Ontology Library Systems: The key to successful Ontology Re-use

Ying Ding & Dieter Fensel
Division of Mathematics and Computer Science
Vrije Universiteit, Amsterdam, the Netherlands
www.cs.vu.nl/~ying,~dieter

Abstract

Increasingly, effort has been devoted to surveying ontology-related research studies from various aspects. However, no survey is available for the ontology library system. For this reason, we decided to examine existing library systems in this paper. First, we identified the main criteria (management, adaptation, and standardization) for evaluating the functionality of the library systems. Then, based on the further enriched criteria, we surveyed most existing ontology library systems. Finally, we summarized the comparison and proposed various important requirements for structuring ontology library systems. The ontology library systems surveyed include: WebOnto, Ontolingua, DAML Ontology Library System, SHOE, Ontology Server, IEEE Standard Upper Ontology, OntoServer and ONIONS.

1. Introduction

With the rapid development of the World Wide Web, the amount of available information online has increased exponentially. A lack of standardization and common vocabulary has continued to generate heterogeneity, which strongly hinders information exchange and communication. *Ontologies*, which capture the semantics of information from various sources and giving them a concise, uniform and declarative description, have gained significance due to the demands in academia and industry [1]. As the number of different ontologies is on the increase, the task of maintaining and re-organizing them in order to facilitate the re-use of knowledge is challenging. A breakthrough in ontology technology would require methodological aids and tools that enable effective and efficient development. A key aspect in achieving this is successful re-use of ontologies. Being developed for supporting knowledge sharing and reuse, it is the lack of proper support of ontology re-use that hampers a broader dissemination of the ontology. *Ontology library systems* are an important tool in grouping and re-organizing ontologies for further re-use, integration, maintenance, mapping and versioning.

An *Ontology library system* is a library system that offers various functions for managing, adapting and standardizing groups of ontologies. It should fulfill the needs for re-use of ontologies. In this sense, an ontology library system should be easily accessible and offer efficient support for re-using existing relevant ontologies and standardizing them based on upper-level ontologies and ontology representation languages. For this reason, an ontology library system will, at the very least, feature a functional infrastructure to store and maintain ontologies, an uncomplicated adapting environment for editing, searching and reasoning ontologies, and strong standardization support by providing upper-level ontologies and standard ontology representation languages.

Recently, increasing effort has been devoted to surveying ontology-related research studies from various aspects, including that of ontology representation languages [2], ontology development [3], and ontology learning approaches [4]. However, no survey has been made of ontology library systems. This prompted us to examine the existing library systems in this paper. We will identify the main criteria for evaluating their functionality. We will also carefully evaluate existing proposals according to these requirements. In order to facilitate ontology re-use, a library system must, at the very least, support the following: (see Figure 1):

- ontology re-use by open *storage*, *identification* and *versioning*.
- ontology re-use by providing smooth *access* to existing ontologies and by providing advanced support in *adapting* ontologies to certain domain and task-specific circumstances (instead of requiring such ontologies to be developed from scratch).
- ontology re-use by fully employing the power of *standardization*. Providing access to *upper-layer ontologies* and standard *representation languages* is one of the keys to developing knowledge sharing and re-use to its full potential.

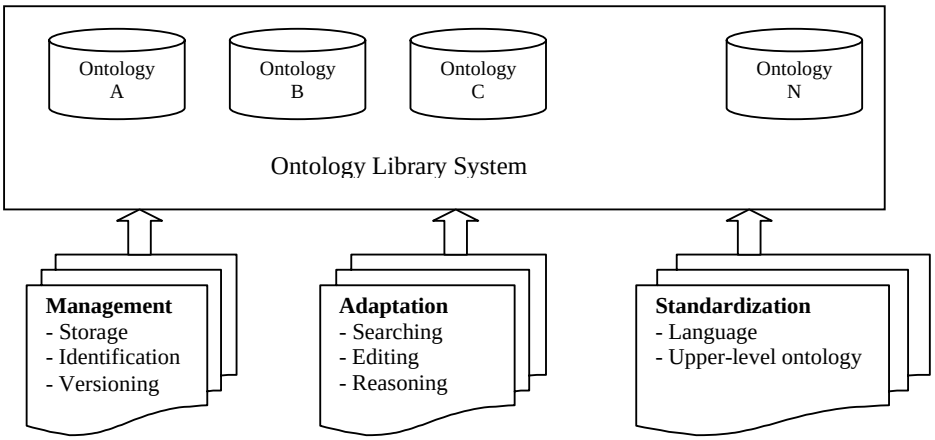

Figure 1. General overview of the structure of the survey

The aspects above can be further specified as the follows:

Management. The most important function of an ontology library system is that of facilitating the re-use of knowledge (ontologies). After all, the main purpose of ontologies is to enable knowledge sharing and re-use [5]. Important aspects of the re-use functionality of an ontology library system are open storage, identification, and versioning support.

- **Storage** (how to store the ontology): (a) Is the ontology easily accessible (via a client/server architecture, Peer-to-Peer, etc.) in supporting remote access and editing?; (b) Are ontologies classified according to some existing or homemade categories? (Classifying ontologies is an important step in reorganizing them such that users can easily search and identify relevant ontologies. It emphasizes the library system function of reorganizing ontologies); and (c) Are ontologies stored in modules? (The modularity structure of an ontology library system can facilitate the process of re-use, mapping and integration; it guarantees proficient ontology re-use).
- **Identification** (how to uniquely identify an ontology): Each ontology must have a unique identifier in the ontology library system.
- **Versioning** (how to maintain the changes of ontologies in an ontology library system): Versioning is very critical in ensuring the consistency among different versions of ontologies.

Adaptation. Ontology library systems should make facilitate the task of extending and updating ontologies. They should provide user-friendly environments for searching, editing and reasoning ontologies. Important aspects in an ontology library system include support in finding and modifying existing ontologies.

- **Searching** (how to search ontology from the ontology library system): Does a library system provide certain searching facilities, such as keyword-based searching or other advanced searching? Does it feature an adequate browsing function?
- **Editing** (how to add, delete and edit specific ontologies in the ontology library system. One of the most important features that an ontology library system should have is one that modifies stored ontologies or adds new ontologies): How does the system support the editing function? Does it support remote and cooperative editing?
- **Reasoning** (how to derive consequences from an ontology): How does the system support ontology evaluation and verification? Does it make it possible to derive any query-answering behavior?

Standardization. Ontology library systems should follow existing or available standards, such as standardized ontology representation languages and standardized taxonomies or structures of ontologies.

- **Language** (the kind of standard ontology language used in the ontology library system, for instance, RDFs[1], XMLs[2] or DAML+OIL[3]): Does the system only support one standard language or other different languages?
- **Upper-level ontologies** (Is the ontology library system 'grounded' in any existing upper-level ontologies, such as Upper Cyc Ontology, SENSUS, MikroKosmos, the PENNMAN Upper Model, and IEEE upper-layer ontology?): The upper-level ontology captures and models the basic concepts and knowledge that could be re-used in creating new ontologies and in organizing ontology libraries.

This survey report is structured as the follows. We will begin by examining current ontology library systems in light of the aspects outlined above. Next, we will provide a summary of our comparison of these systems. Finally, we will discuss various important requirements for structuring ontology library systems.

2. State-of-the-art Survey

This section surveys current important ontology library systems. These include: WebOnto[4] (Knowledge Media Institute, Open University, UK), Ontolingua [5] (Knowledge Systems Laboratory, Stanford University, USA), DAML Ontology library system[6] (DAML, DAPAR, USA), SHOE[7] (University of Maryland, USA), Ontology Server[8] (Vrije Universiteit, Brussels, Belgium), IEEE Standard Upper Ontology[9] (IEEE), OntoServer[10] (AIFB, University of Karlshruhe, Germany) and ONIONS[11] (Biomedical Technologies

[1] http://www.w3.org/RDF/
[2] http://www.w3.org/XML/
[3] http://www.ontoknowledge.org/oil/oilhome.shtml
[4] http://eldora.open.ac.uk:3000/webonto
[5] http://www-ksl-svc.stanford.edu:5915/
[6] http://www.daml.org/ontologies/
[7] http://www.cs.umd.edu/projects/plus/SHOE/
[8] http://www.starlab.vub.ac.be/research/dogma/OntologyServer.htm
[9] http://suo.ieee.org/refs.html
[10] http://ontoserver.aifb.uni-karlsruhe.de/
[11] http://saussure.irmkant.rm.cnr.it/onto/

Institute (ITBM) of the Italian National Research Council (CNR), Italy). ONIONS is a methodology for ontology integration and was successfully implemented in several medical ontology library systems. Strictly speaking, it is not an ontology library system. However, since it defines various criteria for developing such a system, we will examine it briefly here. There are many more ontology library systems than we included in our comparison. We have only included approaches that are publicly available as those offer enough detailed information to enable us to evaluate their actual functionalities.

2. 1 WebOnto

WebOnto is an ontology library system developed by the Knowledge Media Institute of the Open University (UK) [6]. It is designed to support the collaborative creating, browsing and editing of ontologies. It provides a direct manipulation interface displaying ontological expressions and also an ontology discussion tool called Tadzebao, which could support both asynchronous and synchronous discussions on ontologies ([7], see Figure 2).

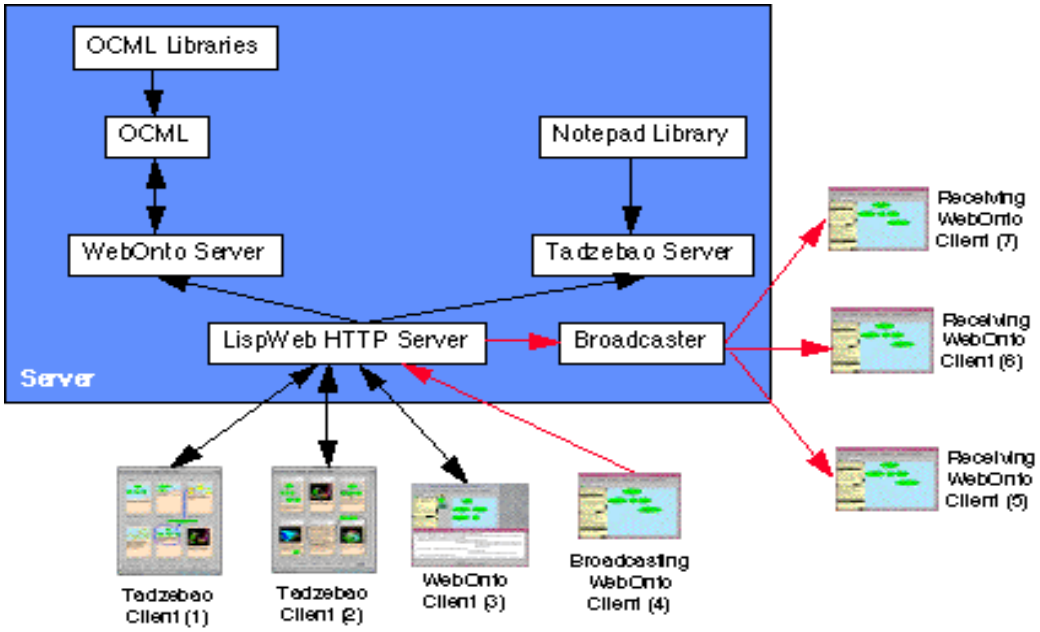

Figure 2. The architecture of the Tadzebao and WebOnto Server [6].

2.1.1 Management

Storage. WebOnto relies on a client/server-based *architecture*. The servers are responsible for storing and maintaining ontologies and user dialogues, the clients are the interfaces to access the stored ontologies. Ontologies stored in the WebOnto are not *classified* according to some existing categories. Ontologies are divided into small units (for instance, ontology is the tree-structural of classes, and the small unit is the class and its parents). They are then stored in a specific *Module* containing name, type, and the names of class parents. This system can draw graphical representations of ontologies based on the modularity storage.
Identification. Ontologies stored in the WebOnto library system are identified by their unique names. Even though an ontology is divided into small units, each unit contains the name, type, and the names of the class parents.
Versioning. WebOnto only mentioned that ontologies can be inherited from ancestor ontologies. No actual versioning support is provided.

2.1.2 Adaptation

Searching. Ontologies are graphically displayed. They can only be browsed by using browsing commands, such as viewing a new ontology or inspecting its structure. No direct query interface is provided.
Editing. TaDzeBao is designed for discussing and editing ontologies (see Figure 2). It supports both asynchronous and synchronous discussions and editing on ontologies. The Tadzebao server is responsible for maintaining the ontologies, and delivering ontologies to requesting Tadzebao clients. The client is responsible for presenting a consistent view of the selected ontologies.
Reasoning. Ontologies in WebOnto are represented in OCML, which supports rule-based reasoning.

2.1.3 Standardization

Language. Ontologies (classes, instances, functions, procedures, or rules) are represented in OCML only [6]. In other words, no standard representation languages for ontologies are supported.
Upper-level ontologies. WebOnto does not include a 'giant' standard upper-level ontology but has a more fine-grained structure ([8] and [9]). At the top there is the base ontology describing the meta-model of OCML (things such as relations, functions, procedures, classes, instances, slots, etc., similar to SHOE). Below are the imported (with a few modifications) 'simple time' ontology from the Ontolingua library and ontologies describing organizations, technologies, events and basic common concepts. Figure 3 shows the typical ontology inclusion hierarchy in the WebOnto ontology library system.

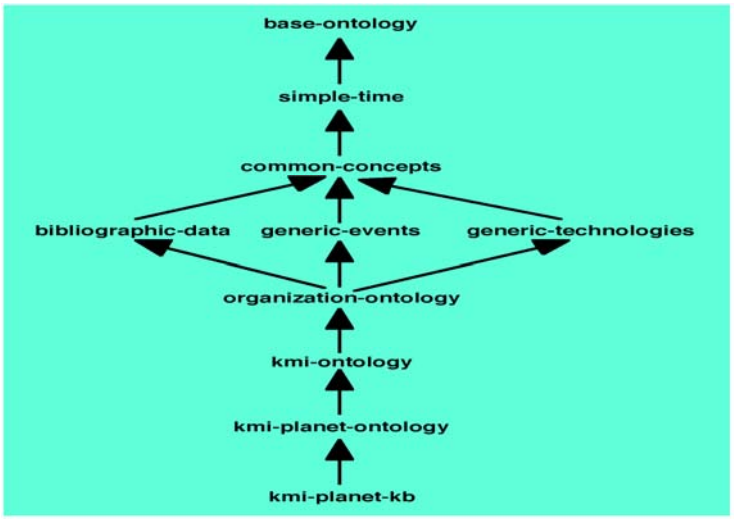

Figure 3. Structure of upper-level ontology in WebOnto

2.1.4 Summary

The WebOnto's ontology library system is client/server and graphically based. It stores an ontology as a module with a unique name for identification. It supports asynchronous and synchronous ontology editing. Ontology searching is limited to ontology navigating or browsing (but graphical-based). The ontology is represented by OCML, which can support rule-based reasoning. It does not have any ontology versioning function or strong support in respect to ontology standardization issues.

2.2 Ontolingua

Ontolingua was developed in the early nineties at the Knowledge Systems Laboratory of Stanford University (see Figure 4, [10]). It consists of a server and a representation language. The server provides a repository of ontologies (ontology library system) to assist users in generating new ontologies and amending the existing ontologies collaboratively. The ontology stored at the server can be converted into different formats [11].

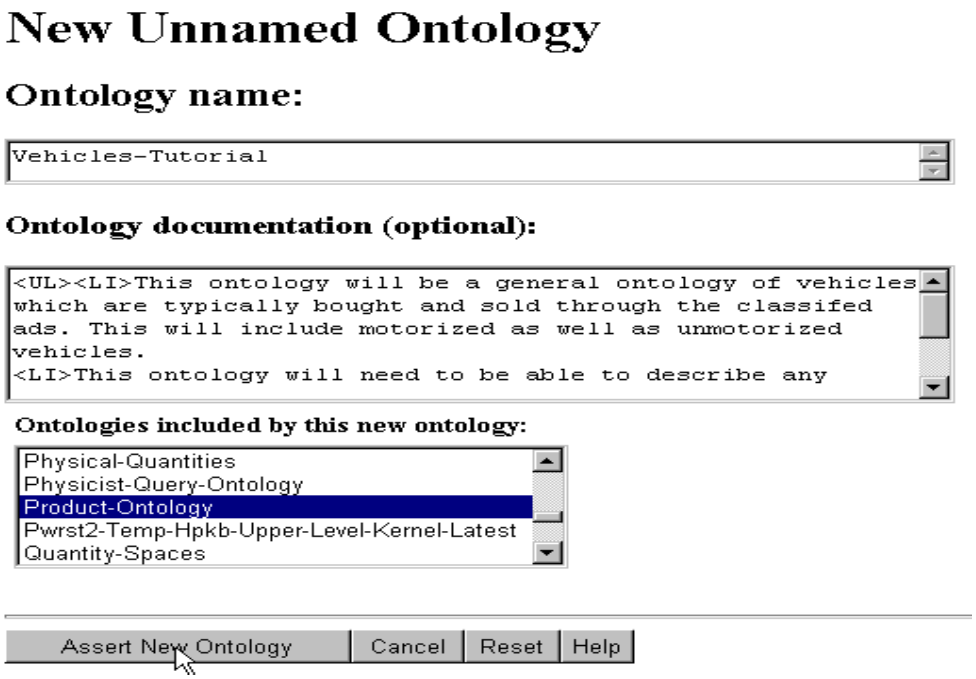

Figure 4. Screenshot of part of Ontolingua system (how to create an ontology)

2.2.1 Management

Storage. Ontolingua is client/server-based. It provides a distributed server *architecture* for ontology construction, use and re-use. Access to the contents of ontologies is provided via a network API and access to information derived from the contents by a general-purpose reasoner. The ontology server works like a database server and can enable distributed ontology repositories for editing, browsing, etc. The ontology server of Ontolingua supports a suite of other services, including configuration management for ontologies, support for ontologies that have components resident on remote servers, and support for an *Ontology-URL* that enables ontologies to be linked to the World Wide Web [12]. Ontologies stored in Ontolingua are not *classified* according to some existing categories. Ontology re-use in Ontolingua is supported by a *modular* structured library based on the following functions: inclusion, polymorphic, refinement, and restriction. Ontologies in this ontology library system are organized based on the lattice theory. Each ontology defines a set of formal terms. Ontologies can include (import from) other ontologies. Terms contained in an ontology are in the namespace of the ontologies that include it. In the lattice, an ontology includes the ontologies under which it is indented (see Figure 5). It applies the minimization and amortization principles enabling ontology writers to re-use existing ontologies in a flexible, powerful way [10].

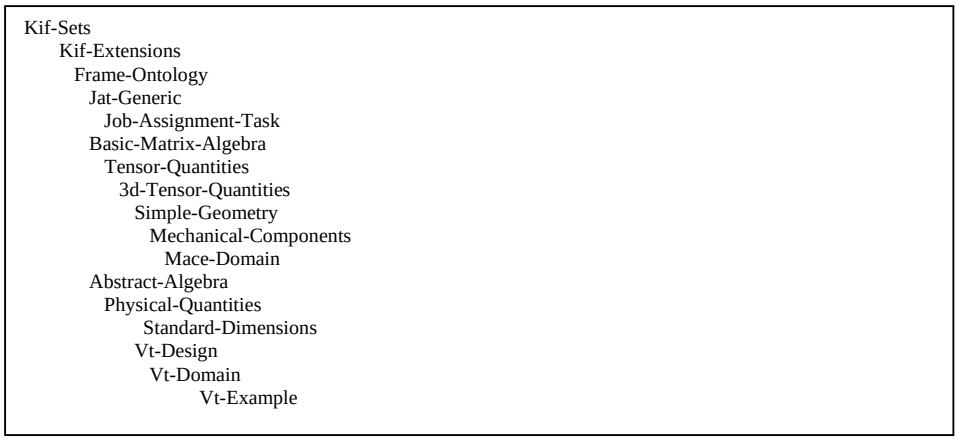

Figure 5. Part of the lattice of ontologies in the ontology library system of Ontolingua

The naming policy for Ontolingua is a good example of how the re-use of knowledge can be facilitated. The ontology in which a symbol is defined is called the symbol's home ontology. For example, if the symbol *S* is defined in ontology *A* as well as in ontology *B*, then from the perspective of ontology *A*, the input text "*S*" is interpreted as "the symbol named *S* defined in ontology *A*". From the perspective of ontology *B*, however, the input text "*S*" is interpreted as "the symbol named *S* defined in ontology *B*". The Ontolingua server input/output system includes a symbol-renaming feature that allows users to assign a name to a symbol, which is local to the perspective of a given ontology. This feature enables ontology developers to refer to symbols from other ontologies using names that are appropriate to a given ontology. It also enables them to determine how to resolve naming conflicts among symbols from multiple ontologies.

Identification. Each ontology has a name that uniquely distinguishes itself from any other ontology.

Versioning. The Ontolingua server does not feature any versioning functions.

2.2.2 Adaptation

Searching. Ontolingua features graphical ontology *browsing* and supports swift jumps from one term in the ontology to others term using hyperlinks. Ontolingua's class/subclass browsers can display an entire hierarchy in compact fashion, offering users a swift overview of an ontology. One particularly difficult task for ontology library systems is that of supporting efficient query answering from ontologies represented in a highly expressive language. Ontolingua develops an **idiom-based retrieval feature** that returns instances of a sentence containing diagrammatic variables from a given ontology. The retrieval feature employs a general purpose reasoner (i.e., theorem prover) and classifier that can be run as a background process to infer and cache sentences that match idioms used by the API and by translators. The general purpose reasoner developed for the Ontolingua representation language can provide basic reasoning support for ontology services including classification, deriving and catching instances of sentence diagrams to support idiom-based ontology access, ontology testing, and client-side execution [12]. Ontolingua also provides several tools that allow users to search for terms within ontologies in the library. A user may choose to use wild cards in searching the entire library for terms whose name matches the specified pattern. Context-sensitive searching is also available when the user needs to fill in the name of a term, by for instance, adding a value to a slot. Constraints are used to limit searches in context-sensitive searching. Finally, Ontilingua provides a *Reference ontology* that serves as an index of the ontology repository for class-based retrieval. Users can browse the reference ontology looking for classes of interest in the repository.

Editing. Ontolingua features four basic types of pages for the simple interface: the table of contents for the ontology library system, ontology summary pages, frame pages (for classes, relations or instances), and the class browser. Remote distributed groups can use their web browsers to build, and maintain ontologies stored at the server. However, this work cannot be carried out at the same time. Ontolingua supports vocabulary translation, which enables ontology builders to specify translation rules declaratively between the vocabulary used in a source ontology and the vocabulary used in a target ontology [12]. Ontolingua allows users to undo or redo any number of modifications made to the ontology since it was last saved.

Reasoning. Ontolingua enables developers to use an ontology to describe familiar situations and to query those situations to determine whether those situations have expected properties. It also includes a feature for specifying a test suite for an ontology, in which each test consists of a situation specification, a set of queries about the situation, and the answers expected to the queries.

2.2.3 Standardization

Language. The ontologies are stored primarily in KIF. KIF is a monotonic first order logic with a simple syntax and some minor extensions to support reasoning about relations. This language provides explicit support for building ontological modules that can be assembled, extended, and refined in a new ontology [1]. KIF is widely used among researchers in the United States .

Upper-level ontology. The public version of CYC upper-level ontology, called HPKB-UPPER-LEVEL with extended material drawn from Pangloss, WordNet, and Penma, is available on the Ontolingua server [13]. It contains approximately 3000 concepts, English definitions, and a few basic relationships between them. This upper-level ontology aims to maximize re-usability, enabling a greater degree of interoperation among knowledge-based systems by trying to account for all features associated with one event.

2.2.4 Summary

Ontolingua's ontology library system is client/server-based. It offers several options for re-using ontology: modular structure storage, lattice of ontology, naming policy, and a reference ontology (upper-level taxonomy). It supports collaborative ontology editing. Users can access the ontology library system via the Web. It also includes some relatively advanced searching features (wild-card and context intensive searching). In addition, Ontolingua supports ontological language translating, ontology testing and ontology integrating. HPKB-UPPER-LEVEL on the Ontolingua server maximizes re-usability and enables a greater degree of interoperation among knowledge-based systems.

2.3 DAML Ontology library system

The DAML ontology library system is part of the DARPA Agent Markup Language (DAML) Program, which officially started in August 2000. The goal of the DAML effort is to develop a language and tools to facilitate the concept of the Semantic Web. The ontology library system contains a catalogue of ontologies developed using DAML (Figure 6)[12]. This catalogue of DAML ontologies is available in XML, HTML, and DAML formats. People can submit new ontologies via the public DAML ontology library system.

[12] http://www.cs.man.ac.uk/~Ehorrocks/DAML-OIL/

```
<ontology uri="http://www.davincinetbook.com:8080/daml/rdf/personal-info.daml" id="2">
   <description>DAML ontology for homework 1</description>
   <poc name="Mark Neighbors" organization="Booz-Allen &amp;amp; Hamilton" email="neighbors_mark@bah.com"/>
   <submitter name="Mark Neighbors" organization="Booz-Allen &amp;amp; Hamilton" email="neighbors_mark@bah.com"
        date="2000-10-31"/>
   <keyword>personal information</keyword>
   <dmoz>http://dmoz.org/dmoz5</dmoz>
   <dmoz>http://dmoz.org/dmoz6</dmoz>
   <funder>DARPA DAML Program</funder>
   <class>AnnotatedBulletList</class>
     <class>Bullet</class>
   <class>BulletList</class>
   <class>Company</class> ……
…….
     <property>bulletList</property>
   <property>companies</property>
   <property>currentEmployerIDs</property>
   <property>currentProjectIDs</property>
   <property>description</property>
…….
<namespace>http://156.80.108.115/2000/10/daml-ont.daml</namespace>
   <namespace>http://www.w3.org/1999/02/22-rdf-syntax-ns</namespace>
   <namespace>http://www.w3.org/2000/01/rdf-schema</namespace>
  </ontology>
```

Figure 6. Example of an ontology in the DAML ontology library

2.3.1 Management

Storage. The DAML ontology library system is client/server-based. The structure of the stored ontology in this library system includes: ontology uri: ontology id; description; keyword; poc (point of contract): name, organization, email; submitter: name, organization, email; dmoz (open directory category); funder; classes (class names); properties (properties names); and namespaces (for example, Figure 6). Ontologies are *classified* according to Open Directory Category (www.dmoz.org), which includes arts, business, computers, games, health, home, kids and teens, news, recreation, reference, regional, science, shopping, society, sports, and world. This library also provides a summary of submitted ontologies, sorted by URI, Submission Date, Keyword, Open Directory Category, Class, Property, Funding Source, and Submitting Organization. The DAML ontology library system is still at its early age. It only provides a simple environment for people to submit and browse ontologies in the library. No *module* storage is considered at this moment.
Identification. Ontologies are identified by the URIs and identifiers.
Versioning. No versioning functions are provided.

2.3.2 Adaptation

Searching. This ontology library system offers no specific searching features. It contains only a catalogue of ontologies in three formats: XML, HTML and DAML. The HTML version can help users search by generated indexing of ontologies on URI, submission date, keyword, open directory category, class, property, namespace used, funding source, and submitting organization. The other two formats support simple browsing only.
Editing. No specific editing functions are available.
Reasoning. At present, this ontology library system does not support any reasoning functions[13].

[13] Developed by University of Manchester (UK), the FaCT (Fast Classification of Terminologies, http://www.cs.man.ac.uk/~horrocks/FaCT/) reasoner can be integrated into the DAML ontology library system and will give it some reasoning supports.

2.3.3 Standardization

Language: The DAML ontology library is very preliminary at this moment. It aims to push and popularize the standard ontology language. It is, therefore, not surprising that the library system supports language standards for the semantic web, i.e. RDF, RDF Schema and DAML-ONT (now is DAML+OIL).
Upper-level ontology. At present, there is no upper-level ontology for the DAML ontology library system.

2.3.4 Summary

The DAML ontology library system is just in its beginning stages. Naturally, the objective is to offer various functions for ontology library system management. So far, however, we have not been able to find any publicly available literature with detailed information on the technology. Even at this early stage, this system is very strong in its support for web-based ontology languages.

2.4 SHOE (University of Maryland, USA)

SHOE (Simple HTML Ontology Extensions) was developed by the University of Maryland (USA) ([14], see Figure 7). SHOE is also the first web-semantics language developed as a markup, and has been used for various applications, including for food safety for the US Food and Drug Administration and a military logistics planning system.

2.4.1 Management

Storage. SHOE's ontology library system contains lists of ontologies. These ontologies are indexed alphabetically. They are also *classified* based on the ontology dependency with clear tree structure. The upper-level ontology (Base Ontology), for instance, forms the root of the tree. The generic ontologies (e.g. Dubline Core Ontology, General Ontology, Measurement Ontology) form the first branch of the tree. And the specific ontologies (e.g. Beer Ontology, Commerce Ontology, Personal Ontology) make up the leaves of the tree. Except for the upper-level ontology, each ontology is stored in the standard format, including ontology ID, version, description, contact, revision date, extended ontologies, renames, categories, relationships, constants, inferences, definitions, notes and change history.
Identification. The unique name becomes of the identifier of ontology.
Versioning. SHOE's versioning scheme is very essential in handling different types of revisions. It maintains each version of ontology as a separate web page and each instance must state the version to which it adheres. Data sources can, therefore, upgrade to the new ontology. To enter a revision in SHOE, the ontology designer copies the original ontology file, assigns it a new version number, then adds or removes elements accordingly. If the revision merely adds ontology elements, then it can be used to form perspectives that semantically subsume the original perspective. Therefore, it can specify that it is compatible with previous versions using the optional BACKWARD-COMPATBLE-WITH field in the <ONTOLOGY> tag. Agents and query systems that discover this ontology can also use it instead of any of the ontologies with which it is backwardly compatible to form an alternate perspective for any data source.

SHOE is currently the ONLY project focusing on the problem of maintaining consistency as the ontology evolves. It separates instances from ontologies so that ontologies can provide different perspectives on the same data. It identifies different types of ontology revisions that could significantly affect the reasoning with existing data

sources. For instance, revisions that add categories or relations have no effect, revisions that modify rules can change the answers to queries, and revisions that remove categories or relations may eliminate certain answers [14].

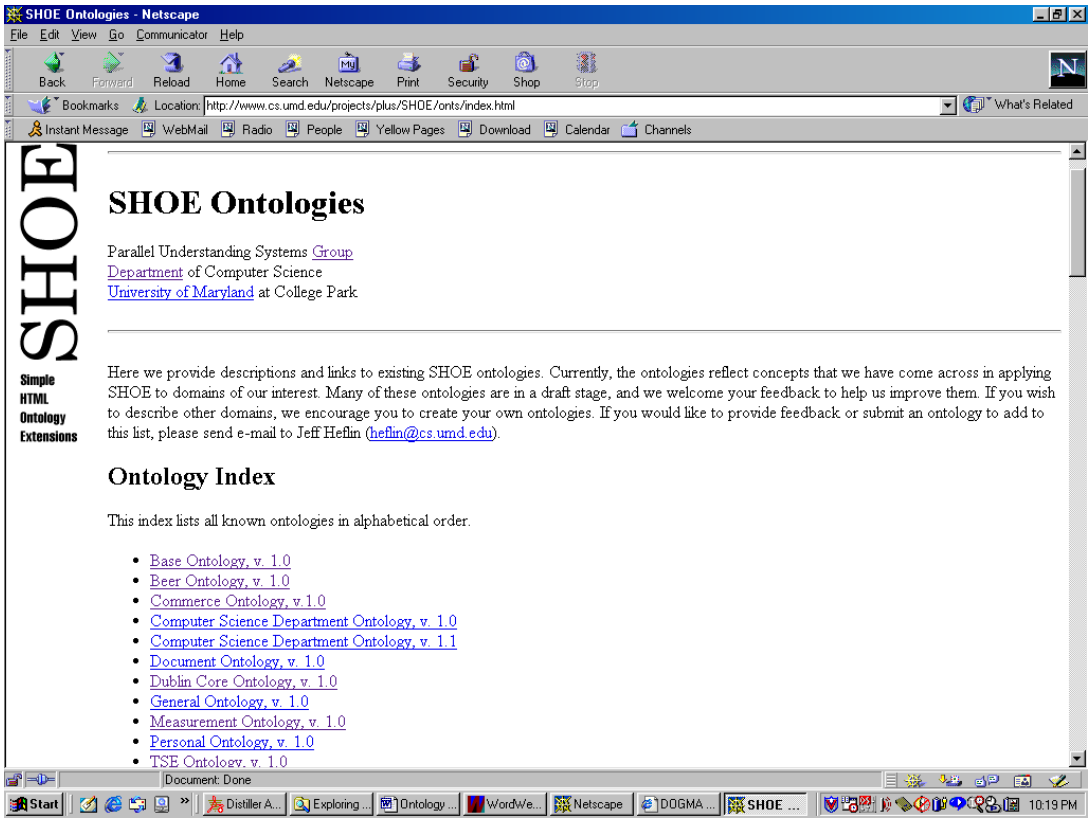

Figure 7. Screenshot of SHOE ontology library system

2.4.2 Adaptation

SHOE features no *Searching* and *Editing* environment. Users have to edit their ontology somewhere else and submit it to SHOE. The alphabetical indexing of ontologies enables users to browse and search SHOE ontologies.

Reasoning. Reasoning supports are provided to handle revision problems. For instance, a revision that adds or removes rules can provide an alternative perspective (p') for a legacy data source. This makes it possible to reason the subsumption relations of the alternative perspective (p') with the original perspective (p) ([14] and [15]).

2.4.3 Standardization

Language: Ontology is written in SHOE. SHOE is an HTML-based knowledge representation language. It is a superset of HTML, which adds the tags to semantic data. There are two categories for SHOE tags: tags for constructing ontologies and tags for annotating web documents. There is also an XML-based version of SHOE [15].

Upper-level ontology. Base ontology is the upper-level ontology for SHOE. It becomes the parent ontology for all SHOE ontologies on the web. All other SHOE ontologies extend the base ontology directly or indirectly. Base ontology declares the global data types (string, number, date and truth), ISA hierarchy (entity, SHOEEntity), and relationships (description and name). There is a one-to-one correspondence between a version of SHOE

and a version of the base ontology. Thus, the version of the Base ontology reflects the version of SHOE.

2.4.4 Summary

The SHOE ontology library system contains various ontologies (written in SHOE) with direct or indirect extensions of the upper-level ontology (Base ontology). SHOE flags itself with its versioning functions to solve inconsistencies caused by ontology evolution. SHOE itself is an extended HTML language with adding tags to represent ontologies and semantic data.

2.5 Ontology Server (Vrije Universiteit Brussels, Belgium)

Ontology Server, which was developed by the Vrije Universiteit in Brussels, links ontology engineering to database semantics. It deploys database techniques to manage and understand ontologies. The database management system (DBMS), equipped with various syntactical constructs, enables database diagrams to present objects, sub-type taxonomies, integrity constraints, derivation rules, etc. (see Figure 8).

2.5.1 Management

Storage. The ontology model consists of 5 basic elements: context, terms, concepts, roles and lexons. The ontology contains a set of contexts, which form the ontology itself. The ontology has a name (mandatory and unique in the ontology server), a contributor, an owner, a status ("under development", "finished") and documentation (an arbitrary string in which the contributor or owner can specify relevant information). The context is a grouping entity to group terms and lexons in the ontology. Every context within an ontology has its own unique name. A concept is an entity representing some “thing” and is identified by a unique ID. A term is an entity representing a lexical representation of a concept. Lexon is a grouping element with a triple structure containing a starting term, a role and a second term. Lexon always appears in a context and describes certain relations that are valid within that context. In this case, the lexons can be considered relations between concepts. The Database Management System is used to implement storage.
Identification. The unique name is the identifier for each ontology in this ontology server.
Versioning. The first prototype currently under construction does not take account of the version control. However, this will become a crucial issue in the next step.

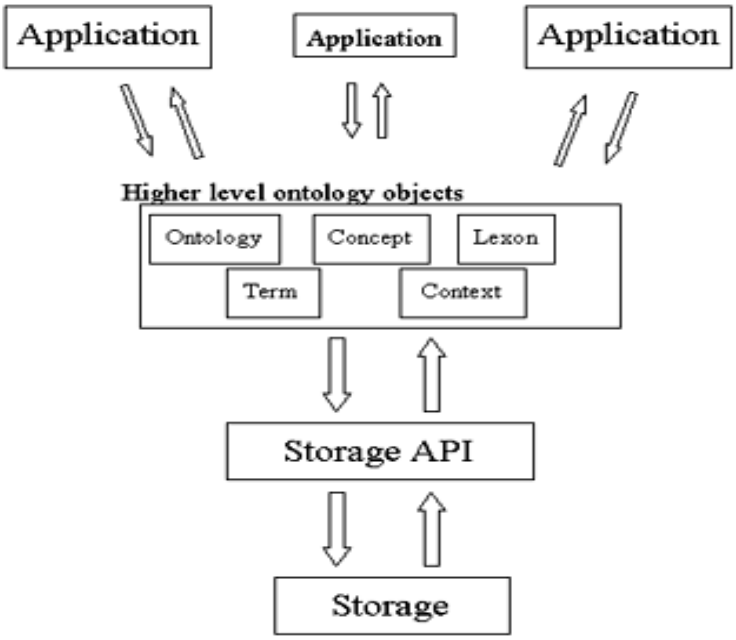

Figure 8. Ontology server architecture

2.5.2 Adaptation

Searching & **Editing.** Database API provides unified access to the basic structures of the ontology server. As the data in the ontology server, ontologies are managed by the DBMS (Database Management System). The API itself is specified as four different java interfaces. One of these is an interface to establish and close the connection to the database. The second interface features all of the basic functions required to add information to the ontology server (specific methods for adding ontology, context, terms, concept, lexons, users and versions are included). The third, the retrieval interface, features all of the basic functions to retrieve information from the ontology server. (The specific methods for this can be divided in two groups: (a) retrieving methods for detailed information about ontologies, contexts, terms, concepts, lexons, users and versions; and (b) retrieving methods for grouped information, such as those that retrieve all ontologies from the ontology server, all contexts from an ontology, all terms from a context, all lexons from a context and all users from the ontology server). And finally, the modification interface features all of the basic functions needed to modify information already present in the ontology server (it includes specific methods for modifying ontologies, contexts, terms, concepts, lexons, users and versions). In the future, an ontology manager will be developed to provide support for storing ontologies expressed in XML. WordNet will be added to the ontology server using the ontology manager. The ontology browser (currently under development) will assist users in accessing and browsing ontologies, contexts, terms, concepts, lexons, users and versions.

Reasoning. According to the documents available, no reasoning functions are specified.

2.5.3 Standardization

Language. The ontology object is expressed in XML.
Upper-level ontology. No upper-level ontology is adopted in this Ontology Server.

2.5.4 Summary

Ontology Server manages ontologies by using DBMS (Database Management System). It separates semantics from ontologies; thus, each ontology model contains 5 basic elements (context, terms, concepts, roles and lexons). Ontology can be accessed and searched through database API, including via the modification interface, the retrieval interface, and in the future, ontology manager, and ontology browser.

2.6 Others

This section discusses systems that are either not very standard in ontology library systems or are still in the very preliminary stages of development.

2.6.1 IEEE Standard Upper Ontology (IEEE)

The IEEE Standard Upper Ontology (SUO) Working Group has invested tremendous effort, working with a large number of ontologists, to create a standard top-level ontology to enable various applications, such as data interoperability, information search and retrieval, automated inferencing, and natural language processing. Their ontology library system is very simple and is accessible in its preliminary form on their website. It contains a group of classified ontologies, such as, ontologies in SUO-KIF, formal ontologies and linguistic ontologies/lexicons. Only the very basic hyperlinks of the ontologies are provided to help users jump to the home pages hosted by the ontologies (see Figure 9). There are no clear management, adaptation and standardization functions, such as those discussed in Section 1.

One special effort worth mentioning here is the first-ever merger of certain SUO sources into a single and coherent ontology, an ontology accessible via the website. This merger was achieved by combining David Whitten's structural ontology, John Sowa's upper ontology, Allen's temporal logic, Russell and Norvig's upper ontology, Casati and Varzi's formal theory of holes, Barry Smith's formal theory of boundaries, Borgo, Guarino, and Masolo's formal theory of physical objects, the Core Plan Representation, and the Agents and Numbers ontologies from the Ontolingua server.

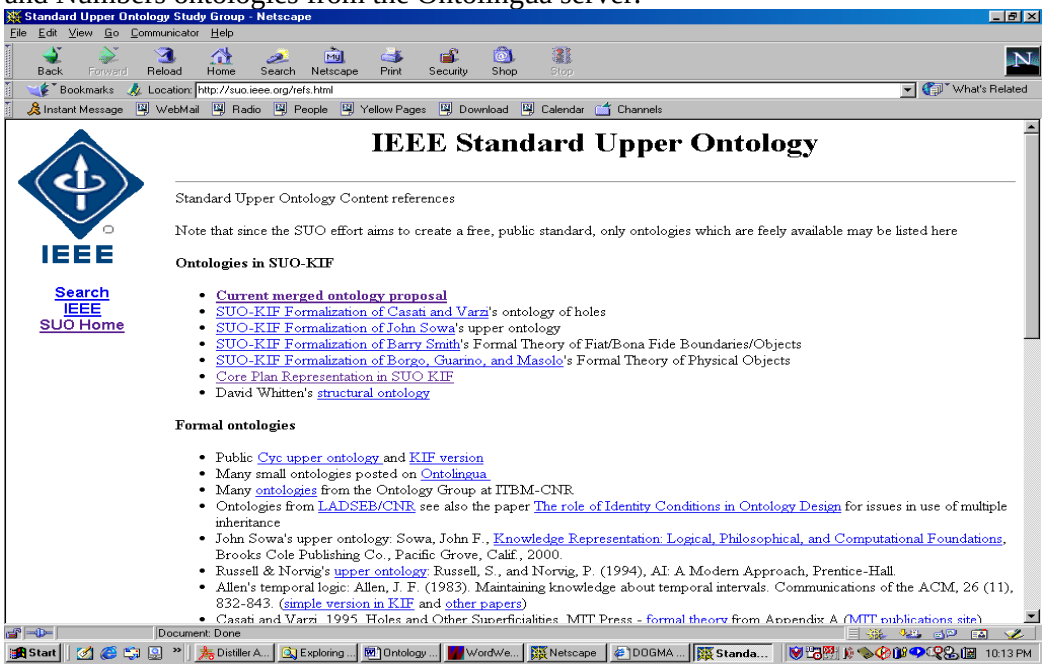

Figure 9. Screenshot of IEEE SUO

2.6.2 OntoServer (AIFB)

OntoServer, which is currently under construction, is an ontology server to support building, maintaining and using ontologies. It has a client/server-based architecture, which integrates various types of software or tools to form tool-based support for an ontology environment (see Figure 10).

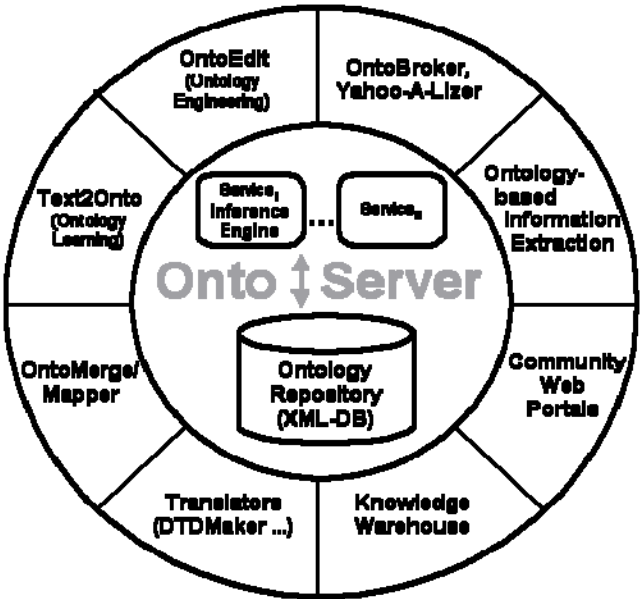

Figure 10. OntoServer infrastructure

OntoServer will integrate tools from ontology engineering (such as OntoEdit, Text-To-Onto, Ontology Merger/Mapper, Ontology Visualization and Translator) and ontology applications (such as OntoBroker, OntoWrapper, Ontology Visualization (Hyperbolic View), DTD-Maker, Intranet Management System and Semantic Community Web Portals). OntoServer is still in its very early stages; no detailed information is available from its homepage. Consequently, we can only sketch the infrastructure.

2.6.3 ONIONS

ONIONS *(ONtological Integration Of Naive Sources)* is a methodology for ontology integration (ontology mediation, alignment and unification). It was developed in the early 1990s to account for the problem of conceptual heterogeneity. ONIONS creates a *stratified* design of an ontology library system. It contains richly documented and formalized generic ontologies and a cognitively transparent top level. Moreover, intermediate *modules* contain the most general concepts of a domain, based on the generic ontologies and the top level. Domain ontologies are designed based on intermediate ontology ([16], see Figure 10).

ONIONS is committed to developing a large-scale ontology library system for medical terminology. This methodology employs a design based on logic description for the modules in the library and makes extended use of generic theories, thus creating a *stratification* of the modules. The current implementation of the methodology employs LOOM, a knowledge representation system that supports classification services based on the description logic. Ontologies are *classified* based on description logic. The ontology library system covers all local definitions and the paradigms used in building multi-local, integrated definitions. Further classification in this library is based on steps pertaining to the diffusion, use, classification, and validation of the models.

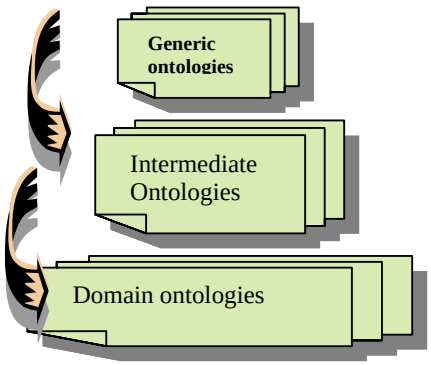

Figure 10. The stratified design of an ontology library system (ONIONS)

ONIONS is mainly an ontology integration methodology, which is implemented by many projects. It creates a *stratified* ontology library system including generic ontology, intermediate ontology and domain ontology.

3. Summary of the Survey

Research on ontology library systems is still a very new field. The following summary is based on our survey of the ontology library systems described above (see also Table 1).

3.1 Management

Storage. The ontology library systems in this survey fall into one of two categories: (a) those with a client/server-based *architecture* aimed at enabling remote accessing and collaborative editing (WebOnto, Ontolingua, DAML Ontology Library); and (b) those that feature web-accessible architecture (SHOE, IEEE SUO). Ontology Server features a database-structured architecture. Most ontologies in this survey of ontology library systems are classified or indexed. They are stored in a modular structured library (or lattice of ontologies). WebOnto, Ontolingua and ONIONS all highlight the importance of a modular structure in an ontology library system as that structure facilitates the task of reorganizing ontology library systems and re-using and managing ontologies.

Identification. The standard way to identify an ontology is by its Unique name or Identifier.

Versioning. Only SHOE supports versioning for handling the dynamic changes of ontologies. Versioning is an important aspect of the ontology library system. Although many of the systems surveyed do not currently have this function, they clearly show that it is needed for future improvements.

Table 1. The summary of the ontology library system survey

		WebOnto	Ontolingua	DAML library	SHOE	Ontology Server	Others (IEEE SUO, OntoServer, ONIONS)
Management	**Storage**	- client/server-based - no classification - modularity storage	- client/server-based - no classification - modular structured library (lattice of ontologies, naming policy)	- client/server-based - classification of ontology - no modularity storage	- web accessible - classification of ontology - tree structure of ontology dependency	- database access - no classification - modularity storage	- web access (IEEE SUO), client/server-based (OntoServer), - classification of ontology (IEEE SUO, ONIONS) - stratified design (ONIONS)
	Identification	- unique name - unique unit name	- unique name	- unique URI and Identifier	- unique name	- unique name	
	Versioning	- indirect: inherited from ancestor ontology	No versioning	No versioning	- versioning support for ontology revision	- no versioning	- no versioning
Adaptation	**Searching**	- graphical display - simple browsing	- simple browsing - idiom-based retrieval facility for simple query answering - wild-card searching - context sensitive searching - reference ontology as the index	- simple browsing	- simple browsing	- database API - DBMS - add, modify, retrieve - ontology manager - ontology browser	- simple browsing (IEEE SUO,
	Editing	TaDzeBao: - asynchronous and synchronous discussions and editing on ontologies,	- simple interfaces - collaborative ontology construction - vocabulary translation - undo/redo - hyperlinked environment	No specific editing functions	- no editing	- add, modify, retrieve	- no editing
	Reasoning	- rule-based reasoning	- use situation to determine the expected properties. - ontology testing	- no reasoning	- limited reasoning support for ontology revision	- no reasoning	- no reasoning
Standardization	**Language**	OCML	KIF - ontology language translation	RDF, RDFs, DAML+OIL	SHOE	XML	
	Upper-level Ontology	- no standard upper-level ontology - a more fine-grained structure: based ontology, simple-time, common concepts	- public version of CYC upper-level ontology (HPKB-UPPER-LEVEL)	No standard upper-level ontology	- Base Ontology	- no standard upper-level ontology	- IEEE SUO (upper-level ontology integration)

*3.2 **Adaptation***

Searching. Most of these ontology library systems can be accessed through the Internet or World Wide Web. They offer simple browsing only. Ontolingua is the only one that offers some functional searching features, such as keyword searching (wide-card searching), simple query answering, context sensitive searching, etc. As it is embedded in the database management system, Ontology Server could also provide SQL-based searching.
Editing. Most ontology library systems only provide simple editing functions. WebOnto and Ontolingua support collaborative ontology editing (asynchronous and synchronous).
Reasoning. Very simple reasoning functions are provided by WebOnto (rule-based reasoning), Ontolingua (ontology testing) and SHOE (ontology revision).

*3.3 **Standardization***

Language. These ontology library systems use different languages to store their ontologies. In this case, the important function for the future ontology library system should support inter-language translating (like Ontolingua) or some standard language should be accepted or proposed within the ontology community (such as DAML+OIL).
Upper-level Ontology. Ontolingua has a public version of CYC upper-level ontology called HPKB-UPPER-LEVEL with some modification drawn from Pangloss, WordNet, and Penma. WebOnto and SHOE doesn't have the standard upper-level ontology but has its own fine-grained structure (e.g., Base Ontology). IEEE SUO tries to set up a public standard upper-level ontology.

4. Conclusions: Ontology library system requirement

Now that we have surveyed the ontology library systems above, we will summarize important requirements for structuring an ontology library system to enhance ontology management, adaptation and standardization:

*4.1 **Management***

Storage. A client/server-based *architecture* is critical to an ontology library system's capacity to support collaborative ontology editing. An ontology library system should also be web accessible.
It is necessary to *classify* ontology in an ontology library system in order to facilitate searching, managing and re-using ontology. Some of the ontology classification mechanisms available are based on distinguishable features of ontologies. Examples include the following:

- the *subject* of ontologies (The DAML ontology library system classifies ontologies according to the Open Directory Category (www.dmoz.org));
- the *structure* of the ontology (The Ontolingua ontology library system has an inclusion lattice showing the inclusion relations between different ontologies);

- inter and intra ontology *features* ([17] indexed ontologies based on the intra and inter ontology features. Examples include *general, design process, taxonomy, axioms, inference mechanism, application, contributions, etc.*);
- the *lattice* structure ([18] built a lattice of ontologies showing the relevance of ontologies);
- the *dimensions* of the ontology ([19] indexed ontologies using dimensions (task/method dependency and domain dependency) to partition the library into a core library and a peripheral library);
- *stratified upper-level* ontology (ONIONS used generic, intermediate and domain layer to index ontologies),
- the *relations* of ontology ([17] indexed ontology based on defined relations, such as the subset/superset relation, extension relation, restriction, and mapping relation),
- the *components* of ontology ([17] also mentioned the indexing of ontology based on the component of ontologies, such as domain partitioning (partition domain in logical units), alternative domain views (polymorphic refinement), abstraction (abstract and detailed ontologies), primary ontologies versus secondary ontologies, terminological, information and knowledge modeling ontologies).

Modular organization in the ontology library system organizes units into modules. This serves to maximize cohesion within modules and minimize interaction between modules [20]. Most of the ontology library systems that aim to facilitate ontology re-use, ontology mapping and integration have adopted this structure. ONIONS also highlights the *stratified* design of an ontology library system. Different *naming policies* assist the ontology library system to achieve the modular organization or stratified storage of ontologies [21]. The disjointed partitioning of classes can facilitate modularity, assembling, integrating and consisting checking of ontologies. If, for instance, a certain class, such as 'people,' were disjointed from another class, say 'countries', then consistency checks could be carried out much sooner and faster. Thus, the partition modification has proven to be extremely valuable for editing purposes. Linking class names with their own contexts or using name space for differentiating them can serve to prevent violation within individual ontologies. As ontologies continue to grow, so too does the importance of systematic and consistent naming and organizational rules.

Identification. Unique ontology URL, Identifier and name are used as the identifier for ontologies in the ontology library systems.

Versioning. A version control mechanism is very important to an ontology library system. Unfortunately, most existing ontology library systems cannot support it, except for SHOE.

4.2 ***Adaptation***

Searching & Editing. An ontology library system should feature a visualized browsing environment, using hyperlinks or cross-references to closely related information. It should support collaborative editing and offer advanced searching features by adopting various existing information retrieval techniques, database searching features, or AI heuristic techniques. Ontology library systems could also monitor user profiles based on access patterns in order to personalize the view of ontologies [22].

Reasoning. A simple reasoning function should be included in order to facilitate ontology creation, ontology mapping and integration.

4.3 ***Standardization***

Language. Syntactically, an ontology representation language should be standardized or inter- or intra- ontology language translation should be supported. Semantically, an ontology library system should feature the *common vocabulary* (or faceted taxonomy). At any rate, it should eliminate the implicitness and misunderstanding of terms in different ontologies (due to synonyms, homonyms, etc.) for most generic classes. Preferably, an ontology library system should also support compatibility with or mapping between multiple controlled vocabularies from different domains. This would not only serve to guarantee flexibility in expressing an ontology semantically, but also to liquidate implicitness. The structures of these common vocabularies or multiple controlled vocabularies must be faceted, or modulated so as to facilitate the re-use, mapping and integration of ontologies [23]. These vocabularies can help in simple synonym matching, sibling analysis, and disjoint partition checking.
Upper-level Ontology. Standard upper-level ontology is important for better organization of ontology library systems (Ontolingua, IEEE SUO).

4.4 ***Others***

Ontology scalability. Ontology library systems should also consider increasing the scale of ontologies.
Maintaining facility. Ontology library systems should also provide some maintenance features, such as consistency checking, diagnostic testing, support for changes, and adaptation of ontologies for different applications.
Explicit documentation. Each ontology in an ontology library system should be extensively documented. The documentation should include such information as how the ontology was constructed, how to make extensions and what the ontology's naming policy, organizational principles and functions are. Such explicit documents about the ontologies themselves will pave the way for efficient ontology management and re-use.

Acknowledgement:
This research effort was supported by OnToKnowledge, a project funded by the European Union (www.ontoknowledge.org). We would like to thank Enrico Motta, Asun Gomez-Perez, Alexander Maedche, York Sure for providing useful information. We would also like to thank Michel Klein and Borys Omeliango for their helpful comments on drafts of this paper.

References

[1] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.
[2] O. Corcho, & A. Gomez-Perez, A road map on ontology specification languages. In *Workshop on Applications of Ontologies and Problem solving methods, 14th European Conference on Artificial Intelligence (ECAI'00),* Berlin, Germany, August 20-25, 2000.
[3] D.Jones, T. Bench-Capon & P. Visser, Methodology for ontology development. In *Proceedings of IT&KNOWS Conference of the 15th IFIP World Computer Congress* (Chapman-Hall), pp, 20-35. 1998.

[4] A. Maedche, & S. Staab, Ontology Learning for the Semantic Web. To appear in: IEEE Intelligent Systems. 16(2), March/April 2001.
[5] P.R.S. Visser, R.W. van Kralingen, & T.J.M. Bench-Capon, A method for the development of legal knowledge systems. *Proceedings of the Sixth International Conference on Artificial Intelligence and Law (ICAIL'97)*, Melbourne, Australia. 1997.
[6] J. Domingue, Tadzebao and WebOnto: Discussing, Browsing, and Editing on the Web. In B. Gaines and M. Musen (editors), *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Worksho*p, April 18th-23th, Banff, Canada, 1998. Available online at http://kmi.open.ac.uk/people/domingue/banff98-paper/domingue.html
[7] E. Motta, *Reusable Components for Knowledge Models*. PhD Thesis. Knowledge Media Institute. The Open University, UK, 1998.
[8] E Motta., S.Buckingham-Shum and J. Domingue, Ontology-Driven Document Enrichment: Principles, Tools and Applications. *International Journal of Human-Computer Studies* **52**, (2000) 1071-1109.
[9] J. B. Domingue and E. Motta, Planet-Onto: From News Publishing to Integrated Knowledge Management Support. *IEEE Intelligent Systems* **15**, (2000) 26-32.
[10] A.Farquhar, R. Fikes, & J. Rice, The Ontolingua server: Tools for collaborative ontology construction. *International Journal of Human Computer Studies* **46**, (1997) 707-728.
[11] A.J.Duineveld, R. Stoter, M.R. Weiden, B.Kenepa, V.R. Benjamins, WonderTools? A comparative study of ontological engineering tools. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Canada, October, 1999.
[12] R. Fikes,& A. Farquhar, Large-scale repositories of highly expressive reusable knowledge. *IEEE Intelligent Systems* **14**, (1999).
[13] S.Aitken, Extending the HPKB-Upper-Level Ontology experiences and observations. In *Proceedings of the Workshop on Applications of Ontologies and Problem Solving Methods(ECAI'98)*, Brighton, England, August 1998.
[14] J. Heflin & J. Hendler, Dynmaic ontologies on the Web. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000).* AAAI/MIT Press, Menlo Park, CA, 2000. pp. 443-449.
[15] J. Heflin, J. Hendler, and S. Luke, *SHOE: A Knowledge Representation Language for Internet Applications*. Technical Report CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science, University of Maryland at College Park. 1999.
[16] D.M. Pisanelli, A. Gangemi, & G. Steve, An Ontological Analysis of the UMLS Metathesaurus, *Journal of American Medical Informatics Association* **5** (1998) 810-814.
[17] P.R.S. Visser, and T.J.M. Bench-Capon, A Comparison of Four Ontologies for the Design of Legal Knowledge Systems. *Artificial Intelligence and Law* **6** (1998) 27-57.
[18] F. N. Noy, & C.D. Hafner, The state of the art in ontology design: A survey and comparative review. *AI Magazine* **4** (1997) 53-74.
[19] G. van Heijst, A.T. Schreiber, and B. J. Wielinga, Using explicit ontologies in KBS development. *Int. Journal of Human-Computer Studies* **45** (1997) 183-292.
[20] G. van Heijst, et al. A Case Study in Ontology Library Construction. *Artificial Intelligence in Medicine* **7** (1995) 227-255.
[21] D.L McGuinness,. R. Fikes, J. Rice, & S. Wilder, An environment for merging and testing large ontologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000).* Breckenridge, Colorado, April 12-15, 2000.
[22] J. Domingue, & E. Motta, A knowledge-based news server supporting ontology-driven story enrichment and knowledge retrieval. In D. Fensel and R. Studer (editors), *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW '99)*, LNAI 1621, Springer-Verlag, 1999.
[23] D.L. McGuinness, Conceptual modelling for distributed ontology environment. In *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS2000), Darmstadt, Germany, August 14-18, 2000.*