements most likely to be of interest in future development of a SCORM Repository Application Profile.

This page intentionally left blank.

Section 2:   Introduction

The advent of the Internet enabled communication, publishing and collaboration for individuals and organizations over a worldwide computer network. Due to limits in bandwidth, PC storage capacity and processor power, as well as the availability of tools for content publishing, initial interactions over the Internet centered mainly on text-based applications and technologies (e.g. HTML, Web Pages, News Groups, Internet Relay Chat, etc.).

The potential for media-rich interactive applications and content increased along with computing power and network bandwidth. Important technical infrastructure and standards have developed, such as the W3C Extensible Markup Language Standard [XMLB01] for document structure, W3C Simple Object Application Protocol [SOMF02], [SOAD02] as a specification for computer communication, the Motion Picture Experts Group (MPEG) standards for video compression and delivery, etc.

These developing technologies provided the underlying infrastructure necessary to package, deliver and present learning content in new ways. In this environment, the Office of the Secretary of Defense (OSD), along with the Office of Science and Technology Policy (OSTP), launched the Advanced Distributed Learning (ADL) Initiative. With the establishment of ADL and the subsequent release of the Sharable Content Object Reference Model (SCORM^{TM}), ADL liberated learning content objects from local implementations and facilitated global accessibility.

The inclusion of a content packaging specification in the SCORM Version 1.2 led to an increased focus on the development of SCORM conformant content. Government organizations are implementing strategies for developing SCORM conformant content. Many now specify SCORM conformance as a requirement for new educational and training content. The DoD is currently developing guidance that will address SCORM conformance for new learning content as a matter of policy.

There has recently been much effort to develop technologies that catalogue and facilitate the location of text, images, video and audio. Current efforts in the development of repository standards and software are broad and varied, with players coming from nearly every major sector. This diversity of interest has resulted in an impressive number of competing standards and supporting technologies.

Repository systems provide key infrastructure for the development, storage, management, discovery and delivery of all types of electronic content. SCORM content packaging, with its inclusion of mandatory self-descriptive meta-data, plays an important role enabling advanced functionality for repository systems. As such, SCORM packaging plays a key role in accomplishing ADL's "-ilities" and in advancing the repository state-of-the-art.

This report provides descriptions of several standards-based repository architectures and technologies, gleaned from an in-depth review of existing documentation and implementations. It does not evaluate the reviewed systems for fitness of any specific purpose, but rather provides an informational starting point from which the ADL Technical Team can begin assessing the development of a repository application profile for SCORM content.

Section 3: Major Organizations

Many organizations are seeking ways to facilitate the reuse of learning objects. The following sections describe some of the major organizations and institutions working with learning object repositories and registries. Vendors such as Sun Microsystems, EMC Corporation and Artesia have developed hardware platform and software solutions to support the storage and retrieval of digital assets. Several academic institutions have developed and deployed distributed learning object repository networks (DLORN) using peer-to-peer protocols. These include: the Technical University of British Columbia (the POOL project [RICH01]), University of Calgary (CAREO system [CANC02]), University of North Carolina at Chapel Hill (iLumina project [MCAR01]) to name a few..

While there are a number of organizations developing software for repository and registry systems, the focus of this section is on organizations working on repository-related standards including work on authorization, object identification, learning object meta-data (LOM) and messaging protocols. The work on these initiatives will be discussed below.

3.1 E-learning Vendors and Academic Institutions

Table 1 shows the major e-learning vendors and academic institutions that are involved in the development of repository and registry systems. The following sections describe the major organizations that are involved in repository-related standards and technologies.

E-learning Vendors	Institutions
Artesia	Cornell University
IBM	National Science Foundation
Sun Microsystems	Old Dominion University
EMC	Simon Frasier University
Learning Objects Network	University of Alberta
Microsoft Corporation	University of Calgary
Digital Concepts, Inc.	University of Wisconsin

Table 1.   **Examples of vendors and academic institutions involved in digital asset and meta-data repository efforts.**

3.2   IMS Global Learning Consortium, Inc.

The IMS Global Learning Consortium, Inc. came into existence as a project within the National Learning Infrastructure Initiative of EDUCAUSE. IMS collaborates with a number of organizations, including the Advanced Distance Learning Initiative (ADL), the World-Wide Web Consortium (W3C), the Institute of Electrical and Electronic Engineers (IEEE), the Aviation Industry Computer-Based Training (CBT) Committee (AICC) and the National Institute of Standards and Technology (NIST). IMS defines and delivers XML-based specifications for exchanging e-learning content and information about learners. Specifications that have already been released include Content Packaging, Question and Test Interoperability, Meta-Data and Learner Information Packaging. Project Groups within IMS are in the process of developing specifications for Competency, Accessibility, Learning Design, Digital Repositories and Simple Sequencing.

3.3   World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C) is a major forum for the development of specifications and the underlying technologies that are used on the Internet and the World Wide Web. Started in 1994 at the Massachusetts Institute of Technology (MIT), the consortium is hosted by the Laboratory for Computer Science at MIT, Institut National de Recherche en Informatique et en Automatique INRIA and Keio University. The consortium also receives support from the Defense Advanced Projects Agency (DARPA) and the European Commission. The W3C has released over forty specifications (called "recommendations"), which include the Simple Object Access Protocol (SOAP), the eXtensible Markup Language (XML) and eXtensible Stylesheet Language (XSL).

3.4   International Organization for Standardization / International Electrotechnical Commission (ISO/IEC)

The International Organization for Standardization (ISO) was founded in 1947. ISO is an international network of standards organizations from 140 countries responsible for over 13,000 published international standards. The International Electrotechnical Commission (IEC) was founded in 1908 and provides standards for all electrotechnologies including electronics, magnetics and electromagnetics, electroacoustics, multimedia and telecommunications. JTC1 is a Joint Technical Committee of ISO and IEC that develops standards on Information Technology. ISO/IEC JTC1 SC32 WG2 is the Metadata Working Group of the Data Management and Interchange subcommittee (SC32). The Working Group is supported by 17 member nations and develops international standards that facilitate the specification and management of meta-data. These include the multi-part standard on meta-data registries (ISO/IEC 11179). The parts of this standard provide a framework and conceptual model for meta-data and meta-data registries including how data should be defined and how it is entered into a registry.

3.5 Organization for the Advancement of Structured Information Standards (OASIS)

OASIS, founded in 1993, was originally started by a consortium of vendors and users devoted to developing guidelines for interoperability among products that support the Standard Generalized Markup Language (SGML). It operates XML.org, a community clearinghouse for XML application schemas, vocabularies and related documents. OASIS produces standards for security, Web services, XML conformance, business transactions and electronic publishing. In particular, the OASIS ebXML Registry Technical Committee develops specifications to achieve interoperable registries and repositories that cover the spectrum from general purpose document registries to real-time business-to-business registries.

3.6 International Digital Enterprise Alliance (IDEAlliance)

IDEAlliance (International Digital Enterprise Alliance) is a not-for-profit membership organization. Founded in 1966 as the Graphic Communications Association (GCA), its mission is to advance user-driven, cross-industry solutions for all publishing and content-related processes by developing standards and identifying best practices. IDEAlliance has participated in the development and adoption of standards such as the Information Content and Exchange (ICE) and the Publishing Requirements for Industry Standards Metadata (PRISM).

3.7 The International DOI Foundation (IDF)

Founded in 1998, the main activity of the International DOI Foundation (IDF) is to encourage the implementation and use of a standard digital identifier methodology for intellectual property, the Digital Object Identifier (DOI). The foundation is funded by a variety of technology and publishing companies and manages the development of the DOI system [PASK02]. In addition, the foundation is responsible for: licensing directory managers, registration agencies and technology providers; setting system policy; encouraging development of enabling technologies necessary to build electronic transaction systems for copyright and digital rights management (DRM).

This page intentionally left blank.

Section 4: Enabling Technologies

Repositories must provide a basic set of functions in order to provide access to learning objects and other assets in a secure environment. The following list of operations presents a sampling of common core functionality gleaned from the different standards outlined this section.

- *search/find* – the ability to locate an appropriate learning object. This can include the ability to browse
- *request* – a learning object that has been located
- *retrieve* – receive an object that has been requested
- *submit* – provide an object to a repository for storage
- *store* – place a submitted object into a data store with a unique, registered identifier that allows it to be located
- *gather (push/pull)* – obtain meta-data about objects in other repositories for federated searches and information clearinghouse
- *publish* – provide meta-data to other repositories

In addition, a repository must handle a number of other issues including DRM, obtaining a globally unique identifier for each learning object or other asset and providing authentication for secure access to existing learning objects.

A number of technologies have been developed to enable repository systems to provide these and other functions. The initiatives that provide some of these enabling technologies are discussed in the following sections.

4.1   ISO/IEC 11179, Information Technology – Metadata Registries

The ISO/IEC 11179, Information Technology – Metadata Registries standard [ISOM02] was written to address issues associated with sharing data between different organizations. It was produced by the Metadata working group of Sub-Committee 32 (SC32/WG2) of the ISO JCT1and the IEC – ISO/IEC JCT1 SC32/WG2. ISO/IEC 11179 defines how data elements can be described, identified and stored in sharable registries. Data elements could be learning objects or media assets and the meta-data used to describe them. Table 2 provides a quick "at a glance" summary for ISO/IEC 11179 including institutions involved, versions and supported functionality, and implemented reference models.

ISO/IEC 11179, Information Technology – Meta-data Registries at a Glance		
Institution	**Version**	**Reference Models**
http://www.iso.org International Organization for Standardization Geneva, Switzerland http://www.iec.ch International Electrotechnical Committee Geneva, Switzerland http://metadata-stds.org ISO/IEC JCT1 SC32/WG2 Metadata Standards Working Group	The six parts of the version 1.0 draft specification are in various stages of development. ISO/IEC 11179 supports: ▪ Standards for data element representation ▪ Unambiguous descriptions of data elements ▪ Globally unique identifiers for data elements	U.S. Environment Protection Agency (EPA) Environmental Data Registry (http://www.epa.gov/edr) U.S. Environment Protection Agency (EPA) MetaPro Metadata Registry Builder (http://www.epa.gov/metapro/) U.S. Federal Aviation Administration (FAA) Data Registry (http://www1.faa.gov/aio/InfoMgmt/projects.htm#FAAdatreg) Australian Institute of Health and Welfare Knowledgebase (http://www.aihw.gov.au/knowledgebase/index.html)

Table 2. **ISO/IEC 11179 at a glance**

ISO/IEC 11179 provides important foundational meta-data registry work for other standards. In particular it serves building block for OASIS registry work including ebXML (described later in this section). This relationship is shown in Figure 1.

Figure 1. **Standards evolving from ISO/IEC 11179.**

Technologies for the Design of Learning Object Repositories

021751

ISO/IEC 11179 contains six parts that describe rules and guidelines for classifying and describing data elements, data naming and identification principles and procedures for storing data elements in a meta-data register (repository) for access in a standardized manner. Each part assists in a different aspect of data element formulation and should be used in conjunction with the other parts.

Part 1 (ISO/IEC 11179-1, *Framework)* establishes the relationships among the parts and gives guidance on their usage as a whole. It also provides definitions of the basic concepts associated with data elements. Data elements in an ISO/IEC 11179 registry are defined as having three components: an *object class*, *properties* and a *representation*.

- An *object class* is a set of ideas or things that can be identified with explicit meaning and whose properties follow the same rules. They are things about which data can be collected and stored. Object classes can be formed by combining other object classes. Examples of object classes are cars, orders and employees.

- *Properties* are characteristics common to the elements in an object class and are used to describe and differentiate between specific objects. Examples of properties include color, age, size, price, etc.

- The *representation* describes how the data is represented. It includes a value domain (the set of permissible/valid values), a data type and, if necessary, a unit of measure.

An object class and a property can be combined to form a *data element concept* that is independent of any particular representation. For example, annual household income is a data element concept, while a particular annual household income (e.g., $48,000) is a data element.

Part 2 (ISO/IEC 11179-2, *Classification for Data Elements*) specifies a set of mandatory meta-data items that shall be provided for each data element. In addition, a list of potential additional items and the detailed characteristics of each basic attribute are described. Part 2 also provides procedures and techniques for associating data element concepts and data elements with classification schemes for object classes, properties and representations. This is necessary to retrieve data element concepts and data elements from a registry.

Part 3 (ISO/IEC 11179-3, *Basic Attributes of Data Elements*) specifies the basic attributes of data elements. Distinctions are made between attributes that are:

- *Mandatory* – attributes that are required for a data element to be registered
- *Conditional* – attributes that are required under specified conditions
- *Optional* – attributes that are allowed but not required

Two of the mandatory attributes are described in extensive detail in parts 4 and 5, respectively data element definitions and element names.

Part 4 (ISO/IEC 11179-4, *Rules and Guidelines for the Formulation of Data Definitions*) provides a number of rules and guidelines that specify exactly how an unambiguous data element definition should be formed. The ability to provide clear, well-formed definitions of data elements is essential for the exchange of information.

Part 5 (ISO/IEC 11179-5, *Naming and Identification Principles for Data Elements*) provides guidance on the naming and identification of data elements. Names are semantic, natural language labels that are given to data elements. One structured naming scheme is based on the object class, property and representation of an individual data element. This type of name can help a user intuitively locate a data element in a taxonomy, however, it does not dependably produce a unique identifier for the data element. IOS/IEC 11179 does not specify the content or format of identifiers beyond stating that identifiers must be unique within a Registration Authority (RA).

Part 6 (ISO/IEC 11179-6, *Registration of Data Elements*) provides instruction on how a data element can be registered. It also describes the allocation of unique identifiers for each data element and the maintenance of data elements that have already been registered. Data elements are registered by a submitting organization (which may be separate from the organization that is responsible for the content of the data element) with a RA. The RA is an organization that has been authorized to register data elements. When a data element is registered, a unique identifier is assigned by the RA. This identifier is determined by the combination of the identifier for the RA, the unique identifier assigned to a data element within the RA, and the version of the data element.

Government Reference Implementations of ISO/IEC 11179

Several organizations have used ISO/IEC 11179 as the basis for data or meta-data registry/repository systems. Some of these have produced reference implementations of the standard such as the MetaPro Metadata Registry Builder [EPAM02] and Environmental Data Registry [EPAE02] produced by the Environmental Protection Agency (EPA) and the Australian Institute of Health and Welfare's Knowledgebase [AIHW02]. Other agencies, including the Federal Aviation Administration (FAA) [FAAD02] and the U.S. Bureau of the Census [GILL99], are using the Data Standards from ISO/IEC 11179, Part 1 as the model for representing data in systems they are developing.

Work is also being done to incorporate ISO/IEC 11179 into other standards. The National Committee on Information Technology Standards Technical Committee L8, Data Representation (NCITS L8) establishes standards for specifying and standardizing data. The focus of the committee's work is on establishing ways to describe data to enable intelligent computer processing. Meta-data issues addressed by the committee include naming, identification, definitions, classification and registration. NCITS L8 is concerned with the development of proposed standards (notably ISO/IEC 11179) that will facilitate the standardized naming, definition and description of data elements [NCIT98].

4.2 IMS Digital Repository Interoperability

The IMS Global Learning Consortium is a specification authoring organization with headquarters in Burlington, Massachusetts. The IMS Digital Repository Interoperability (IMS DRI) model is the product of the IMS Digital Repository Working Group. The goal of the IMS DRI is to provide repository technology to support the "presentation, configuration and delivery of learning objects." The IMS DRI Version 1.0 Public Draft Specification [IMSG02] was approved in August of 2002. The specification is comprised of three documents, the *IMS Digital Repositories Interoperability Information Model* which defines the information model, describes the core functions, *IMS Digital Repositories Interoperability Best Practice and Implementation Guide* and the *IMS Digital Repositories Interoperability XML Binding*. Table 3 provides a quick "at a glance" summary for the IMS DRI effort.

IMS Digital Repository Interoperability at a Glance		
Institution	**Version**	**Reference Models**
http://www.imsglobal.org IMS Global Learning Consortium Burlington, MA	Version 1.0 (Public Draft Specification – 8/9/2002) IMS DRI supports: • User searches performed directly, through a "gateway" or through a harvest intermediary (aggregator). • The storage and retrieval of IMS-compliant Content Packages. • XQuery/SOAP-based and Z39.50-based implementations. • The storage of assets and the associated meta-data in separate repositories.	Learning Objects Network

Table 3.   IMS Digital Repository Interoperability at a glance.

The *IMS Digital Repositories Interoperability - Core Functions Information Model* [IMSG02] defines the information model, describes the core functions (described below) and provides eight use cases. These use cases describe scenarios including:

- an individual submitting a course to a repository
- an individual modifying a course in a repository
- an individual searching a repository, then requesting a discovered resource
- an individual searching multiple repositories, then requesting a discovered resource
- a software agent searching a repository, then requesting a discovered resource
- an aggregator repository gathering meta-data from multiple repositories and populating its own repository
- an individual being alerted when a specified meta-data change occurs in a single repository
- an individual being alerted when a specified meta-data change occurs in multiple repositories

The *IMS Digital Repositories Interoperability - Core Functions Best Practices* [IMBP02] document describes technology recommendations for implementing the core functions using technologies such as XML Query Language (XQuery), Simple Object Access Protocol (SOAP), or ANSI/NISO Z39.50 (Z39.50) information retrieval protocol[1]. It also provides example recommendations for several topics including: requesting and retrieving IMS Content Packages using Z39.50 and XQuery, storing IMS Content Packages and performing cross-domain searches using IMS meta-data (i.e. meta-data defined by the IMS Learning Resource Specification).

The *IMS Digital Repositories Interoperability - Core Functions XML Binding* [IMXB02] document describes the use of SOAP as the underlying messaging service for the core functions and provides scenarios illustrating the XML bindings for:

- searching learning object repositories using the XQuery protocol over XML meta-data, adhering to the IMS Meta-Data Schema with IMS Content Packaging as the format for Submit/Store.

- searching General Repositories of resources not purposed specifically for learning. This search assumes the use of Z39.50 protocol for searching, with no provision for Submit/Store.

- Performing a Cross-Domain Search that assumes simple keyword searching using the Boolean operators AND, OR, and ANDNOT over a flattened schema of IMS meta-data elements.

The recommendation does not specify how a repository functions internally, but defines how a repository exposes itself to the outside world.

The IMS DRI is included in this review because the specification outlines functions that are generic to any repository implementation such as authentication, authorization, enrollment, location and retrieval, intellectual property rights (IPR) management, user preferences and profiling, transactions and search as shown in Figure 2. The IMS DRI model provides recommendations for the interoperability of a set of "core" repository functions.

---

[1] SOAP and XQuery are discussed in sections 4.1 and 4.2, respectively. ANSI/NISO Z39.50 is the American National Standard Information Retrieval Application Service Definition and Protocol for Open Systems Interconnection. Z39.50-1988 or version 1 is an American National Standards Institute (ANSI) standard, Z39.50-1992 or version 2 is a National Information Standards Organization (NISO) revision of the original ANSI standard. [MOEN02]

These core functions include:

- *Search/Expose*
- *Gather/Expose*
- *Submit/Store*
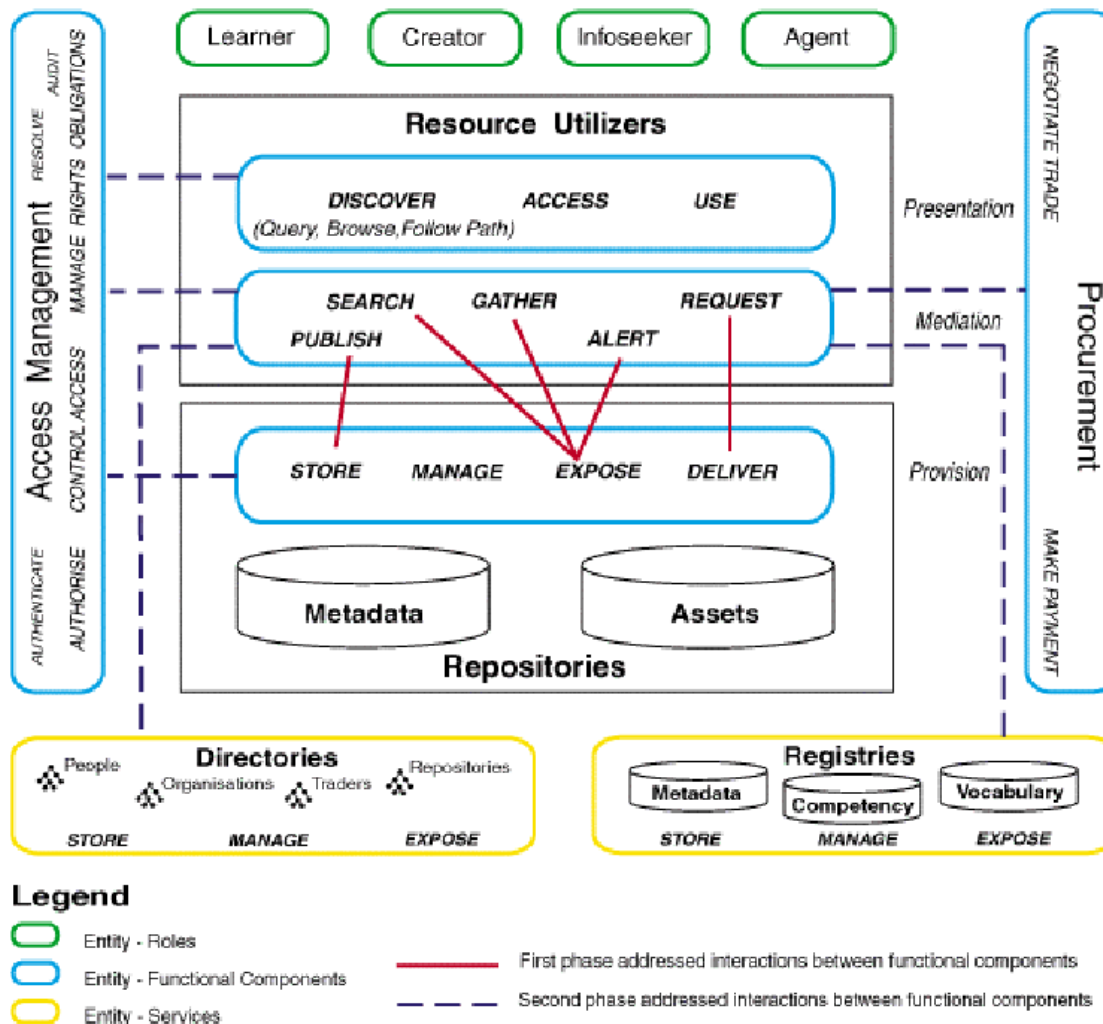- *Request/Deliver*
- *Alert/Expose*[2]

Figure 2.   **IMS Digital Repository Interoperability functional architecture. Reproduced from IMS Global Learning Consortium, 2002.**

---

[2] To be addressed in a future DRI specification

Technologies for the Design of Learning Object Repositories
021751

The DRI specification takes into account the needs of existing digital repositories including different content formats, technologies and practices. It is intended for both systems utilizing established interoperability technologies such as Z39.50 and repositories than can use the XQuery and SOAP recommendations provided in the specification. These recommendations describe a collection of resources for exposing meta-data to allow users to search, gather, store and deliver assets. An IMS DRI repository may contain actual assets or meta-data describing the assets. An asset and its meta-data do not need to reside in the same repository.

Overview of IMS DRI Functions

The `Search` function defines the searching of meta-data for assets "exposed" by repositories. A repository can be searched directly or using an intermediate search engine. XQuery, discussed in section 4.2, is used when searching meta-data in the IMS XML format, while Z39.50 is used for searching library information. Cross-domain queries search repositories containing different types of meta-data. One possible method for implementing cross-domain searches would rely on an intermediary that would translate a query, based on a subset of the XQuery grammar, into the appropriate syntax for one or more target repositories. Repositories from non-IMS domains could provide additional search attributes (Dublin Core or IMS) to facilitate searching from IMS-based learning environments.

The `Gather` function allows the aggregation of meta-data from repositories for use in subsequent searches. The `Gather` function may actively request meta-data from a repository ("pull") or it can subscribe to a service that notifies the `Gather` component when meta-data in the repository has been added, deleted or changed ("push").

The Open Archive Initiative (OAI) described below provides a model for "pull" gathering. In this model, a Gather Engine periodically searches a set of target repositories and retrieves meta-data based on a range of dates. Using date as the primary criterion for retrieval is effective in harvesting all the meta-data that has been added or modified. However, it requires that a new element be defined in the IMS Meta-data Specification to provide the date information that allows the Gather Engine to determine which meta-data to harvest. Another option for Pull Gather is to periodically harvest all the meta-data from all the target repositories. While this approach is very inefficient (records are repeatedly pulled over the network), it assures a very high level of completeness.

Push Gather relies on repositories notifying specific aggregators each time meta-data is added or updated. The notice could be a message indicating that the meta-data had changed or it could be the actual meta-data record. An adapter that is external to the repository could also provide this functionality. The adapter could forward the meta-data to the aggregators as content is added to the repository. It could also handle any required message translations between the repository and the network.

`Submit/Store` refers to the way an object is moved to a repository and made accessible. `Submit` places an object into a repository. `Store` allows a repository to store the object so that it may be retrieved later. Existing repository systems may already have a submission mechanism such as File Transfer Protocol (FTP) in place. For newer repositories the `Submit` function could be performed using SOAP messages with attachments, where the attachments are IMS compliant Content Packages. For repositories that deal specifically with learning objects, the DRI refers to the IMS Content Packaging Specification. This specification defines a compressed file package that contains learning object(s), meta-data and a manifest listing the contents in the package.

The `Request/Deliver` functions allow a system user to request learning objects or other resources located with the `Search` function. The `Search` function returns repository object identifiers as a list of locations or as a method, such as a DOI, that resolves to one or more locations . The location returned by `Search` resolves to a URL that can then be used to `Request` the object. The protocol used to `deliver` a requested learning object depends on the object type. For example, online materials and streaming media would be delivered using HTTP, while documents and executable files would be delivered using FTP.

The `Alert/Expose` functions provide a method for notifying interested parties of any changes made to content stored in a repository or repository system. They are not considered in Phase 1 of the DRI specification. It is envisioned that the `Alert` function could be provided using a mechanism such as SMTP (Simple Mail Transfer Protocol) e-mail.

Learning Objects Network – IMS DRI Reference Implementation

The Learning Objects Network (LON) has developed a reference implementation of the IMS DRI core functions. The LON proof-of-concept implementation was evaluated as part of the ARTI in the summer of 2002. Figure 3 illustrates the proposed architecture that was developed in Python and Java and uses a Native XML database to store SCORM Meta-data.

Figure 3.   **Learning Objects Network (LON) architecture.**

The architecture was designed for the following components:

- a learning object registry
- a learning object repository
- an XML meta-data search engine
- a Client message broker

All the components interact with other components using a set of SOAP web services. The learning objects in the repository are identified using a unique identifier implemented using the IDF's Digital Object Identifier (DOI). The LON Message Manifest Specification [LONM02] describes a messaging API that supports the basic functions for a LON repository. The Message Manifest describes the structure and format of the messages exchanged by services within the repository and registry. Messages are defined for all the functions provided by the client and server components of the system. The following functions were demonstrated during the evaluation:

- RgRegister – make an entry in a registry
- RgResolve – resolve a registry entry to a unique repository ID
- RgUpdate – update a registry entry
- RpPut – put an object in a repository
- RpGet – get an object from a repository
- RpUpdate – update an object in a repository
- RpDelete – remove an object from a repository

Figure 4 shows the flow of a repository Put message (`RpPut`). This models the following simple transaction. In step one, an Authoring System contacts a Registry with a request to Register a new object. The Registry replies with a Unique Object Identifier (UOI) , in step two. In step three, the Authoring System puts the object into the Repository via the RpPut call. In response to the RpPut call the Repository returns a Unique Internal Identifier (UII) that the repository uses to identify the new object within that specific Repository, step four. In step five, the Authoring System combines the Repository UII with the Registry UOI and makes one final call to the Registry to store the two associated identifiers.

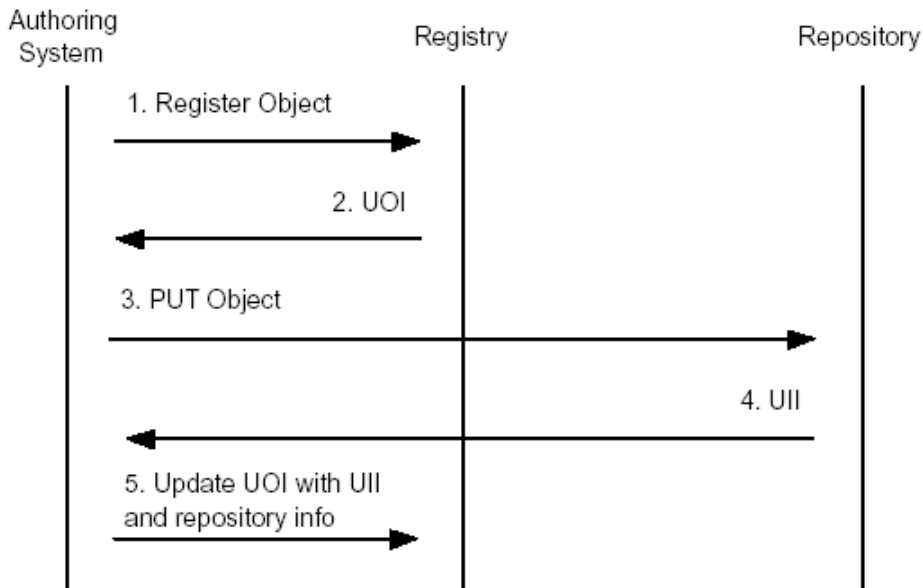**Figure 4. Flow for a RpPut repository message. Reproduced from Learning Object Networks, 2002.**

The results from the evaluation showed the core functions defined in the IMS Digital Repository Interoperability Information Model were successfully implemented and that the system could be used as the basis for an API to support the storage and retrieval of SCORM Version 1.2 conformant meta-data packages.

4.3   Open Archive Initiative

The ability to harvest and search repositories is an important requirement to provide delivery of content from distributed repositories. The Open Archive Initiative (OAI) [LAGO02] is a technical and organizational meta-data harvesting framework that is designed to facilitate the discovery of content stored in distributed archives. The Digital Library Federation, the Coalition for Networked Information (CNI), the National Science Foundation (NSF) and DARPA are the sponsors for the OAI. The OAI is designed for large collections of peer-reviewed information (e.g. scientific papers). The OAI is included in this review because the architecture includes many elements that can be applied to communication with a repository of SCORM content, and in the design and implementation of a clearinghouse portal through which a user or system may search repositories and meta-data to discover specific learning content. Table 4 provides a quick reference for the OAI effort.

OAI at a Glance		
Institution	**Version**	**Reference Models**
http://www.orpenarchives.org OAI Executive – Cornell University (CS) – Los Alamos National Library Technical Committee – NASA Langley Research Center – Cornell University (CS)	Version 1.1 (released 7/2001) Version 2.0beta (released 5/2002) OAI–PMH supports: • Unique identification of items in a repository. • Selective harvesting of meta-data in multiple formats. • HTTP GET or POST requests. • Error and exception handling.	List of OAI repositories (118 sites) http://www.openarchives.org/Register/BrowseSites.pl OAI Related Utilities http://www.openarchives.org/tools/tools.html MetaArchive.org http://www.metaarchive.org/ OAIster http://oaister.umdl.umich.edu/index.html

Table 4.   OAI at a glance.

The OAI–PMH Protocol

The Open Archives Meta-data Harvesting Protocol (OAI–PMH) is described in this section. OAI–PMH defines a mechanism to enable harvesting of meta-data from distributed repositories. OAI–PMH consists of two parts: the definition of a set of simple meta-data elements and a common protocol to enable extraction of document meta-data and archive-specific meta-data from participating repositories. Meta-data can be harvested from aggregated collections (meta-data from all source repositories) or from specific repositories. The protocol mandates that the core meta-data set is Dublin Core [DUBL02] however OAI supports "parallel" meta-data sets to allow repositories to expose meta-data in formats that are specific to their applications.

Two examples are the iLumina digital library project [MCAR01, MCLE02], which uses IMS meta-data to provide standardized description of the learning objects it manages and the Learning Object Virtual Exchange (LOVE) [CHEN02]. Several repositories harvest meta-data such as the Machine Readable Cataloging (MARC) commonly used by the library systems and the format for Bibliographic Records described in the Internet Engineering Task Force (IETF) RFC1807 [RFC1807]. Meta-data can be exchanged in any format as long as it is based on XML. The OAI–PMH does require that you expose meta-data as unqualified Dublin Core. For example, a repository interface might allow users to harvest "IMS or SCORM meta-data" from your repository, however to be OAI–PMH compliant, you must also provide an unqualified Dublin Core view of the meta-data as well. The reason that Dublin Core was chosen is to provide base level interoperability between services. However, adhering to compliancy is left up to the implementer. Figure 5 shows the LOVE architecture for harvesting meta-data from various sources.
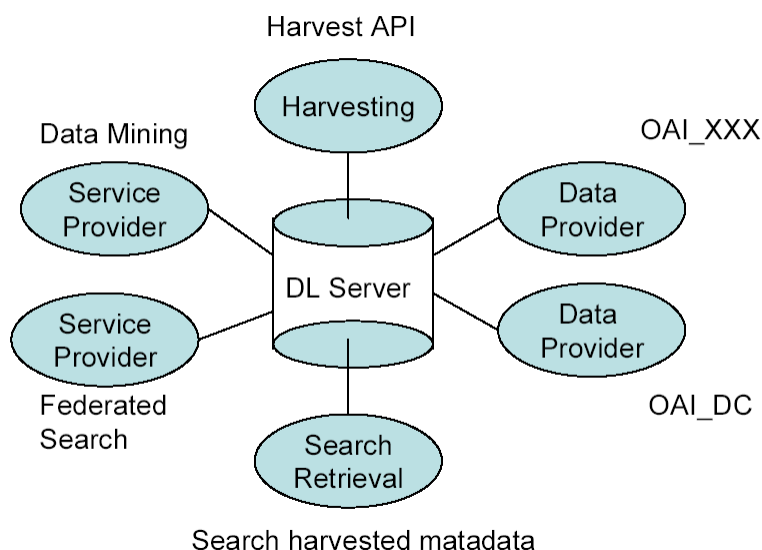
Figure 5. LOVE architecture to support OAI searching and harvesting. Reproduced from Chen and Choo. University of Florida, 2002.

Technologies for the Design of Learning Object Repositories
                                                                      021751

The OAI–PMH protocol is very simple and allows regular gathering of meta-data, and is based on Web standards such as HTTP, XML and XML Schema Definition (XSD). Implementations have been developed in Perl, VB, C, C++ and Java (see Table 4). The largest OAI compliant repository is located at Los Alamos (physics archives – arXiv.org). The following concepts are important to understanding the flow of information between OAI-PMH systems. Figure 1 provides a visual representation:

- A *repository* is a network accessible server that can process the six OAI requests (described later in this section) and is managed by a *data provider*.

- A *data provider* maintains one or more repositories (servers) and makes meta-data about its content available to a *service provider*. OAI maintains a list of registered data providers [OAIR02].

- A *service provider* gathers meta-data and makes it available for searching.

- An *OAI aggregator* is both a service provider and data provider. An *aggregator* gathers meta-data records from *data providers* and then makes them available for gathering by other services.

- An *item* is a container that stores (or dynamically generates) meta-data about a resource in multiple formats (e.g., IMS, MARC) and has a unique identifier within the scope of the repository in which it resides.

- A *record* is meta-data expressed in a single format. It is returned in response to an OAI–PMH request for meta-data for an item.

- A *unique identifier* is intended to provide persistent resource identifiers for items in repositories that implement OAI–PMH and is implemented as a Uniform Resource Identifier (URI) [BERN98].
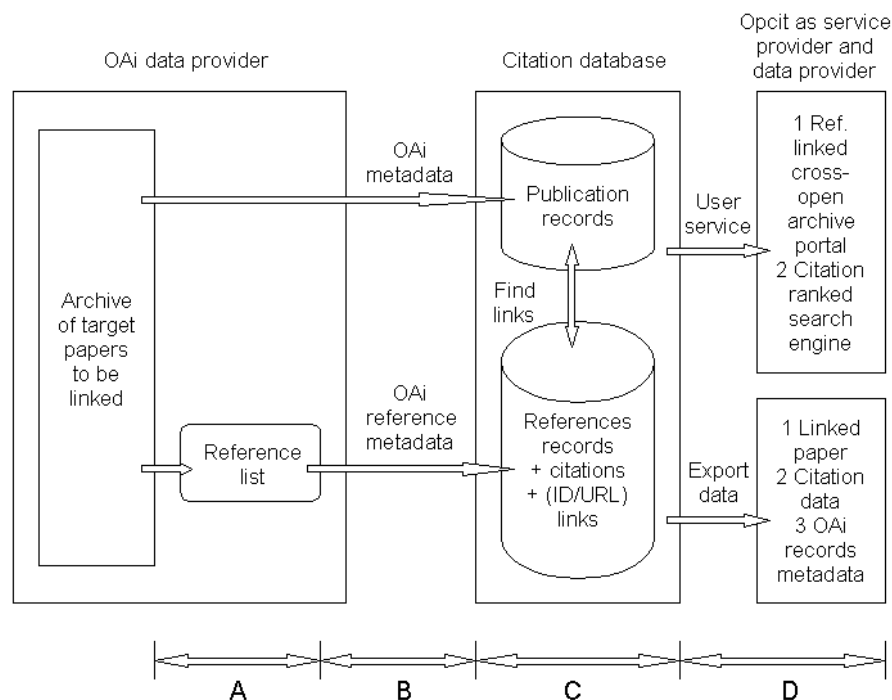
Figure 6. **The OAI Open Citation (OpCit) application displaying data inputs and outputs. Reproduced from Brody, Jiao, Hitchcock, Carr, and Harnad, University of Southampton, 2001.**

Harvesting Meta-data

The protocol supports six OAI requests that are used to harvest and request meta-data, validate and perform error processing. A detailed description of each command (know as *verbs*) can be found in [LAGO02]. The commands are implemented as keyword arguments. For example, the GetRecord request to a repository is used to retrieve an individual meta-data record. Using a base URL of http://a.oa.org/AI-script

for a HTTP GET request would be coded as:

http://arxiv.org/oai2?verb=GetRecord&identifier=oai%3AarXiv.
org%3Ahep-th%2F9901001&metadataPrefix=oai_dc

You can try this command in the address field of a Web browser and the following information will be returned. The dc:identifier field http://arXiv.org/abs/hep-th/9901001 would link a user to the actual abstract and paper for this meta-data record.

Technologies for the Design of Learning Object Repositories

```
<?xml version="1.0" encoding="UTF-8" ?>
- <OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
    <responseDate>2002-10-28T22:09:17Z</responseDate>
    <request verb="GetRecord" metadataPrefix="oai_dc"
      identifier="oai:arXiv.org:hep-th/9901001">http://arXiv.org/oai2</request>
  <GetRecord>
    <record>
      <header>
          <identifier>oai:arXiv.org:hep-th/9901001</identifier>
          <datestamp>2002-06-24</datestamp>
          <setSpec>physics:hep-th</setSpec>
        </header>
        <metadata>
          <oai_dc:dc
              xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
              xmlns:dc="http://purl.org/dc/elements/1.1/"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oa
              i_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
              <dc:title>String Junctions and Their Duals in Heterotic
                String Theory</dc:title>
              <dc:creator>Imamura, Yosuke</dc:creator>
              <dc:description>We explicitly give the correspondence
                between spectra of heterotic string theory
                compactified on $T^2$ and string junctions in type
                IIB theory compactified on $S^2$.</dc:description>
              <dc:description>Comment: 13 pages + 4 eps figures,
                PTPTeX, typographical errors
                corrected</dc:description>
              <dc:date>1998-12-31</dc:date>
              <dc:date>1999-05-10</dc:date>
              <dc:type>text</dc:type>
              <dc:identifier>http://arXiv.org/abs/hep-
                th/9901001</dc:identifier>
              <dc:identifier>Prog.Theor.Phys. 101 (1999) 1155-
                1164</dc:identifier>
          </oai_dc:dc>
        </metadata>
      </record>
    </GetRecord>
  </OAI-PMH>
```

Figure 7. An example of an OAI meta-data record returned using the Dublin Core meta-data format.

The Identify command is used to retrieve information about a repository (e.g., repository name, base URL). The ListRecords command is used to harvest records from a repository. *Selective harvesting* allows record selection based on when they are created, deleted, or modified. Meta-data records can also be grouped into sets for selective harvesting. The ListSets function retrieves the structure or hierarchy of a set. The ListIdentifiers function retrieves record headers instead of a full record. A server can then issue requests based on whether a record is present or has been deleted. As described above, OAI–PMH supports "parallel" meta-data sets. The ListMetadataFormats command is used to return the available meta-data formats from a repository.

4.4   Information and Content Exchange (ICE)

Information and Content Exchange (ICE) is an application of the eXtensible Markup Language (XML). It is a protocol for content syndication and subscription and was submitted to the W3C as a NOTE in 1998 [WEBB98]. Version 1.0 of the ICE specification was released in 1998. This section describes the protocol and implementation for ICE Version1.1. The ICE Working Group is currently working on v1.2 of the specification that will use Web Services. Table 5 "ICE at a Glance" summarizes ongoing efforts.

The ICE protocol includes functionality that could be applied to the implementation of a repository where content and meta-data is requested "manually or automatically and scheduled." The ICE specification also includes a set of Use Cases (*ICE Implementation Cookbook* [SOUZ00]) that outline syndication and subscription guidelines.

ICE at a Glance		
Institution	**Version**	**Reference Models**
IDEAlliance Alexandria, VA	ICE v1.1 (released 1998) ICE v1.1 (released 2000) ICE v1.2 (started 3/2002) ICE supports: - HTTP POST 'request/response' transport model. "Push" or "pull" of content as well as delivery times and frequency. - Incremental or full content updates. - Data management (error logging, delivery notification).	Open source, Java-based implementation: *TwICE* *http://twice.sourceforge.net* Industry Implementations: *Kinecta Corporation* *National Semiconductor* *Oracle 9i* *Vignette Syndication Server* *Reuters* *InterWoven*

Table 5. Information and Content Exchange (ICE) at a glance.

Similar to the *data providers* and *service providers* used by the OAI model described in the previous section, ICE uses the terms *syndicator* and *subscriber* to describe the relationship between a content provider that distributes large volumes of content to consumers. The following concepts are basic to understanding the ICE model:

- *Syndicator*: a content aggregator and content distributor.

- *Subscriber*: a content consumer.

- *ICE package*: a content payload that is independent of the protocol.

- *ICE payload*: an XML document used to carry protocol information.

ICE DTD and Protocol

The basic concepts in the ICE model are represented using the element/attribute markup model of XML. However, ICE is a *request/reply* protocol, not just a DTD, with a single `ice-payload` root element and a structured hierarchy of tags that describe operations and data as shown in Figure 8. ICE uses the *payload* as its fundamental protocol model, where a *payload* is defined as a single instance of an XML document formatted according to the ICE protocol definition [BROD02]. ICE operates within existing network infrastructures and transmits payloads using the HTTP POST/Response mechanism. ICE/HTTP does not define any new HTTP headers or modify the HTTP protocol. An `ice-request` and `ice-response` occurs entirely within a single HTTP POST/Response transport level transaction.
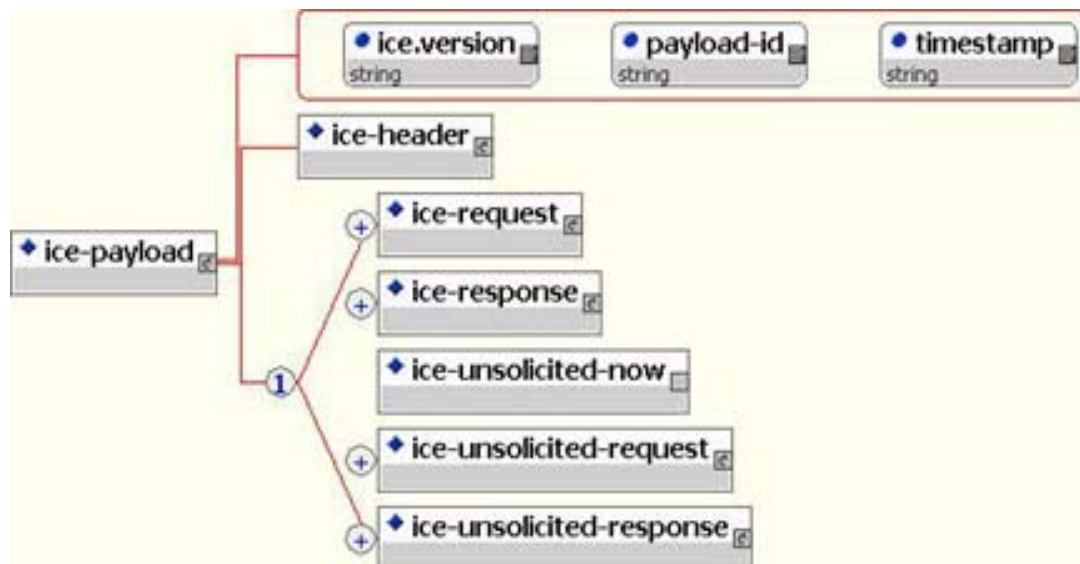

Figure 8.   **Excerpt from the ICE DTD.**

The protocol allows for fully symmetric implementations, where both the *syndicator* and *subscriber* can initiate requests. It also supports a "minimal subscriber" implementation (i.e., only the *subscriber* can initiate requests). *Subscriber* and *syndicator* can also agree on a delivery schedule and methods. ICE also defines a mechanism by which event logs can be automatically exchanged between parties to diagnose problems. ICE requires that both parties use a globally unique identifier. The identifier must conform to the Universal Unique Identifier (UUID) using a 32 hexadecimal digit format.

Delivery of Content

A *package* is the atomic unit of information distribution in ICE and is defined separately from the protocol. ICE defines a *sequenced package model* that allows *syndicators* to support both "incremental" and "full update" models. ICE defines operations for content delivery in either push or pull mode. In *push* mode ICE payloads containing packages are delivered whenever there is new content, while in *pull* mode a subscriber initiates content delivery (polls for updates). Content can be inline or delivered by reference (`ice-item-ref`). Several other features are provided for content delivery including the ability to specify:

- The length of time that content will be available (`ice-access-window`)

- Basic authentication (`ice-access-control`)

- Confirmed delivery processing (using the `confirmation` attribute)

- Delivery rules (`ice-delivery-rule`, `ice-delivery-policy`)

```xml
<?xml version="1.0" ?>
<!DOCTYPE ice-package SYSTEM
"http://www.icestandard.org/dtds/ICE1_1.dtd">
<ice-package >
<ice-item-ref
url="http://www.bradsgadgets.com/news/techtips/<todaysdate>/news.html"
item-id="BG1" >
<ice-access >
<ice-access-window starttime="2000-07-21T08:00:00"
stoptime="2001-08-01T00:00:00" />
</ice-access >
</ice-item-ref >
</ice-package >
```

Figure 9.   Example of specifying content availability in ICE. Reproduced from Souzis, Popkin, Khoury and Hunt, IDEAlliance, 2002.

4.5 **OASIS ebXML**

OASIS ebXML (electronic business XML; hereafter ebXML) is a collection of standards, or standard consisting of multiple interoperating pieces, developed by the OASIS global consortium in conjunction with the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) for automating business processes and interactions electronically using XML. OASIS ebXML currently consists of seven specifications: Technical Architecture Specification [EBTA01], Business Process Specification Schema [EBBP01], Requirements Specification [EBRQ01], Registry Information Model [EBRM01], Registry Services Specification [EBRS01], Message Service Specification [EBMS02], and Collaboration-Protocol Profile and Agreement Specification [EBCP02]. For an in-depth history, see the ebXML Technology Report on the OASIS Cover Pages [CPEB02]. Table 6 provides a quick executive overview of this standards effort including current versions of the each specification.

"The goal is to provide an XML-based open technical framework to enable XML to be utilized in a consistent and uniform manner for the exchange of electronic business data in application to application, application to human, and human to application environments--thus creating a single global electronic market." [EBRQ01]

Other ebXML goals include:

- Ensuring compliance with W3C XML technical specifications.
- Enabling interoperability between ebXML compliant trading partner applications.
- Providing interoperable and efficient transition paths to ebXML from EDI and XML business standards.

OASIS ebXML at a Glance		
Institution	**Version**	**Reference Models**
OASIS http://www.oasis-open.org UN/CEFACT http://www.unece.org/cefact/ <u>Technical Committees</u> • Collaboration Protocol Profile and Agreement (CPPA) • Implementation, Interoperability, and Conformance • Messaging Services • Registry	<u>Specifications</u> Registry Information Model (RIM) v2.1 (June 2002) Registry Services (RS) Specification v2.1 (June 2002) Message Service Specification v2.0 rev C (February 2002) Collaboration-Protocol Profile and Agreement Specification v2.0 (September 2002) Technical Architecture Specification v1.04 (February 2001) Requirements Specification v 1.06 (May 2001) Business Process Specification Schema v1.01 (May 2001)	Open Source Implementations: http://ebxmlrr.sourceforge.net Registry Information Model http://www.ebxmlsoft.com/solutions/ebxmlsol.html <u>List of implementations</u>: http://www.ebxml.org/implementations/index.htm

Table 6.   OASIS ebXML at a glance

The OASIS ebXML system and standards are the most extensive of those reviewed. ebXML seeks to provide rules and functionality for enabling the automation of most types of business transactions. As a result, the scope of the ebXML standards is much broader and functions at a higher level of abstraction than those directed at one specific problem. The following concepts are key to understanding ebXML:

- *Registry*: Central server that stores the following information available in XML format: Business Process and Information Meta Models, Core Library, Collaboration Protocol Profiles, and Business Library. Organizations use the information stored in the registry to identify potential partners and learn how to interact with them.

- *Business Processes*: Activities an organization participates in, for which it may want partners. These are modeled in XML or UML using the Business Process Specification Schema (a W3C XML Schema or DTD; see Table 7).

Technologies for the Design of Learning Object Repositories

- *Collaboration Protocol Profile (CPP)*: registry stored profile that contains descriptions of Business Processes and Business Service Interfaces for a company seeking to engage in ebXML transactions.

- *Business Service Interface*: describes the way an organization interacts with partners, including the types of Business Messages and protocols used.

- *Business Messages*: information communicated as part of a business transaction.

- *Core Library*: standard functional pieces provided by ebXML that may be incorporated as part of larger ebXML elements.

- *Collaboration Protocol Agreement (CPA)*: a contract between two or more organizations that can be automatically derived and acted upon from information provided in a CPP.

- *SOAP*: used by ebXML as a messaging framework (see section 4.1)

Business Process

The ebXML Business Process paradigm addresses the method by which a company finds another trading partner and commences business in an automated fashion. Specifically the Business Process outlines the method by which companies can create and list a CPP in a registry; search other published organizations' profiles to find potential trade partners; and obtain all of the information necessary to start transacting business with a listed organization via a CPP and CPA.

```
<!ELEMENT ProcessSpecification
        (Documentation*,
        (Include* | DocumentSpecification* |
         ProcessSpecification* | Package |
         BinaryCollaboration | BusinessTransaction |
         MultiPartyCollaboration)*)>
<!ATTLIST ProcessSpecification
        name    ID   #REQUIRED
        version CDATA #REQUIRED
        uuid    CDATA #REQUIRED >
```

Table 7   **DTD declaration for ebXML Business Process Specification document**
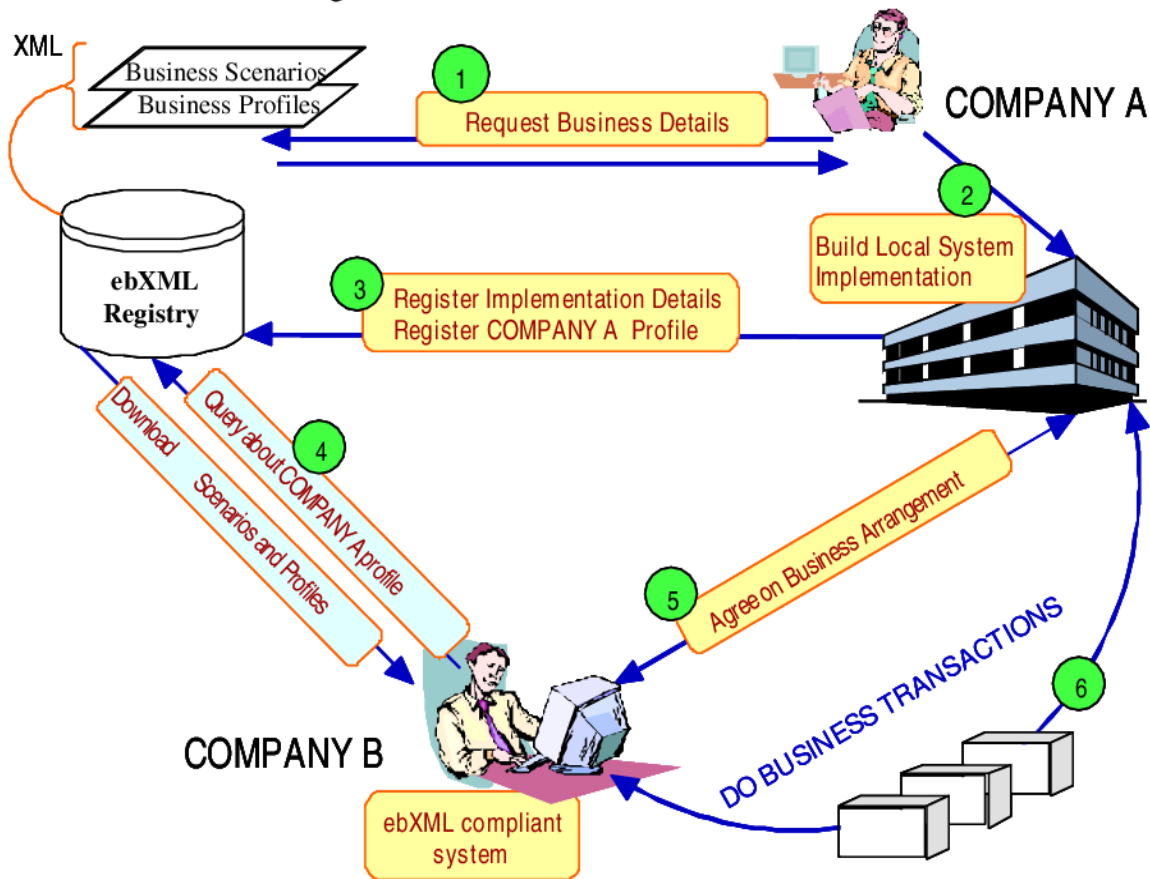
Figure 10. **ebXML Business Process. Reproduced from OASIS ebXML Technical Architecture 2001**

ebXML Business Process displayed in Figure 10 outlines six steps through which two companies develop a business relationship.

1. Company A searches a Registry for business opportunities and business profiles.
2. Company A identifies Company B as a potential partner and implements a system for interaction based on the information retrieved.
3. Company A lists their own profile in the registry.
4. Company B retrieves company A's newly registered profile.
5. Based on the information contained in their respective profiles, Companies A and B arrange to do business. Assuming that the CPA provided by Company A conforms to requirements listed in Company B's CPP this may be conducted automatically.
6. Business commences.

Technologies for the Design of Learning Object Repositories
                                                                                021751

Messaging

The ebXML Messaging Service specification is an open, non-proprietary standard designed to enable the *secure, reliable* exchange of any type of business information regardless of encoding [FERR01]. It is designed to be a simple, scalable solution for enabling the transfer of XML encoded content of any type (e.g. XML, EDI transactions, binary data) over a variety of communication protocols.

The ebXML Message Service follows several important design objectives:

1. *ebXML Message Service Specification incorporates existing standards wherever possible.* Though initially developed separately from the W3C SOAP messaging effort, ebXML Messaging Service later incorporated SOAP as a core part of the specification. The current specification defines a layered set of extensions on top of SOAP 1.1 and SOAP with Attachments. These extensions address areas such as security and reliability not directly addressed in the SOAP specifications (see Section 4.1).

2. *ebXML Message Service utilizes a message structure and protocol that is independent of the underlying transport.* Having the message structure separate from the underlying transport protocol allows for use of ebXML messages in a wider variety of circumstances. Specifically, ebXML messages can be sent over any protocol capable of handling MIME data (e.g. SMTP, FTP, HTTP, etc.).

3. *ebXML Message Service is payload neutral.* One of the major differences between the ebXML and SOAP messaging lies in the ebXML messaging service's ability to transmit content of any type. This includes everything from XML structured textual data, to binary data. In an ebXML message the MIME packaging scheme separates the header information from the message content as shown in Figure 11. This separation allows the message headers and body to be processed separately. Header information may contain instructions to be used in routing and in route message processing without requiring the entire message to be readable (for more on this, see "SOAP Message Path" under Section 4.1). In addition, allowing the transmission of any type of data in the message body not only accommodates a broader set of messages, but also facilitates improved security through the transmission of encrypted data.
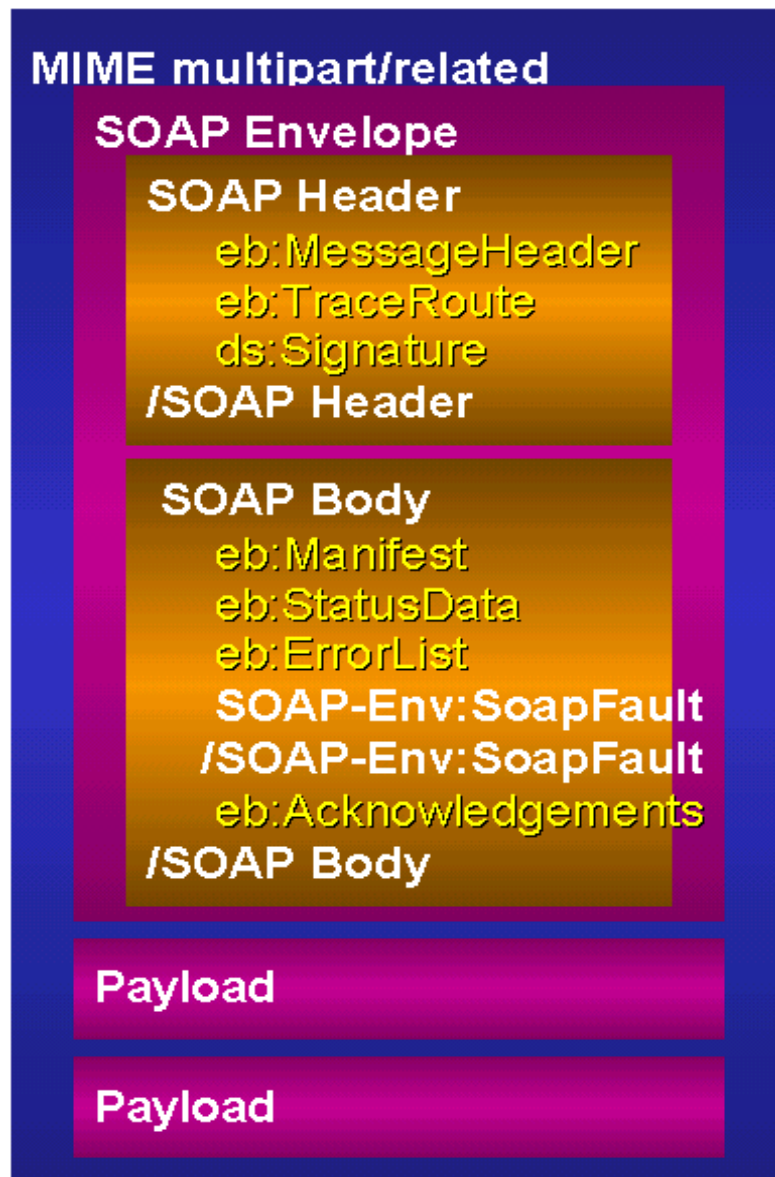
Figure 11.   ebXML Message Framework. Reproduced from Ferris, 2001

4. *ebXML Messaging Service provides for reliable messaging.* Since ebXML
messaging may be used for transactional exchanges, it must allow for faults in the
system (e.g., network, software, etc.). The messaging service accomplishes
reliability through a simple set of rules: In an ebXML message exchange, only
one copy of the message will be delivered to the receiver; the receiver will notify
the sender once the message has been successfully received; if no
acknowledgement is received, the messaging service will retry until it receives an
acknowledgement or notify the sending application as shown in Figure 12.

Technologies for the Design of Learning Object Repositories
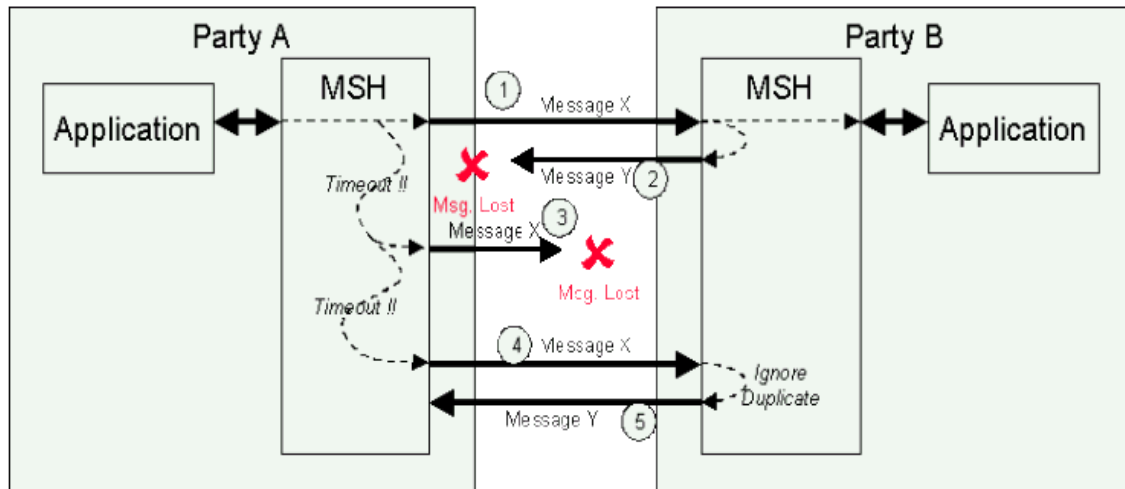   021751

Figure 12. **ebXML Reliable Messaging example. Reproduced from Ferris, 2001**

5. *ebXML Messaging Service accommodates a variety of security models.* Security is a major concern for any Internet messaging service. The ebXML Messaging Service is no exception, however, its design accommodates a variety of security measures, specifically: an ebXML message may be digitally signed for authentication and integrity verification using the methodology contained in the W3C/IETF XML Signature Standard; the payload of an ebXML message can be encrypted using S/MIME or PGP/MIME so that only the recipient can decrypt and access it; the transport agnostic nature of ebXML allows messages to be sent with secure protocols such as SSL or IPSEC.

For these reasons. the ebXML Message Service provides one of the most complete XML-based messaging standards currently available. Though developed in conjunction with the other portions of the ebXML standard, the Messaging Service does not require functionality defined in the other specifications and may be used independently.

Registry and Collaboration Protocol Profiles and Agreements

The ebXML Registry provides services to enable electronic sharing of information for the purpose of facilitating the integration of business processes based on ebXML specifications. Specifically, the ebXML registry stores business process schemas and documents, Collaboration Protocol Profiles for sellers, classification schemas to facilitate discovery and Collaboration Protocol Agreements for buyers. ebXML Messaging is used for interaction with an ebXML registry.

The Registry architecture consists of a Registry Service component, and a Registry Client component. These Registry Service interface shown in Figure 13 provides the primary method for communicating with the ebXML registry. ebXML clients, whether a person or another application, communicate with the Registry via the same interface. In some cases, such as when a person uses a thin client like a Web browser to access the registry, an extra layer (i.e., a Web server) will need to be deployed on the server side as an intermediary between the thin client and the Registry interface.
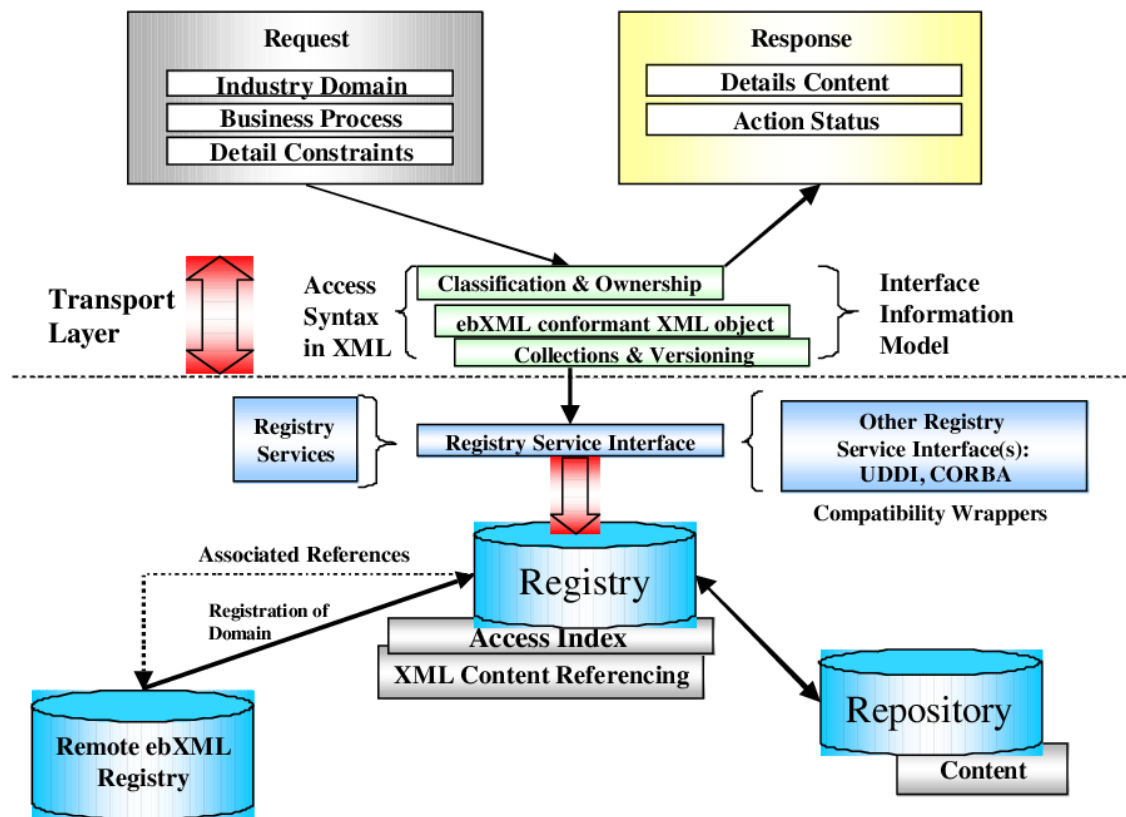
Figure 13.   **Registry Architecture. Reproduced from OASIS ebXML Technical Architecture 2001.**

The ebXML Registry includes Life Cycle and Query Management interfaces. The Life Cycle interface accommodates the management of Registry entries, such as Create, Update, and Delete functionality, whereas the Query Management interface supports the query and retrieval of stored ebXML documents. Clients may make calls to either interface assuming they have permissions.

Automating Business-to-Business transactions is a key focus of ebXML. As mentioned previously the CPP and CPA are at the heart of the ebXML registry system and provide the key for automated business negotiations. A CPP provides information about the businesses an organization supports, the service interfaces they provide to support their business processes, the definitions for the business messages their service interfaces

exchange, and information about the technical configuration they support (e.g., transport, security, and encoding information). Published to an ebXML registry, a potential partner, discovering this information for the first time via a registry search can retrieve everything they need to know in order to write a CPA. A CPA may be derived directly from the other organizations' CPP information. Assuming the CPA is created properly, the agreement may be finalized in an automated fashion, and the new partners may begin transacting.

OASIS/ebXML efforts are ongoing. As stated earlier, ebXML aims to bring together other important e-commerce related standards as they emerge, in order to provide interoperable and efficient transition paths to ebXML. For example, the current Registry Technical Committee is working to create liaisons with other standards efforts, including OASIS UDDI, ISO 11179, and W3C XML Query Working Group (XQuery). [EBRT02] In another effort, collaboration with the RosettaNet effort led to adoption of ebXML Messaging Services Specification for RosettaNet supply chain management [CPRN02].

4.6   UDDI/Web Services

Universal Discovery, Description and Integration (UDDI) is another OASIS business registry standards effort. UDDI provides service registry functionality. Acting as an advanced services directory, UDDI allows organizations to search for and discover other registered organizations, and retrieve information about how to interact with each other over the internet.

Functionally UDDI, when coupled with the Web Services Definition Language (WSDL) and SOAP, provides a functional system very similar to ebXML. Considering these similarities, one may reasonably ask why OASIS is leading two major competing standards efforts. A bit of history explains the circumstance. ebXML and UDDI were developed concurrently. ebXML by OASIS and UN/CEFACT, and UDDI by a community of more than 300 companies. In July of 2002, having achieved their original charter, the UDDI community and OASIS announced that OASIS would oversee further development of the UDDI standard. Version 3 of the UDDI standard followed shortly thereafter. Though there is significant functional overlap between ebXML and UDDI, the two represent somewhat different approaches with differing relative strengths. Efforts are underway within OASIS and the vendor community to harmonize the two efforts as much as possible. The Cover Pages, hosted by OASIS, are an excellent resource that provide an extensive and ongoing log of ebXML [CPEB02], UDDI [CPUD02], and other related efforts.

This section covers UDDI and Web-Services. UDDI uses SOAP as a messaging service, which is discussed in section 4.1. Table 8 provides quick reference information about UDDI.

UDDI at a Glance		
Institution	**Version**	**Reference Models**
OASIS http://www.oasis-open.orgUDDI Website: http://www.uddi.org	<u>Standards</u> UDDI v3.0 (July 2002) <u>UDDI supports:</u> ▪ White, yellow and green pages categorization of Web Services. ▪ Unique identification of registry entries ▪ Digital signatures ▪ Public, Private and Shared registries ▪ Registry interaction	List of implementations: http://www.uddi.org/solutions.html

Table 8.   UDDI at a glance

A UDDI registry stores data published by entities to the registry and makes it available to other organizations when searched. This functionality for storing, searching and classifying service data within a UDDI registry is often explained using the phonebook metaphor of White, Yellow, and Green Pages. UDDI "White Pages" identify organization specific information such as business name, description, contact information, and other known identifiers (e.g. abbreviations/aliases). UDDI "Yellow Pages" identify businesses categorically by standard classifications such as Industry, Product/Services, and Location. UDDI "Green Pages" document technical information about services that are exposed.

As shown in Figure 14, UDDI supports a variety of network models. As web services do not exist exclusively in public, the UDDI standard focuses on supporting a variety of network model implementations including public–Internet, private–intranet, and shared/semi-private–extranet environ,ments in which a registry may be deployed. Initially, UDDI registries were designed for deployment as peers in a network. The Version 3.0 release of the UDDI standard changed this to a hierarchical model analogous to the model currently used for DNS. Microsoft, IBM, NTT Com, and SAP currently operate the Universal Business Registry (UBR), the root UDDI registry in the public domain.

Technologies for the Design of Learning Object Repositories
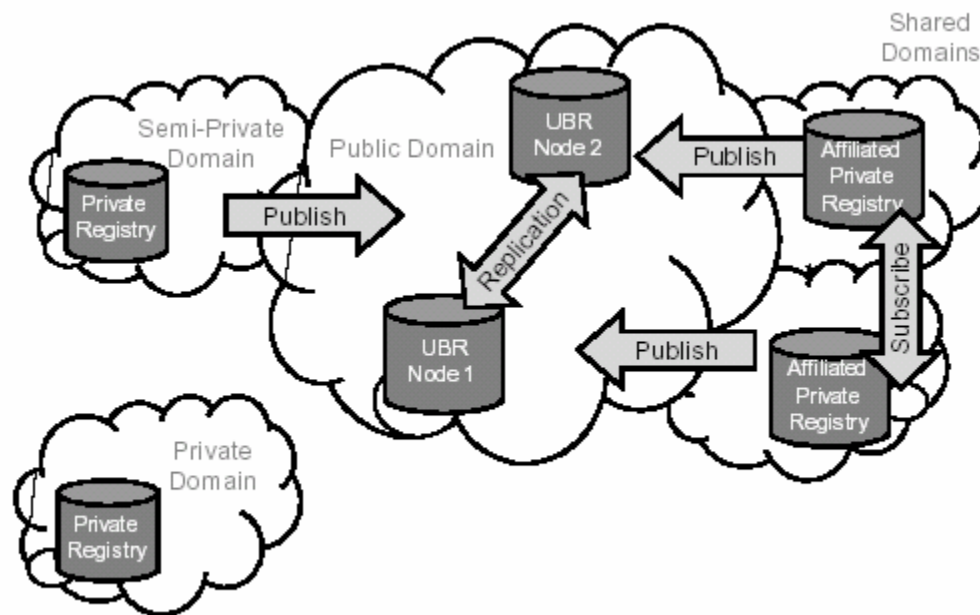
Figure 14.   **UDDI Registry Network Model. Reproduced from The Stencil Group, July 2002**

The UDDI registry defines two APIs (similar to the two ebXML registry interfaces): a Publication API, and a Subscription API. Users and systems utilize the Publication API to publish information to a UDDI registry. All UDDI registries require a unique ID for registering objects. A new unique ID is generated and assigned to each new object submitted to the directory. The Subscription API allows the user monitor changes to services listed in the registry.

Web Services

The Web Services Definition Language (WSDL) standard is still maturing and is currently a W3C Working Draft in development. Functionally, Web Services play a similar role to the ebXML CPP and CPA specifications, acting as a glue to bring applications together. The Web Services standard allows organizations to describe services they want to make available, publish that information, discover other services and connect to them.

Dedication to open interoperable standards is another commonality WSDL shares with ebXML. To ensure this, WSDL is based on XML, which allows it to accommodate various operating systems, programming languages, and applications. The approach taken for WSDL accommodates the loose coupling necessary for interoperating Web components.

Technologies for the Design of Learning Object Repositories

4-31

021751

A typical Web Service provides a self-descriptive XML-based protocol stack of sorts. This information is necessary to enable programmers and systems to derive rules and create applications capable of interacting with others published on a UDDI registry. This protocol stack progressively builds across six major elements:

- *Types*: containers for data type definitions using some type system (such as XSD).
- *Message*: an abstract, typed definition of the data being communicated.
- *Operation*: an abstract description of an action supported by the service.
- *Port Type*: an abstract set of operations supported by one or more endpoints.
- *Binding*: a concrete protocol and data format specification for a particular port type.
- *Port*: a single endpoint defined as a combination of a binding and a network address.
- *Service*: a collection of related endpoints.

A WSDL *type definition* is implemented as an XML Schema and provides a name and XML Schema type as shown below in Table 9. Since the implementation of similar types varies system to system, using XSchema defined types provides a critical foundation for interoperability.

```
<!—type defs -->
<types>
<xsd:schema xmlns:xsd=http://www.w3.org/1999/XMLSchema/>
<xsd:element name="phone" type="xsd:string"/>
<xsd:element name="fullname" type="xsd:string/>
</xsd:schema>
</types>
```

Table 9.   WSDL Type Definition

The *Message Declaration* comes next. In WSDL a message declaration defines message names and parameters. The parameters are derived from the previously defined types as shown in Table 10.

```
<!—message declns -->
  <message name="AddContactInfo"
    <part name="fullname" type="xsd:string"/>
    <part name="phone"  type="xsd:string"/>
  </message>
```

Table 10.   WSDL Message Declaration

Technologies for the Design of Learning Object Repositories
                                                                                                021751

The *Operation* and *Port Type* declarations are combined here. Notice how the operation includes the already defined message, see Table 11.

```
<!—port type declns -->
  <portType name="ContactDB">
     <operation name="addContact">
       <input message="AddContactInfo"/>
     </operation>
  </portType>
```

Table 11.   WSDL Operation and Port Type Declaration

The *Binding* declaration binds the Port Type and Operation just declared to a specific transport. In this case SOAP RPC, see Table 12.

```
<!—binding declns -->
 <binding name="ContactDBSOAPBinding" type="ContactDB">
  <soap:binding style="rpc"
     transport=http://schemas.xmlsoap.org/soap/http/>
   <operation name="addContact">
     <soap:operation soapAction=""/>
     <input>
       <soap:body use="encoded"
          namespace="urn:ListUpdater"
          encodingStyle=http://schemas.xmlsoap.org/soap/encoding/>
     </input>
     <output>
       <soap:body use="encoded"
          namespace="urn:ListUpdater"
          encodingStyle=http://schemas.xmlsoap.org/soap/encoding/>
     </output>
   </operation>
 </binding>
```

Table 12.   WSDL Binding Declaration

Finally, the *Service Declaration* defines a port tied to the binding and provides an address to the specific SOAP service as shown in Table 13.

```
<!—service decln -->
 <service name="ContactDBService">
   <port name="ContactDB" binding="ContactDBSOAPBinding">
    <soap:address
    location="http://localhost:2020/soap/servlet/rpcrouter
   </port>
 <service>
```

Table 13.   WSDL Service Declaration

This page intentionally left blank.

Section 5: Supporting Technologies

5.1. Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is a lightweight XML-based protocol for exchanging structured electronic information between peers in a decentralized distributed environment. Much as HTML facilitated the growth of the Internet by providing a framework for information layout of information downloaded via HTTP, SOAP aims to enable communication among Web services by providing a common method for structuring and transmitting data. The multi-part SOAP specification [SOMF02] [SOAD02] is currently a working draft last call, two levels below an official recommendation, at the World Wide Web Consortium (W3C).

Table 14, "SOAP at a glance," provides information about current SOAP efforts.

| SOAP at a Glance		
Institution	**Version**	**Reference Models**
World Wide Web Consortium http://www.w3c.org/	Version 1.2 Working Draft (Released 6/2001) SOAP supports: ▪ A Simple XML message structure. ▪ Peer-to-Peer distributed transactions. ▪ HTTP and RPC bindings. ▪ Fault handling.	Open SOAP implementation: Apache SOAP http://xml.apache.org/soap/ Programming Languages with SOAP Support: Java, C/C++, Python, Perl, Visual Basic, PHP, ColdFusion, ASP/.NET and others. Industry Implementations: – IBM – Microsoft – SUN – WebMethods/Vignette – Oracle – SAP – others

Table 14.   Simple Object Access Protocol at a glance.

The fundamental SOAP paradigm is a stateless one-way messaging between peers in a decentralized distributed environment. In a typical interaction, a SOAP sender creates a message and sends it to a SOAP receiver. Despite this simplistic sender/receiver model, SOAP includes functionality via the SOAP header to allow for much more sophisticated interactions such as request/response, "conversational" exchanges, and multi-node message paths.

SOAP does not specify semantics for the application-specific data it conveys, but rather provides a common framework for enabling application-to-application data exchange. In this respect, SOAP is similar to TCP/IP and other communication protocols, which provide a standardized way of transmitting data without dictating content specific formatting. In short, SOAP provides syntax for including content in a message, but not for the content per se.

SOAP Message Path

As mentioned previously, SOAP messages are fundamentally one-way transmissions between a SOAP sender and receiver as depicted in Figure 15. To accommodate more complex messaging scenarios, the SOAP message structure allows for the inclusion of optional SOAP headers. When included, headers may contain information instructing specific nodes about how to process the SOAP message and where to send it next. It is important to note, that even in message paths consisting of many nodes the interaction between any two successive nodes still utilizes the simple SOAP send and receive model discussed previously. The chief difference in such cases is that intermediate nodes, or SOAP intermediaries, function as both receivers and senders: first, receiving the message, and then sending it along to the next node, after reading the pertinent headers and processing the message as directed.
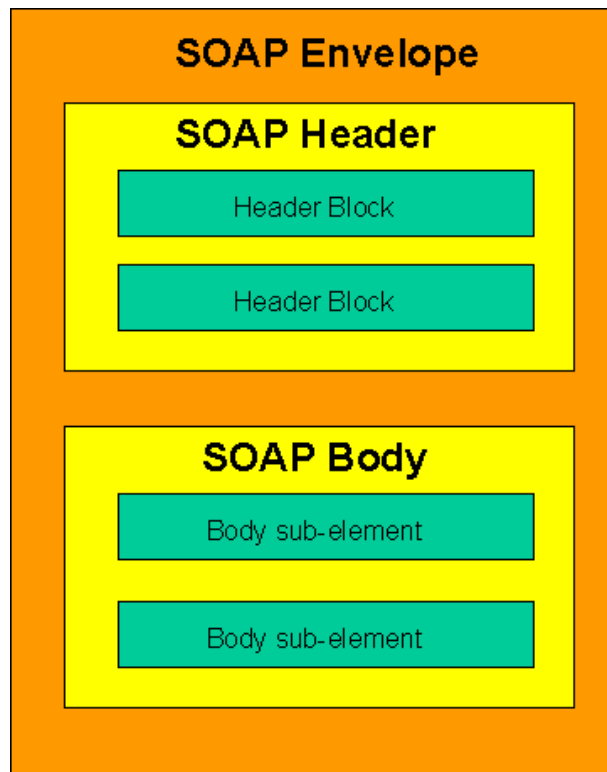
Figure 15. **SOAP Message Structure.**

SOAP Message Structure

A SOAP message consists of three parts SOAP Envelope, Header, and Body, see Figure 15. The SOAP envelope provides a wrapper for the message internals, namely the header and the body; the optional SOAP header or headers are used for including message processing information for SOAP processing nodes along a message path; the SOAP body consists of two parts: a payload made up of zero or more element information items; and a SOAP fault, an element information item which consists of information generated by a SOAP node.

The optional SOAP Header is included as part of a SOAP message to provide flexibility in anticipation of application specific needs such as a workflow and/or routing in scenarios including more than one processing node for a single SOAP message. A message may contain no headers, a single header or multiple headers.

The SOAP Body Consists of two parts: SOAP Body parts (payload or content); and the SOAP Fault.

XML provides the necessary structure for the SOAP message. Figure 15 depicts an XMLL example of a SOAP message for requesting electronic content from a fictitious SOAP enabled ADL source.

```
<soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/">
    <soap-env:Header/>

    <soap-env:Body>
        <adl:SoapExample
         xmlns:adl="http://www.example.com/adlsoapexample">
            <adl:NewContentRequest>
                <adl:ContentTitle>New ADL Sample Content
                </adl:ContentTitle>
                <adl:RequestDate>10/31/2002
                </adl:RequestDate>
            </adl:NewContentRequest>
        </adl:SoapExample>
    </soap-env:Body>

</soap-env:Envelope>
```

Table 15   **Example of a SOAP message requesting content from an enabled ADL source.**

Looking at the message we see the same SOAP structure as illustrated in Figure 15. The SOAP envelope, notated soap-env:Envelope, is the outermost element. Notice that the entire message is contained within the envelope. The SOAP header (`<soap-env:Header>`) comes next. In this case the header is empty. The SOAP Body comes third. Unlike the empty header in this example, the SOAP body contains the message.

The SOAP fault mechanism allows for error reporting. If a SOAP receiver or intermediary cannot process a SOAP message, the node that encountered the error generates error information, appends it to the SOAP message and returns it to the sender.

5.2   *Repository Data Storage/Searching using XML Databases and XQuery*

Native XML databases have been defined as ones that define a logical model for an XML document, not the data in the document, and store and retrieve documents according to that model [BOUR02]. They use an XML document as the basic unit of storage just as a relational database uses a row in a table but are not required to use any particular underlying storage model. Table 16 shows a comparison of the XML Model with the relational model [HARO01].

XML database	Relational database
Based on collections of XML documents with same DTD	Based on tables containing records with same schema
XML document is a tree of nodes.	Record is an unordered list of named values.
XQuery returns an unordered sequence of nodes.	SQL query returns an unordered set of records.

Table 16.   XML Database Model vs. Relational Database Model. Adapted from Harold, 2001.

Native XML databases fall into two major categories:

- *Text-based* – where the entire document is stored in text form and some sort of database functionality is provided to access the document. This approach can return an exact "image" of the original document (e.g., replicating the amount of white space or the types of quote marks used).

- *Model-based* – where a binary model of the document (e.g., Document Object Model or DOM) is stored in a custom database. While this approach can only return a model of the document, it is generally faster when combining text fragments from multiple documents.

Native XML databases differ from *XML-enabled* databases in three major areas:

- Native XML databases preserve the physical structure of the original document as well as any comments, DTDs, etc.

- Native XML databases can store documents without knowing the XML schema or DTD.

- XML and related technologies such as XPath, the DOM and XML database APIs provide the only access to the data in a native XML database. ODBC and other mechanisms that provide direct access to the data are not supported.

There are a number of direct XML-based search mechanisms. These include XML Query/XQuery [BOAG02], XPath v1.0 [CLAR99] and its successor, XPath v2.0 [BERG02]. These methods are based on a model that deals with the hierarchy in an XML document directly. In many implementations, XPath is currently the native XML query language of choice. Databases supporting the XPath query mechanism include 4Suite, EXcelon's eXtensible Information Server (XIS), GoXML DB, Ipedo XML Database 3.0, NeoCore XMS (XML Management System) 2.0, Oracle 8i and 9i, Tamino XML Server 3.1, and Xindice.

In order to function as a database query language, XPath has been extended to allow queries across collections of documents. However, the first release of XPath did not allow joins or sorting of query results [DYCK02]. The second version of XPath (v2.0) is

derived from XPath v1.0 and XQuery 1.0 and the new XPath draft shares much of its text with the working draft of XQuery. Major database vendors including IBM, Oracle and Microsoft are implementing XQuery support for their products [SCAN02].

There are a number of sources for information on XML databases. One source is the XML:DB project mailing list located at http://www.xmldb.org/projects.html. The site has a listserv for ongoing discussions on XML databases. Table 17 lists some of the commercial and Open Source XML database products that support XPath and XQuery [BOUR02].

Product	Type	XPath	XQuery
4Suite http://4suite.org/index.xhtml	OpenSource	✓	
EXcelon's eXtensible Information Server (XIS) 3.1 http://www.exln.com/products/xis/	Commercial	✓	✓
eXist 0.8.1 http://exist.sourceforge.net	OpenSource	✓	✓
GoXML DB http://www.xmlglobal.com/prod/db/index.jsp	Commercial	✓	✓
Ipedo XML Database 3.0 http://www.ipedo.com/html/products_xml_dat.html	Commercial	✓	✓
NeoCore XMS (XML Management System) 2.0 http://www.neocore.com/products/products.htm	Commercial	✓	✓
Oracle 8i and 9i http://www.oracle.com	Commercial	✓	✓
Tamino XML Server 3.1 http://www.softwareag.com/tamino/architecture.htm	Commercial	✓	✓
Xindice http://xml.apache.org/xindice/	OpenSource	✓	

Table 17.   Comparison of Representative Native XML Databases

Section 6: Gap and Risk Assessment

In reviewing the technologies and standards for learning object repositories, there are risks associated with the various standards and implementation approaches that can be applied. These include the *time-to-adoption*, *industry acceptance*, *competing standards* and the *unique identification of learning objects*. Each of these areas can have an impact on the design, development and deployment of learning object repository systems and their availability in the marketplace.

6.1 Projected Risks

6.1.1. Time-to-Adoption

A major risk to the timely adoption of repository systems is the lack of mature standards. Examples include:

- The Open Archive Initiative (OAI) is an experimental model.
- The SOAP messaging protocol is a W3C Working Draft Final Call.
- UDDI requires WSDL, a W3C Working Draft in development.
- ebXML is a standard that references SOAP and SOAP with attachments.
- IMS also depends on SOAP and SOAP with attachments.
- All of these standards reference XML schema that is still in draft status.

6.1.2 Industry Acceptance

Vendors are supporting multiple standards, but there is uncertainty in the market as to which standards will gain acceptance. Many vendors have their own strategy to ensure that they are not left out. Selecting a viable long-term implementation is still problematic. For example:

- IBM supports – UDDI, ebXML, WSDL, SOAP and Web Services
- Microsoft has retracted BizTalk as a standard, but still provides it as a product
- UDDI has been accepted, but there are a few UDDI public registries
- DOI has been accepted and is being used by ACM and the Association of American Publishers among others. However, it has not gained acceptance as a widespread method of identifying objects.
- Businesses do not want to pay for significant infrastructure unless they know it will work and be adopted

- Several universities are implementing systems, however there is still not a wide spread developer community.
- OpenSource software can be a driver for new technology. It can speed adoption by some businesses, but others may wait for vendor-supported implementations.

6.1.3 Standards

Competing standards can delay acceptance and some standards are in the process of resolving intellectual property issues. Examples include:

- Multiple standards addressing the same problems and competing with each other can delay user acceptance.
- Patents and other Intellectual Property (IP) issues can delay implementations while licensing and royalty issues are addressed. For example, WebMethods may own patents related to SOAP, while the W3C only incorporates royalty-free technologies into its standards. There may be similar issues with IBM and WSDL.

6.1.4 Unique Identification

A standard method of uniquely identifying learning objects and other assets is required to implement interoperable repository systems. There are several methods for providing unique identifiers (e.g., GUID, UUID, DOI, URN, URI). A number of system implementation issues exist that must be resolved to create an interoperable SCORM repository. These include:

- Packages are not uniquely identified at the package, Sharable Content Object (SCO), or asset level.
- Any technology or standard that is selected must provide unique identification for a variety of asset types.
- Different standards use different identifiers, which impedes interoperability. For example, IMS uses DOI while OAI uses URI while ICE and ebXML use UUID, and so on.

Section 7: Appendix A – Glossary

AAP	American Association of Publishers
ANSI	American National Standards Institute
CAREO	Campus Alberta Repository of Educational Objects
Data Element	A unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of *attributes*.
Data Element Concept	A concept that can be represented in the form of a data element, described independently of any particular representation.
DLORN	Distributed Learning Object Repository
DOI	Digital Object Identifier
DRM	Digital right management
DTD	Document Type Definition. Original XML 1.0 solution for defining the structure and constraints of an XML document. (See also Schema).
federation	In a federation, a group of organizations agree that their services will be built to certain specifications. Organizations that build systems to these specifications form a federation. The main challenge in forming a federation is the effort required by each organization to implement and keep current with all the agreements.
harvesting	The underlying concept for harvesting is that participants make some efforts to enable some basic shared services, without specifying a complete set of agreements.
IDF	International DOI Foundation
IEEE	Institute for Electrical and Electronic Engineers
IETF	Internet Engineer Task Force
IMS	Instructional Management System
institutional repository	Digital collections that capture and preserve the intellectual output of university communities
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
gathering	Examples of gathering are the web search engines. Gathering can provide services that embrace large numbers of repositories. However, the services are of poorer quality than can be achieved by partners who cooperate more fully (e.g., federation).
MARC	MAchine Readable Cataloging. A format commonly used in library systems for bibliographic meta-data.
meta-data	Data that defines and describes other data.

NISO	National Information Standards Organization
OASIS	The Organization for the Advancement of Structured Information Standards
Registration Authority (RA)	Any organization authorized to register data elements.
responsible organization	The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the data element is specified.
RFC1807	An Internet Engineering Task Force Request For Comments relating to the creation of a common format for Bibliographic Records
Schema	Second generation XML technology for defining the structure and constraints for the content of an XML document. Significantly extends the capabilities of XML 1.0 document type definitions. (See also DTD)
SOAP	Simple Object Access Protocol. The W3C is currently developing an official recommendation (i.e. standard) that defines this simple XML-based protocol.
submitting organization	The organization or unit within an organization that has submitted the data element for addition, change, or cancellation/withdrawal in the data element dictionary.
taxonomy	Classification according to presumed natural relationships among types and their subtypes.
UDDI	Universal Description, Discovery and Integration
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
URI	Uniform Resource Identifier. URI's are defined in IETF RFC2396.
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition (see Schema)
Z39.50	Interoperability standard for information retrieval in a distributed environment. The Z39.50 protocol is used extensively for interoperability purposes in the Library community.

Section 8: Appendix B – References

[AIHW02]   Australian Institute of Health and Welfare Knowledgebase. Available at
           http://www.aihw.gov.au/knowledgebase/, Accessed October 2002

[BERN98]   Berners-Lee, T., Fielding, R., Masinter, L., *RFC2396 : Uniform Resource
           Identifiers (URI) : Generic Syntax*. Internet Engineering Task Force, 1998.
           Accessed November 2002.

[BERG02]   Berglund, A., Boag, S., Chamberlin, D., Fernandez, M.F., Kay, M., Robie,
           J., and Siméon, J., *XML Path Language (XPath): Version 2.0, W3C
           Working Draft*. World Wide Web Consortium. Available at
           http://www.w3.org/TR/xpath20/. Accessed September, 2002.

[BOAG02]   Boag, S., Chamberlin, D., Fernandez, M.F., Florescu, D., Robie, J., and
           Siméon, J., *XQuery 1.0: An XML Query Language, W3C Working Draft*,
           World Wide Web Consortium. Available at
           http://www.w3.org/TR/xquery/. Accessed September, 2002.

[BOUR02]   Bourret, R.P. *XML Database Products: Native XML Databases*. Available
           at: http://www.rpbourret.com/xml/ProdsNative.htm. Accessed September,
           2002.

[BROD01]   Brody T., Jiao, Z., Hitchcock, S., Carr, L., and Harnad, S. *Enhancing OAI
           Meta-data for Eprint Services: two proposals*. Open Citation Project, IAM
           Research Group, Department of Electronics and Computer Science,
           University of Southampton SO17 1BJ, UK. . Available at:
           http://opcit.eprints.org/ecdl-oai/ecdl-submit.pdf.

[BROD02]   Brodsky, J., Hunt, B., Khoury, S., and Popkin, L. *The Information and
           Content Exchange (ICE) Protocol Version 1.1*. IDEAlliance. Available at:
           http://www.icestandard.org/Spec/SPEC-ICE-200000701.html. Accessed
           September, 2002.

[CANC02]   *CAREO: Campus Alberta Repository of Educational Objects*. The
           University of Calgary, 2002.

[CHEN02]   Chen, S. and Choo, C., "*ADL Server with OAI Capabilities: LOVE*".
           University of Florida at Gainesville, 2002. Accessed September, 2002.

[CLAR99]   Clark, J. and DeRose, S., *XML Path Language (XPath): Version 1.0, W3C
           Recommendation*, World Wide Web Consortium. Available at
           http://www.w3.org/TR/xpath. Accessed September, 2002.

[CPEB02] *Cover Pages Technology Reports: Electronic Business XML Initiative (ebXML)*, hosted by OASIS. Available at: http://xml.coverpages.org/ebXML.html. Accessed October, 2002.

[CPRN02] *Cover Pages Technology Reports: RosettaNet*, hosted by OASIS. Available at: http://xml.coverpages.org/rosettaNet.html. Accessed October, 2002.

[CPUD02] *Cover Pages Technology Reports: Universal Description, Discovery, and Integration (UDDI)*, hosted by OASIS. Available at: http://xml.coverpages.org/uddi.html. Accessed October, 2002.

[DUBL02] The Dublin Core Meta-data Initiative. Available at: http://dublincore.org. Accessed October 2002.

[DYCK02] Dyck, T., *Going Native: XML Databases*. PC Magazine, June 2002. Available at: http://www.pcmag.com/print_article/0,3048,a=27479,00.asp. Accessed September, 2002.

[EBBP01] *ebXML Business Process Specification Schema, Version 1.01*, May 2001. Available at: http://www.ebxml.org/specs/ebBPSS.pdf. Accessed November, 2002.

[EBCP02] *ebXML Collaboration-Protocol Profile and Agreement Specification, Version 2.1*, September 2002. Available at: http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf. Accessed November, 2002.

[EBMS02] *ebXML Message Service Specification, Version 2.0 rev C*, February, 2002. Available at: https://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0rev_c.pdf. Accessed October, 2002.

[EBRM01] *ebXML Registry Information Model*, December, 2001. Available at: https://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrim.pdf. Accessed September, 2002.

[EBRQ01] *ebXML Requirements Specification Version 1.06*, 2001 Available at: http://www.ebxml.org/specs/ebREQ.pdf. Accessed September, 2002.

[EBRS01] *OASIS/ebXML Registry Services Specification, Version 2.1*, December, 2001. Available at: http://www.oasis-open.org/committees/regrep/documents/2.1/specs/ebrs.pdf. Accessed October, 2002.

Technologies for the Design of Learning Object Repositories
                                                                                              021751

[EBRT02]   *OASIS ebXML Registry TC webpage*. Available at:

            http://www.oasis-open.org/committees/regrep/documents/revised-charter-0627.shtml. Accessed October, 2002

[EBTA01]   *ebXML Technical Architecture, Version 1.04*, February 2001. Available at: http://www.ebxml.org/specs/ebTA.pdf. Accessed October, 2002

[EPAE02]   U.S. EPA Environmental Data Registry. Available at http://www.epa.gov/edr, Accessed October 2002

[EPAM02]   U.S. EPA MetaPro Meta-data Registry Builder. Available at http://www.epa.gov/metapro/, Accessed October 2002

[FAAD02]   FAA Data Registry, Available at http://www1.faa.gov/aio/InfoMgmt/projects.htm#FAAdatreg, Accessed October 2002

[FERR01]   Ferris, C., *ebXML Message Service*. Available at: http://www.gca.org/papers/xmleurope2001/papers/html/s09-3.html. Accessed: September 2001

[GILL99]   Gillman, D.W. and Appel, M.V. *The Statistical Meta-data Research at the Census Bureau.* Bureau of the Census. Available at: http://www.fcsm.gov/99papers/gillman.pdf. Accessed November, 2002.

[HARO01]   Harold, E.R., *The XML Model vs. the Relational Model from XSLT 2.0 and Beyond*. Available at: http://www.cafeconleche.org/slides/sd2002west/xslt2/74.html. Accessed June, 2002.

[HUSK01]   Hsu, K., *ebXML, Web Services, UDDI and SOAP*. IBM Global Services, 2001. Available at: http://www.twtec.org.tw/Download/Report/ebXML,%20Web%20Services%20and%20SOAP.pdf. Accessed October, 2002.

[IMBP02]   *IMS Digital Repositories Interoperability – Core Functions Best Practices Guide, Version 1.0 Public Draft Specification.* IMS Global Learning Consortium, Inc., Burlington, MA. Available at: http://www.imsglobal.org/digitalrepositories/driv1p0pd/imsdri_bestv1p0pd.html. Accessed November, 2002.

[IMXB02]   *IMS Digital Repositories Interoperability – Core Functions XML Binding, Version 1.0 Public Draft Specification.* IMS Global Learning Cosortium, Inc., Burlington, MA. Available at: http://www.imsglobal.org/digitalrepositories/driv1p0pd/imsdri_bindv1p0pd.html. Accessed November, 2002.

[IMSG02]   *IMS Digital Repositories Interoperability - Core Functions Information Model, Version 1.0 Public Draft Specification*. IMS Global Learning Consortium, Inc., Burlington, MA. Available at http://www.imsglobal.org/digitalrepositories/driv1p0pd/imsdri_infov1p0pd.html. Accessed October, 2002

[ISOM02]   *ISO/IEC 11179, Information Technology – Meta-data Registries (MDR), Draft Specification.* ISO/IEC JCT1 SC32/WG2, Available at http://pueblo.lbl.gov/~olken/X3L8/drafts/draft.docs.html, Accessed August 2002

[LAGO02]   Lagoze, D. and Van de Somple, H., *The Open Archive Initiative Protocol for Meta-data Harvesting, Document Version 2002-05-15T11:00:00Z – cjl*. Available at http://www.openarchives.org/OAI/openarchivesprotocol.htm, Accessed May 2002

[LONM02]   Learning Objects Networks, Inc., *Message Manifest Specification v1.2*, March, 2002.

[MERT01]   Mertz, D., *Understanding ebXML*. Available at: http://www-106.ibm.com/developerworks/library/x-ebxml/index.html. Accessed September, 2002

[MCAR01]   McArthur, D., Giersch, S. and McClelland, M., *IMS Meta-data in a digital library*. Available at: www.imsproject.org/membersexchange/ME_ilumina.pdf. Accessed September, 2002.

[MCLE02]   McClelland, M., McArthur, D., Giersch, S. and Geisler, G. *Challenges for Service Providers When Importing Meta-data in Digital Libraries.* D-Lib Magazine, April 2002. Available at: http://www.dlib.org/dlib/april02/mcclelland/04mcclelland.html Accessed May 2002.

[MOEN02]   Moen, W., *The ANSI/NISO Z39.50 Protocol: Information Retrieval in the Information Infrastructure.* Available at: http://www.cni.org/pub/NISO/docs/Z39.50-brochure/. Accessed November, 2002.

[NCIT98]   *NCITS L8 Introduction*. Available at:
             http://pueblo.lbl.gov/~olken/X3L8/L8intro.html. Accessed November,
             2002.

[OAIR02]   *OAI Registered Data Providers*. Available at:
             http://www.openarchives.org/Register/BrowseSites.pl. Accessed October,
             2002

[PASK02]   Paskin, N., *Digital Object Identifier: implementing a standard digital identifier
             as the key to effective digital rights management.* The International DOI
             Foundation, April, 2002.

[RFC1807]   *An XML Schema for the rfc1807 meta-data format.* Available at:
             http://www.openarchives.org/OAI/2.0/guidelines-rfc1807.htm

[RICH01]   Richards G. and Hatala, M., *POOL, POND and SPLASH: A Peer to Peer
             Architecture for Learning Object Repositories.* Technical University of British
             Columbia, 2001. Available at: http://www.edusplash.net. Accessed
             September, 2002

[SCAN02]   Scannell, E. and Krill P., *Databases Wrestle XML*, InfoWorld, June 2002.
             Available at:
             http://www.infoworld.com/articles/hn/xml/02/06/24/020624hnxmldb2.xml.

             Accessed September, 2002.

[SOMF02]   *SOAP Version 1.2 Part 2: Messaging Framework*. Available at:
             http://www.w3.org/TR/soap12-part1/. Accessed September, 2002

[SOAD02]   *SOAP Version 1.2 Part 2: Adjunct*s. Available at:
             http://www.w3.org/TR/soap12-part2/. Accessed September, 2002

[SOUZ00]   Souzis, A., Popkin, L., Khoury, S., and Hunt, B., *The ICE Implementation
             Cookbook*. IDEAlliance, Alexandria, VA. pp. 45. Available at:

             http://www.icestandard.org/servlet/RetrievePage?site=ice&page=specifications
             Accessed September, 2002.

[STAK01]   Staken, K. *Introduction to Native XML Databases*, XML.com, October, 2001.
             Available at: http://www.xml.com/lpt/a/2001/10/31/nativexmldb.html.
             Accessed August, 2002.

[TEUD02]   The Stencil Group, *The Evolution of UDDI: UDDI.org White Paper*, July
             2002. Available at:
             http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf. Accessed
             September, 2002.

[WEBB98]   *The Information and Content Exchange (ICE) Protocol, NOTE-ice-19981026*, World Wide Web Consortium. Available at http://www.w3.org/TR/NOTE-ice.htm. Accessed October, 2002.

[WILL01]   Williams, K. *XML for Data: Native XML databases: a bad idea for data?*, IBM DeveloperWorks. Available at: ftp://www6.software.ibm.com/software/developer/library/x-xdnat.pdf. Accessed September, 2002.

[XMLB01]   *XML Base*, June 2001. Available at: http://www.w3.org/TR/xmlbase/. Accessed September, 2002.

Technologies for the Design of Learning Object Repositories
021751