Date : 02/06/2006

A Perspective on Metadata and Ontology Curriculum
In Library and Information Science Education

Sam Gyun Oh
Department of Library & Information Science,
Sungkyunkwan University, Seoul, Korea

Meeting:	**107 *Off-site:* Education and Training**
Simultaneous Interpretation:	No

WORLD LIBRARY AND INFORMATION CONGRESS: 72ND IFLA GENERAL CONFERENCE AND COUNCIL

20-24 August 2006, Seoul, Korea

http://www.ifla.org/IV/ifla72/index.htm

Abstract

This paper will discuss a perspective on how to structure 'metadata and ontology' curriculum in Library and Information Science (LIS) education. This curriculum is composed of three sequel classes that deal with major concepts and technologies related to this field: 1) Metadata Basics; 2) Metadata Schema Design; and 3) Ontology Modeling and Design. The first class covers topics such as XML namespaces, uniform resource identifiers (URI), standard metadata schemas and application profiles, and metadata registries. The second class deals with design principles of metadata schemas and application profiles, and provides students with knowledge of how to implement application profiles using XML technology. The major focus of the second class is to achieve syntactic interoperability among diverse metadata schemas and application profiles. The third class focuses its attention to semantic interoperability among different metadata and ontology. It builds on the previous two classes and further elaborates on concepts and technology such as RDF, RDF Schema, Web Ontology Language (OWL), and Topic Maps. The ontology design is viewed as a way to achieve advanced and semantic data modeling of

complex data that exist in the real world. Successful completion of these classes will provide students with competency in designing metadata and ontology. The recommended tools for these classes are XMLSpy for designing metadata schemas and application profiles, Protégé for designing RDF/OWL ontology, and Ontopoly for designing Topic Map ontology.

The purpose of this paper is to outline major components of course(s) related to metadata and ontology curriculum in LIS. It proposes three sequel classes to provide students with necessary conceptual and technical understanding of metadata and ontology needed for managing information and knowledge in the semantic web environment.

Course 1: Metadata Basics

The first lecture should cover the following topics: metadata definitions, metadata development overview, XML namespaces, uniform resource identifiers (URI), metadata types and functions, international standard metadata schemas, application profiles, metadata record creation and tools, and metadata registries.

1.1 Metadata Definitions

This portion should cover various definitions of metadata including the following, but not limited to:

- NISO's Definition: "Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information."
- The American Library Association (ALA) Definition: Metadata are structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities.
- Various definitions of metadata are collected at the following site: http://www.libraries.psu.edu/tas/jca/ccda/tf-meta6.html#appx2

1.2 Metadata Development Overview

This portion of the lecture can be divided into two groups: 1) pre-Internet era of metadata – MARC ; and 2) the Internet arena and evolving metadata standard including bibliographic objects (MARC, MODS, Dublin Core, TEI Header), archival inventories and registers (EAD), geospatial objects (FGDC), museum and visual resources (CDWA, VRA Core, CIMI), educational material (LOM), software implementation (CORBA), E-Commerce (INDECS, ONIX), Media-specific (MPEG4, MPEG7), and encoding descriptive, administrative, and structural metadata regarding objects within a digital library (METS).

1.3 XML Namespaces

This section should deal with what we mean by XML namespace, and then introduce how Dublin Core Metadata Initiative (DCMI) has adopted XML namespace to maintain its terms. This example should help students to see how metadata elements can be managed distinctively in a semantic web environment.

W3C Definition: An XML namespace is a collection of names, identified by a URI reference, that are used in XML documents as element types and attribute names. The use of XML namespaces to uniquely identify metadata terms allows those terms to be unambiguously used across applications, promoting the possibility of shared semantics.

1.4 Importance of Persistent Identifiers (URI)

It is imperative to assign unique and persistent identifiers to classes, properties, and

resources. Without accomplishing this task, all the efforts to manage digital information will not be successful. This section should introduce the major URI system available including DOI initiated by International DOI foundation, PURL by DCMI, and UCI by Korea Computerization Agency. It should also discuss the handle resolution system developed by CNRI.

DCMI Namespace Policy

An XML namespace [XML-NAMES] is a collection of names, identified by a URI reference [RFC2396], that are used in XML documents as element types and attribute names. The use of XML namespaces to uniquely identify metadata terms allows those terms to be unambiguously used across applications, promoting the possibility of shared semantics. DCMI adopts this mechanism for the identification of all DCMI terms. The following specifies the conventions used for identifying current and future DCMI namespaces. All DCMI recommendations that make use of namespaces will conform to this recommendation.[1]

Namespace URIs used by the DCMI

The URI of the namespace for all DCMI elements that comprise the Dublin Core Metadata Element Set, Version 1.1 is:

http://purl.org/dc/elements/1.1/

The URI of the namespace for all DCMI elements and DCMI qualifiers (other than those elements defined in the Dublin Core Metadata Element Set, Version 1.1 above) is:

http://purl.org/dc/terms/

The URI of the namespace for DCMI terms defined in the DCMI Type Vocabulary is:

http://purl.org/dc/dcmitype/

Therefore, the three currently approved DCMI namespace URIs are:

http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1 (15 elements)
http://purl.org/dc/terms/	DCMI elements and DCMI qualifiers (other than those elements defined in the Dublin Core Metadata Element Set, Version 1.1 above)
http://purl.org/dc/dcmitype/	DCMI terms in the DCMI Type Vocabulary (a DCMI controlled vocabulary)

All DCMI namespace URIs will resolve to a machine-processable DCMI term declaration for all the terms within that namespace.

The URI for each DCMI term is constructed by appending the term *name* to the namespace URI for that term. For example:

[1] http://dublincore.org/documents/dcmi-namespace/index.shtml [retrieved on 5/15/2006]

http://purl.org/dc/elements/1.1/title
is the URI for the Title element in the Dublin Core Metadata Element Set, Version 1.1,

http://purl.org/dc/terms/extent
is the URI for the Extent qualifier in the Dublin Core Qualifiers recommendation and

http://purl.org/dc/dcmitype/Image
is the URI for the Image term in the DCMI Type Vocabulary. Each DCMI term can be so identified.

All future DCMI namespace URIs (additional DCMI controlled vocabularies for example) will conform to this pattern:
http://purl.org/dc/namespace_label/

1.5 Metadata Types and Functions

This section should introduce NISO definition of main types of metadata including:

- descriptive metadata
- structural metadata
- administrative metadata
- rights management metadata
- preservation metadata

It will also be useful to discuss Getty's definition of types of metadata:

- administrative metadata
- descriptive metadata
- preservation metadata
- technical metadata
- use metadata

After this, the major functions of metadata can be discussed further:

- resource discovery
- organizing e-resources
- facilitating interoperability
- digital identification
- archiving preservation

When we discuss different types of metadata, this section cannot be completed without discussing simple Dublin Core (DC) and qualified DC[2].

1.6 International Standard Metadata Schemas

Metadata schemas (schemes) generally specify names of elements and their semantics. They may also specify rules for how content must be formulated, representation rules for content, and allowable content values. The following list needs to be discussed in this section. This list and grouping is offered by Marcia Zeng[3]:

- Metadata Schemas for Bibliographic Description
  - MARC, MODS(Metadata Object Description Schemas), MARC XML
  - Dublin Core

[2] http://dublincore.org/documents/dcmi-terms/ [retrieved on 05/15/2006]
[3] http://www.slis.kent.edu/%7Emzeng/metadatabasics/completelist.htm [retrieved on 05/12/2006]

- GILS (Government Information Locator Service/Global Information Locator Service)
- RFC 1807 (Format for Bibliographic Records)
- TEI Headers (Text Encoding Initiative)
- W3C PICS (Platform for Internet Content Selection)

- Metadata Schemas for Images and Objects
  - Categories for the Description of Works of Art (CDWA)
  - Consortium for the Computer Interchange of Museum Information (CIMI)
  - VRA Core Categories version 3.0
  - NISO Data Dictionary for Technical Metadata for Digital Still Images (released June 7, 2002 as a Draft Standard for Trial Use. Z39.87-2002)
  - OBJECT ID

- Metadata Schemas for Geospatial Data
  - Content Standards for Digital Geospatial Metadata (CSDGM)

- Metadata Schemas for Archives
  - EAD (Encoded Archival Description) DTD
  - Recordkeeping Metadata Standard for Commonwealth Agencies (1999)

- Metadata Schemas for E-commerce and Right Management
  - The INDECS project
  - ONIX (Online Information Exchange)
  - Rights Metadata, by the Book Industry Communication
  - Publishing Requirements for Industry Standard Metadata (PRISM)
  - DOI -- Digital Object Identifier, by the International DOI Foundation

- Educational Purpose Metadata Schemas
  - Instructional Management Systems (IMS): IMS Learning Resource Meta-data Specification
  - Learning Object Metadata (LOM)
  - GEM Element Set, The Gateway to Educational Materials
  - DC-Ed (Dublin Core Education Working Group) Extensions
  - The Sharable Content Object Reference Model (SCORM)

- Media-Specific Metadata Schemas
  - MPEG-4 and MPEG-7 for Audio and Video
  - Public Broadcasting Metadata Initiative
  - Standard Media Exchange Framework (SMEF), BBC

- Metadata Schemas for Preservation of digital objects
  - Preservation Metadata for Digital Objects
  - CEDARS Project: CEDARS Preservation Metadata Elements
  - Preservation Metadata for Digital Collections by National Library of Australia.
  - Networked European Deposit Library. Metadata for Long Term Preservation

- Metadata Schemas for Collection Level Description
  - EAD (Encoded Archival Description) DTD
  - Z39.50 Profile for Access to Digital Collection

- Metadata Schemas for Numeric Data
  - ICPSR Data Documentation Initiative (DDI)
  - Standard for Survey Design and Statistical Methodology Metadata (SDSM),

1.7 Metadata Record Creation and Tools

This section should deal with different tools that help one to create metadata records. The following is a sample list to consider:

- Dublin Core Tools
- FGDC Metadata Tools
- OAI-Specific Tools
- RDF Editors and Tools
- TEI Software
- Customized Templates for EAD-Encoded Finding Aids

1.8 Application Profiles

According to Heery and Patel[4], application profiles are defined as schemas which consist of data elements drawn from one or more namespaces, combined together by implementers, and optimized for a particular local application. This section should deal with the principles of designing application profiles and introduce different types of application profiles. Application profiles are useful as they allow the implementer to declare how they are using standard schemas. In the context of working applications where there is often a difference between the schema in use and the 'standard' namespace schema.

1.9 Dumb-Down Principle

When designing an application profile, it is important to understand "dumb-down" principle. The so called "Dumb-Down Principle" simply means that in any use of a qualified DC element, the qualifier may be dropped and the remaining value of the element should still be a term that is useful for discovery. For example, there are several date qualifiers that might be used to enhance the precision of various dates associated with a resource. Dropping the date qualifier (for example, Date-Created) will still leave a useful date for discovery, though perhaps not quite as useful as if the qualifier were included. Similarly, the specification of a subject term from LCSH, for example, is still useful even if one does not know it was selected from a controlled vocabulary. The basic idea is that qualifiers should improve the precision of a piece of metadata, but the metadata should still be useful even without that extra precision (that is, dropping the qualifier has 'dumbed-down' the metadata.)[5]

1.10 Metadata Registries

The main purpose of metadata registry is to discover elements associated with metadata schema and application profiles. ISO/IEC 11179 provides the standard guide to creating metadata registry (MDR). This section should discuss most DCMI metadata registry[6] because it is most widely used internationally. DCMI MDR[7] is designed to promote the discovery and reuse of existing metadata definitions. It provides users, and applications, with an authoritative source of information about the Dublin Core element set and related vocabularies. This simplifies the discovery of terms and related definitions, and illustrates the relationship between terms. The reuse

[4] http://www.ariadne.ac.uk/issue25/app-profiles/ [retrieved on 05/08/2006]
[5] http://dublincore.org/resources/faq/#dumbdown [retrieved on 05/08/2006]
[6] http://www.dublincore.org/dcregistry/ [retrieved on 5/8/2006]
[7] http://dublincore.org/dcregistry/ [retrieved on 5/15/2006]

of existing metadata terms is essential to standardization, and promotes greater interoperability between metadata element sets. The discovery of existing terms is an essential, and prerequisite, step in this process. This application promotes the wider adoption, standardization and interoperability of metadata by facilitating its discovery, and reuse, across diverse disciplines and communities of practice.

Course 2: Designing Metadata Schemas and Application Profiles

The second class deals with design principles of metadata schemas and application profiles, and provides students with knowledge of how to implement application profiles using XML technology. The major focus of the second class is to achieve syntactic interoperability among diverse metadata schemas and application profiles. The recommended tool for this class is XMLSpy(http://www.xmlspy.com) .

2.1 Defining Metadata Schema

ALA Definition of Metadata Schema: A **metadata schema** provides a formal structure designed to identify the knowledge structure of a given discipline and to link that structure to the information of the discipline through the creation of an information system that will assist the identification, discovery, and use of information within that discipline

2.2 XML DTD

This section should discuss the functions and strength of XML technology. The main advantage of using XML is in its ability to provide data interoperability regardless of platforms. The major shortcoming of DTD is in its inability to support namespaces and very limited support of data types. These shortcomings of XML DTD should be dealt with in detail. Recommended topics for this section are as follows:

- Fundamental concepts of XML technology
- What makes up an XML system?
  - XML document
  - XML documents type definition (DTD)
  - XML parser
  - XML application
- XML documents in depth
  - Well-formed XML documents
  - Valid XML documents
- Creating XML documents
  - Element tags
  - Attributes
  - Comments
  - Processing instructions
  - Character data section
  - Documents type definition
- DTD Basics
  - Element type declaration
  - Descriptions of element type content models
  - Attribute list declaration
  - Attribute default
  - Entity declaration
- Shortcomings of DTD
  - No support for namespaces
  - Limited support for data types

- Limited support for cardinality

2.3 XML Schemas

W3C XML schema is designed to overcome all the shortcomings that are discussed in the DTD section. This section should cover the following topics:

- Schema Uses and Development
  - What schemas do for XML?
  - W3C XML schema
- Using Predefined Simple Datatypes
  - Lexical and value spaces
  - Whitespace processing
  - String datatypes
  - Numeric datatypes
  - Date and time datatypes
  - List types
- Creating Simple Datatypes
  - Derivation by restriction
  - Derivation by list
  - Derivation by union
- Creating Complex Datatypes
  - Simple versus complex types
  - Examining the landscape
  - Simple content models
  - Complex content models
  - Mixed content models
- Creating Building Blocks
  - Schema inclusion
  - Schema inclusion with redefinition
  - Other alternatives
- Defining Uniqueness, Keys, and Key References
  - xs:ID and xs:IDREF
  - XPath-based identity check
  - ID/IDREF versus xs:key/xs:ekyref
  - Using xs:key and xs:unique as co-occurrence constraints
- Controlling Namespaces
  - Namespaces present two challenges to schema languages
  - Namespace declarations
  - To qualify or not to qualify?
  - Referencing other namespaces
  - Namespace behavior of imported components
  - Importing schema with no namespaces
  - Allowing any elements or attributes from a particular namespace

2.4 Resource Description Framework(RDF)

The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be *identified* on the Web, even when they cannot be directly *retrieved* on the Web. Examples include information about items available from on-line shopping facilities

(e.g., information about specifications, prices, and availability), or the description of a Web user's preferences for information delivery. RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created. RDF is based on the idea of identifying things using Web identifiers (called *Uniform Resource Identifiers*, or *URIs*), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a *graph* of nodes and arcs representing the resources, and their properties and values.[8] The following topics should be discussed in this section:

- RDF: An Introduction
  - The semantic web and RDF: a brief history
  - The specifications
  - When to use and not use RDF
  - Some uses of RDF/XML
  - Related technologies
- RDF: Heart and Soul
  - The search for knowledge
  - The RDF triple
  - The basic RDF data model and the RDF graph
  - URIs
  - RDF serialization: N3 and N-triples
- The Basic Elements within the RDF/XML Syntax
  - Serializing RDF to XML
  - RDF blank nodes
  - URI references
  - Representing structured data with rdf:value
  - The rdf:type property
  - RDF/XML shortcuts
  - RDF datatypes
  - RDF/XML: separate documents or embedded blocks
- Specialized RDF Relationships: Reification, Container, and Collections
  - Containers
  - Collections
  - Reification

2.5 Implementing Metadata Schemas

Learning the syntax of XML DTD and schema can be complex and tedious. Using a good editor can enhance the effectiveness of acquiring competency in those languages. The author recommends the use of XMLSpy editor to implement metadata schemas using XML DTD and schema. A gentle introduction of XMLSpy editor will help students to learn the concept of XML namespaces and how to reuse of different namespaces can actually be implemented using XML technology. This section should explain how to implement the well-known metadata schemas using the following references:

- Expressing Simple Dublin Core in RDF/XML:

[8] http://www.w3.org/TR/rdf-primer/#intro [retrieved on 5/12/2006]

http://www.dublincore.org/documents/2002/07/31/dcmes-xml/

- Guidelines for implementing Dublin Core in XML: http://www.dublincore.org/documents/2003/04/02/dc-xml-guidelines/
- Expressing Qualified Dublin Core in RDF / XML: http://www.dublincore.org/documents/2002/05/15/dcq-rdf-xml/
- Expressing Dublin Core Elements Using XML Schema: http://lis.skku.ac.kr/ohs/xmlschema/edna/dc.xsd
- Expressing Qualified Dublin Core Using XML Schema: http://lis.skku.ac.kr/ohs/xmlschema/edna/dcterms.xsd

2.6 Implementing Application Profiles

This section should cover the characteristics of application profiles. Heery and Patel[9] distinguish 'namespace schema' from 'application profile schema'. Namespace schema contains all those elements defined by the managing body or registration authority (whatever that might be) for a particular namespace. Application profiles are tailored for particular implementations and will typically contain combinations of sub-sets of one or more namespace schemas.

'Namespace' is defined within the W3C XML schema activity and allows for unique identification of elements. Within the W3C XML and RDF schema specifications, namespaces are the domain names associated with elements which, along with the individual element name, produce a URL that uniquely identifies the element. In W3C terms the namespace does not have to be a 'real' registration authority, nor does the element identifying URL need to point to a 'real' web address. However in order to ensure a well managed metadata environment, it is strongly argued that the namespace should refer to a real registration authority that takes responsibility for the declaration and maintenance of their schema.

There is a continuum of formality in such registration authorities from those where the authority is an internationally recognized standards body through to those where the authority derives from national or de facto standards, and at the other end of the continuum, to self-contained schemas defined within a local project or service. By means of 'namespace' we can

- Identify the management authority for an element set
- Support definition of unique identifiers for elements
- Uniquely define particular data element sets or vocabularies

The DESIRE project[10] constructed a prototype metadata registry schema with a data model within which 'namespace' consisted of three parts:

- Registration authority
- Namespace concept
- Namespace

It may be useful to consider how, in combination, these entities might help us to identify well managed metadata element sets. By use of these entities, a distinctive element set can be identified by a 'namespace', that namespace may have different instantiations over time (versioning) each of which require a separate namespace but all are associated with a namespace concept. A namespace concept is therefore a

[9] http://www.ariadne.ac.uk/issue25/app-profiles/ [retrieved on 5/12/2006]
[10] http://desire.ukoln.ac.uk/registry/ [retrieved on 5/12/2006]

grouping mechanism for successive versions of a namespace. Each namespace and namespace concept is associated with a registration authority. Within the DESIRE registry this enabled us to consider that one registration authority might have several different element sets associated with it.

Schema application profiles are distinguished by a number of characteristics. They

- May draw on one or more existing namespaces
- Introduce no new data elements
- May specify permitted schemes and values
- Can refine standard definitions

The application profile can refine the definitions within the namespace schema, but it may only make the definition semantically narrower or more specific. This is to take account of situations where particular implementations use domain specific, or resource specific language.

This section should explain how to construct application profiles using XML schemas. The following sample coding can be useful in helping students to see the connection between the concept of application profiles and implementation of them.

- Reusing declared simple DC: http://lis.skku.ac.kr/ohs/xmlschema/edna/dc.xsd
- Reusing declared qualified DC: http://lis.skku.ac.kr/ohs/xmlschema/edna/dcterms.xsd
- Defining EdNA namespace in XML schema: http://lis.skku.ac.kr/ohs/xmlschema/edna/edna.xsd
- Defining EdNA application profile in XML schema: http://lis.skku.ac.kr/ohs/xmlschema/edna/ednaAPP.xsd

Course 3: Ontology Modeling and Design

The third class focuses its attention to semantic interoperability among different metadata and ontology. It builds on the previous two classes and further elaborates on concepts and technology related to Topic Maps, RDF Schema, and Web Ontology Language (OWL). The ontology design will be viewed in this class as a way to achieve advanced and semantic data modeling of complex data that exist in the real world. Successful completion of these classes will provide students with competency in designing metadata and ontology. The recommended tools for these classes are Protégé for designing RDF/OWL ontology and Ontopoly for designing Topic Map ontology.

3.1 What Is Ontology?

There are several definitions on ontology: 1) Ontology is a conceptual data model used to link together other, more granular data models in order to reach agreement on data meanings – either by committing to shared vocabularies or by explicitly modeling vocabulary differences, context, and meaning; 2) Ontology is a model of any bounded region

3.2 Identity Issues

This section should cover the issue of identity. However, RDF has not been clear about whether a URI like http://www.w3.org/Consortium identifies the W3C or a web page about the W3C. Throughout RDF, strings like http://www.w3.org/1999/02/22-rdf-

syntax-ns#type are used with no consistent explanation of how they relate to the web. Why is this important? Because without clarity on this issue, it is impossible to solve the challenge of the Semantic Web, and it is impossible to implement scaleable Web Services. It is impossible to achieve the goals of "global knowledge federation" and impossible even to begin to enable the aggregation of information and knowledge by human and software agents on a scale large enough to control infoglut.

Ontologies and taxonomies will not be reusable unless they are based on a reliable and unambiguous identification mechanism for the things about which they speak. The same applies to classifications, thesauri, registries, catalogues, and directories. Applications (including agents) that capture, collate or aggregate information and knowledge will not scale beyond a closely controlled environment unless the identification problem is solved. And technologies like RDF and Topic Maps that use URIs heavily to establish identity will simply not work (and certainly not interoperate) unless they can rely on unambiguous identifiers. A solution to the "identity crisis of the Web" is clearly essential.

There are two ontology languages that this class should discuss. One is topic map which is ISO standard ontology language, and the other is RDF/OWL which is the language used by W3C for ontology development. This section introduces major concepts of topic maps and RDF/OWL and presents other topics that need to be included in this class.

3.3 Topic Maps

Topic Maps is the ISO standard for describing a knowledge structure and representing the links among related information resources[11]. Topic Maps has been established as an ISO/IEC standard since 2000. Topic Maps has employed XML as a main Topic Maps syntax and most Topic Maps tools support this syntax.

Topic Maps, which is very similar to the traditional back-of-the book index, is an ontology language that has potential to satisfy representation needs discussed in traditional indexing within library science and also for representing sub-structures of the semantic web. Topic maps employ the concept of ontology, which represents implicit information structure explicitly. It represents not only relationships found expressed in thesauri, indexes, and taxonomies, but also attempts to support management of knowledge and information layers effectively and efficiently. The main concepts of Topic Maps consist of topics, occurrences, and associations.

Topics

A topic, which is a very similar concept with that of a subject in knowledge organization, can be anything such as a person, an object, or a concept. For example, "University of Washington" is an instance of a *school*, "John Miller" is a *student*, "Stuart Kim" is *a professor*, "Metadata Registry Project" is *a project*, "Metadata" is a *research area* – all can be defined as topics. A topic can have more than one name such as base name, display name, and sort name, and different topics can have the same name. This is the main difference function between taxonomy and thesaurus. Furthermore, a topic can play a role in integrating similar instances by types. These types are called topic types. In the foregoing example, *student, professor, research area, and project* are Topic Types. That is, a topic and a topic type has a 'IS-A' relationship.

[11] Oh et al. Ontology-Driven Knowledge Organization – Enhancing UDDI Web services in Korea using Topic Maps, *Proceedings of the 68th Annual ASIST meeting*. October 28-November 2. 2005. Charlotte, North Carolina. USA

Occurrences

Occurrences link information resources to topics. An occurrence is compared to a "subject" property in Dublin Core. While a subject property in Dublin Core is intended to assign subjects to a resource, an occurrence in Topic Maps links resources to a topic. Occurrences can be grouped in a meaningful way. Examples of external occurrence types include *homepage*, *review*, *synopsis*, and *email address*. Examples of internal occurrence types are *description*, *creator*, *version*, and *date*, etc.

For example, a topic, "Clay Shirky" as a professor, is linked to many different resources such as *phone number, picture, and web site*. That is, resources to be linked to each topic can be grounded by occurrence types. Topic Maps provides an internal structure to support the second objective of cataloging[12], which requires display of search results in meaningful groups for intended users. The second purpose of cataloging asserts that merely displaying relevant information is not good enough, but any good search system must provide ways to group search results in a useful way for users. The occurrence types in Topic Maps can be used to achieve this goal.

Associations

An association represents a relationship between topics. There is no limitation in terms of establishing associations between topics, and any relationships can be defined. It can be compared to relationships in a thesaurus, but thesaurus relationships are limited to some specific types such as broad term/narrow term, and related terms.

For example, "Stuart Kim" *teaches* LIS550, "John Miller" *takes* LIS550, and "Stuart Kim" is *involved in* "Metadata Registry Project." *Teaches, takes,* and *involved in* are *association types* used to link different topics. In addition, each topic can define a role that it plays in an association, which refers to *association roles*. In the '*teaches*' association, "Stuart Kim" plays the role as a professor and "LIS 550" as a class. Each topic can play different roles in different associations. "Stuart Kim," is an instance of a professor, has several associations with other topics, and is linked to numerous information resources as occurrences.

The following topics can be discussed in this section:

- How Topic Maps achieve their purpose
  - Reification
  - Topic Characteristics: Scope, Names, Occurrences, Associations and Roles, Topic Identity, Class-Instance Association and Topic Types.
- Historical Development – from Topic Navigation Maps to XTM
- Model and Syntax
  - The XTM Conceptual Model
  - The XTM Syntax
- XTM Processing
- Topic Maps in Practice
  - Topic Map Creation
  - Topic Map Merging
  - Association Template
  - NewsML and Topic Maps

[12] Cutter, C.A. 1875. *Rules for a printed dictionary catalogue*. Washington, DC: Government Printing Office.

3.3 RDF Schema and OWL

RDF Schema

RDF does not provide a way to specify resource and property types. Once you have some instance of a resource, you can associate a property with it. However, you cannot express what kind of property and resource it is. That is the job of RDF schema. RDF schema develops classes for both resources and properties. Defining objects in classes is very much attractive for resource description because it allows more special classes of resources to be derived from more general ones. Since everything that RDF describes is a resource rdfs:Resource is the most basic resource class. Every resource is an rdfs:Resource. The property rdf:type is used to say that a resource is of a certain type. The idea of a type of resource is expressed in the concept of a class. A class is a resource whose type is rdfs:Class. Every resource belongs to the class rdfs:Resource and all the classes used in RDF schema are subclasses of resources.

Properties have two important properties themselves. One is the domain of a property. The domain of a property refers to the zero or more classes of resources to which the property may be applied. For example, the property "sex" may be applied to man and beast and nouns in many languages. If a property has no explicit domain, it may be applied to any resource. A second important property of a resource is the range of values that it can take. The range constrains the values that the property can take. If the property has no range, it can take any value. If the property has one or more range properties, each of the values of these range properties must be of type rdfs:Class. In that case the classes indicate the types of objects that the value of the property can take.[13]

The follow topics can be elaborated on this section:

- Why do we need a schema for RDF?
  - Data integrity
  - Validating statements
- Defining the schema
  - Checking the object of a statement
  - Checking the predicate of a statement
- Hierarchy of types
  - Resources with multiple types
  - Classes
- Using the schema on the schema
- Other elements
  - Resources
  - Schema control
  - Constraints
  - RDF elements
  - Container elements
  - Typing in RDF and RDF schema
- New layers on RDF schema

OWL (Web Ontology Language)

[13] Krichel, Thomas. 2004. The Semantic Web and an Introduction to Resource Description Framework. Edited by Michael Koenig. Knowledge Management: lessons learned. Information Today, Inc.

OWL is a *Web* Ontology language. Where earlier languages have been used to develop tools and ontologies for specific user communities (particularly in the sciences and in company-specific e-commerce applications), they were not defined to be compatible with the architecture of the World Wide Web in general, and the Semantic Web in particular.

OWL uses both URIs for naming and the description framework for the Web provided by RDF to add the following capabilities to ontologies:

- Ability to be distributed across many systems
- Scalability to Web needs
- Compatibility with Web standards for accessibility and internationalization
- Openess and extensiblility

OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. [14]

- OWL Lite – a lightweight implementation that focuses on classification hierarchy and basic and basic constraints for ontology construction. The expectation is that OWL Lite will provide an easier reference model for tools and a quicker migration path for content owners who wish to convert taxonomy and simple thesauri into the OWL Lite format.
- OWL DL – the core OWL implementation that provides maximum expressiveness while still providing computational completeness and decidability. The DL part of OWL refers to its usage of the description logic paradigm for ensuring that computational completeness.
- OWL Full – a layer that enables modelers to inject ontological assertions into RDF documents without being restricted by syntactic constraints, which guarantee computational completeness. This is the most expressive part of OWL but it makes no guarantees about the decidability of the resulting ontology.

The following topics can be further discussed in this section:

- Introduction
  - The Species of OWL
  - Structure of the Document
- The Structure of Ontologies
  - Namespaces
  - Ontology Headers
  - Data Aggregation and Privacy
- Basic Elements
  - Simple Classes and Individuals
  - Simple Properties
  - Property Characteristics
  - Property Restrictions

[14] http://www.w3.org/2004/OWL/ [retrieved on 5/13/2006]

- Ontology Mapping
  - Equivalence between Classes and Properties
  - Identity between Individuals
  - Different Individuals
- Complex Classes
  - Set Operators
  - Enumerated Classes
  - Disjoint Classes

3.4 Ontology Modeling

This section should discuss entity-relationship model and its relationship with topic maps and RDF/OWL. One way to show the power of ontology modeling is via implementation of complex relationships that exist among entity types or classes. In other words, a powerful function of ontology-driven system is in its ability to implement all the relationships that are expressed in ER diagrams. The current relational system reduces all the rich relationships that exist between entities into 1:1, 1:N, and M:N. This is quite a loss. This section should clearly demonstrate the difference between relational implementation vs. ontology implementation.

3.5 Ontology Development

Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED (Price and Spackman 2000) and the semantic network of the Unified Medical Language System (Humphreys and Lindberg 1993). Broad general-purpose ontologies are emerging as well. For example, the United Nations Development Program and Dun & Bradstreet combined their efforts to develop the UNSPSC ontology which provides terminology for products and services (www.unspsc.org). An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. The following provides an outline that needs to be discussed to help students learn how to proceed with ontology development.

- What is ontology?
- A simple knowledge-engineering method
  - Determine the domain and scope of the ontology
  - Consider reusing existing ontologies
  - Enumerate important terms in the ontology
  - Define the classes and the class hierarchy
  - Define the properties of classes – slots
  - Define the facets of the slots
  - Create instances
- Defining classes and a class hierarchy
  - Ensuring that the class hierarchy is correct
  - Analyzing siblings in a class hierarchy
  - Multiple inheritance
  - When to introduce an new class (or not)
  - A new class or a property value?
  - An instance or a class?
  - Limiting the scope
  - Disjoint subclasses
- Defining properties – more details
  - Inverse slots
  - Default values

- What is in a name?
  - Capitalization and delimiters
  - Singular or plural
  - Prefix and suffix conventions
  - Other naming considerations

3.6 Ontology Design Patterns

Ontology is not just one kind of thing; nor is it used in one kind of way. In fact, ontology is simply an enabler for system architects to apply core problem solving patterns in new and innovative ways.

Ontology Types and Usage

The number of possible ontology types is limited only by the imagination of the human mind. Therein lies its greatest strength as well as its greatest weaknesses. As a tool for modeling concepts, the many forms ontology can take give us windows into the true nature of what we are attempting to model in the first place. It is valuable to consider at least four main categories of ontology: interface, process, information, and policy.

- **Interface Ontology**: an ontology that models an application programming interface (API) of some kind. These interface ontologies are used to describe expected API behavior and results so that machine agents may "discover" how a given interface should be used and what to expect from it.
- **Process Ontology**: the category of process ontology is distinct because of the requirements to accurately model concepts related to time, cause, effect, events, and other nonphysical aspects of the real world. Process ontology has innumerable applications in software that span from providing event services for single applications, to work flow utilities for users, to orchestrating complex machine-to-machine interactions among multiple businesses in various industry segments.
- **Policy Ontology**: the policy ontology is intended to represent the systems, functions, and components deployed inside a large enterprise. The purpose of this type of ontology is not for interoperability, but for visibility and monitoring.
- **Information Ontology**: is a specification of a set of concepts for a given scope – usually some sort of business or technical domain. Ontology is used to identify concepts free of implementation details that could obscure important truths about a specific domain or area of ontology. Within the sphere of information ontology, there are at least six distinct categories of ontology.
  - **Industry Ontology**: used to capture concepts and knowledge useful to a particular industry domain such as manufacturing, life sciences, electronics, and logistics. (HL7 Health Care Ontology)
  - **Social Organizational Ontology**: used to represent humans and the human organizations they establish. This kind of model captures the structure and goals of an organization in terms of companies, divisions, groups, roles, locations, facilities, objectives, strategies, and metrics of the enterprise alongside a model of human communications, human skills and competencies, cultural dispositions, and other social concerns. (SKwyRL Social Architectures Project)
  - **Metadata Ontology**: used to describe unstructured published content, which resides either online or in print. (Dublin Core RDF Schema)
  - **Common Sense Ontology**: used to capture general facts about the world like "water is wet." These ontologies often have a unique notation and are considered valid across multiple domain spaces. (OpenCyc Upper Ontology)

- **Representational Ontology**: used to define information about information; sort of a metaontology to describe the relationships, associations, and containment among data elements. (OMG's UML Meta-Object Facility)
- **Task Ontology**: used to associate terms with tasks within fairly narrow scopes, for instance, task description for computer-based training programs. The notion of task ontology can be closely associated with both process and interface ontologies but comes at the problem from a data centric angle. (Computer-Based Training Task Ontology).[15]

Spectrum of Ontology Representation Fidelity

Content ontologies can be represented in a variety of ways, ranging from informal to very formal; this results in a spectrum of data representation sophistication. The following topics can be discussed in this section and differences among them should be stressed:

- Controlled Vocabulary
- Taxonomy
- Relational Schema
- Object-Oriented Models
- Knowledge Representation Languages

3.7 Developing Ontology Using Ontopoly (Topic Map Editor)

The ontoploy, a topic map editor developed by Ontopia, is an excellent tool to learn how to design an ontology using topic maps. It can be downloaded at the following site for free: http://www.ontopia.net/download/freedownload.html. *Ontopoly* provides a good interface for the subject matter experts who design the ontology's underlying model. The knowledge base is then populated through a combination of automated processing and human intellectual effort. Using the Ontopia Knowledge Suite, integrators can aggregate content from databases, documents and web feeds. Additional, "tacit" knowledge and links to information resources (as required) can then be added manually using *Ontopoly*'s ontology-driven interface. Ontologies built using *Ontopoly* can be used to organize portals, structure Content Management Systems, drive auto classification of information, and underpin many other processes, thereby providing a foundation for total knowledge management and publishing solutions based on semantic technology.

The ontopoly editor is designed to support both ontology modeling and instance population. Students can see what they constructed immediately. Modeling, instance population, and viewing is effectively integrated. The following topics can be discussed in this section.

- General concepts
  - Application architecture
  - Configuring and populating topic maps
  - User interface conventions
  - Topic Map Index Page
- Using the Type Configuration Pages
  - Type Index Pages
  - Type Configuration Pages

[15] Pollock JT and Hodgson, R. 2004. Adaptive Information: Improving Business Through Semantic Interoperability, Grid Computing and Enterprise Integration. Wiley. Pp. 148-153.

    - Configuring topic types
    - Configuring occurrence types
    - Configuring association types
    - Configuring role types
    - Configuring name types
    - Topic Map Description Page
    - Exporting a topic map
    - Saving a topic map
- Using Instance Editing Pages
    - Topic Type Index Page
- Using Ontopoly with multiple users
    - Storing user data in a topic map

3.8 Developing Ontology Using Protégé (RDF/OWL editor)

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

An ontology describes the concepts and relationships that are important in a particular domain, providing a vocabulary for that domain as well as a computerized specification of the meaning of terms used in the vocabulary. Ontologies range from taxonomies and classifications, database schemas, to fully axiomatized theories. In recent years, ontologies have been adopted in many business and scientific communities as a way to share, reuse and process domain knowledge. Ontologies are now central to many applications such as scientific knowledge portals, information management and integration systems, electronic commerce, and semantic web services. The following topics can be discussed in this section:

- What are OWL ontologies?
    - The three species of OWL
    - Components of OWL ontologies
- Building an OWL ontology
    - Named classes
    - Disjoint classes
    - Using OWL wizard to create classes
    - OWL properties
    - Inverse properties
    - OWL property characteristics
    - Property domains and ranges
    - Describing and defining classes
    - Using a reasoner
    - Necessary and sufficient conditions
    - Automatic classification
    - Universal restriction
    - Cardinality restrictions
- Creating other OWL constructs
    - Creating individuals
    - hasVaue restrictions
    - Enumerated classes

- Other Topics
  - Namespaces and importing ontologies
  - Ontology test

4. Conclusion

A sound curriculum of "metadata and ontology" is important in library and information science discipline because they have the direct link to cataloging and classification. One way to look at metadata and ontology curriculum is that it inherits the strength and knowledge accumulated in cataloging and classification and takes them further to provide users with more inter-connected information and knowledge that people need in the 21 century.

This paper stresses the importance of assigning unique and persistent identifier to metadata elements, ontology classes, and resources. It introduced the best practices of XML namespaces so that students understand how to set up a namespace policy in an organization. Students are recommended to learn standard metadata schemas and well-known application profiles and to implement them using XML DTD and schema so students grasp the concepts of metadata interoperability in detail. After gaining a solid background in metadata, students can expand their knowledge to understand data modeling because future cataloging model (FRBR) is expressed using entity-relationship model. Ontology is presented as a way to implement all the relationships that are important in the real world. FRBR model can be implemented much more flexibly if it uses ontology technology. Ontology-driven implementation is a significant progress compared to relational system implementation. In conclusion, students will have grasp of the benefits of ontology-based systems in terms of providing seamless knowledge to users.