Building domain ontologies from lecture notes

Seminar Report

Submitted in partial fulfillment of the
requirements for the degree of

Master of Technology
by

Neelamadhav Gantayat
Roll No : 09305045

under the guidance of

Prof.Sridhar Iyer

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

April 2010

Abstract

Now a days many methodologies, tools and languages are available for building ontologies. Too many ways will mislead the Ontology developers and users, also it will create difficulties in integrating the existing Ontologies. A single centralized platform will help developers and the users of ontology to understand better and integration also can be made easy.

When a new ontology is going to be built, several basic question arise related to the methodologies, languages and tools. This study will present the main characteristics of methodologies, langauages and tools which can help developers to obtain knowledge about the available tools. The main objective here is to compare and review the main methodologies, languages and tools for building ontologies.

The future work in this field should be done towards the creation of a common platform for ontology developers to facilitate ontology development, exchange, evaluation, evoluation and management.

Contents

List of Figures

List of Tables

1 Introduction

The aim of ontology research is to develop knowledge representations that can be shared and reused. The main objective here is to compare the main methodologies, tools and languages for building ontologies. The main aim of this study is to develop a system which not only provides the user with the most relevant learning module for his query, but also provides him/her with the relevant information which is necessary for the user to know before proceeding with the querry. Suppose a user want to learn about a particular subject, the search tools typically just returns a large number of links to the user in reponse to his/her querry. Instead the search tools should provide the most relevant learning module for his query, and the pre-requisites and follow up modules that should be learned. This can be done by creating domain based ontology from the lecture videos. But right now no such tool exists[5].

2 Background

Ontology Borrowed from philosophy - the study of *"The nature of being"*. In general, an ontology provides a mechanism to capture information about the objects, Classes and the relationships that hold between them in some domain of interest. Ontology represents the particular meanings of terms and relations among them as they apply to that domain. Ontology is nothing but large number of ides and concepts to gather in a hierarchical order. The aim of ontology is to develop knowledge representations that can be shared and reused. Guber[6] defined an ontology as

"A formal explicit specification of a shared conceptualization."

2.1 Domain Ontology

Domain Ontology is one type of ontology Model which provides definitions and relationships of the concepts, and major theories and principles and activities in the domain. Domain ontologies provide shared and common understanding of a specific domain.

2.2 Developing Ontology from Text

Use some Natural Language processing tool to extract the keywords from the text files. Then find out the parts of speech, Noun-verb-Noun phrases. Find out the concepts and classes their dependencies and relationships. This can be done by many methods like finding the word frequencies, Noun ver patterns etc. Then build domain concepts and relationships and construct the dependency graph in hierarchical order which is the final formal domain ontology. The completely process is shown in the Figure 1.

2.3 Necessity of Ontology

- To share common understanding of the structure of information among people.
- To enable reuse of domain knowledge.

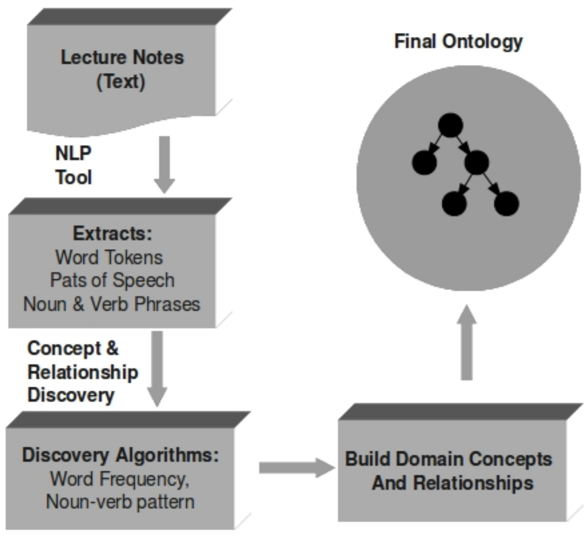

Lecture Notes (Text)
NLP Tool
Extracts: Word Tokens Pats of Speech Noun & Verb Phrases
Concept & Relationship Discovery
Discovery Algorithms: Word Frequency, Noun-verb pattern
Build Domain Concepts And Relationships
Final Ontology

Figure 1: Ontology Development from Text, Taken from[6]

- To analyze domain knowledge.

sharing common understanding of the structure of information among people or software agents is one of the common goals in developing ontologies. If the web sites share and publish the ontology having common terms, then computer agents can extract and aggregate information from these different sites.

Enabling reuse of domain knowledge is one of the important aspects now a days. For example if some one develops ontology for the notion of time such as notion of time intervals, relative measures in time then it can be reused by other developers. We can also reuse a general ontology and extend it to describe our domain of interest.

Analyzing domain knowledge is possible once a formal ontology with complete specification of the terms is available.

Ontology together with a set of individual instances of classes constitutes a knowledge base. Classes describe concepts in the domain. A class can have subclasses that represent concepts that are more specific than the superclass. Slots describe properties of classes and instances.

2.4 Applications of Ontology

Main application areas of ontology technology are knowedge management, Web commerce, and electronic business and in e-learning.

Knowledge Managment is concerned with acquiring, maintaining, and accessing an organization's knowledge. Nowadays organizations are distributed geographically and organized around the globe. With the large number of Online documents there are difficulties with Searching, Extracting and Maintaining these documents. Using Ontologies, semantic notations will allow structural and semantic definitions of documents. And these notations will provided intelligent search instead of keyword matching, query answering instead of information retrieval and document exchange between departments through ontology mappings.

Web commerce is an important and growing business area as it extends existing business models and reduces cost. examples of new business fields are online market places, and auction houses. The new task for customers is to find a shop that sells the product they are seeking, getting the desired quality, quantity and time, and paying as little as possible. Ontology mappings, which translate different product descriptions, can be used. It can describe the various products and help navigate and search automatically for the required information.

Electronic Business is nothing but automation of business transactions. Ontology based trading will significantly extend the degree to which data exchange is automated and will create completely new business models.

E-learning To find ou the dependencies between the repositories, and to fecilitate the user with more recent and relevant data.

2.5 Differences between Object Oriented and Ontology:

Object oriented programming and Ontology development both centers primarily around methods on classes, but in Object oriented programming the programmer makes design decisions based on the operational properties of a class, whereas an ontology designer makes design decisions based on the structural properties of a class. Therefore structure and relations among classes in an ontology are different from the structure for a similar domain in an object-oriented program.

In Object oriented programming Classes and objects in a program are about data structure where as in Ontology design Classes and objects in ontology are about the real world.

In a program a Name may be a string. In ontology a name is an object with many properties, it is not a string.

3 Various Methods of Developing Ontology:

Practically, developing an Ontology includes

- Defining classes in the ontology
- Arranging classes in a taxonomic (subclass-superclass) hierarchy

- Defining slots and describing allowed values for these slots.
- Filling in the values for slots for instances.

Before getting into various methods of constructing ontologies let us first emphasize on the fundamental rules in Ontology design[2].

1. There is no one correct way to model a domain. There are always alternatives.
2. Ontology development is necessarily an iterative process.
3. Concepts in the ontology should be close to objects (Physical or logical) and relationship in your domain of interest. There are mostly nouns(Objects) or verbs (relationships) in sentences that describe your domain.
4. An ontology is a model of reality of the world and the concepts in the ontology must reflect this reality.

3.1 Skeletal methodology

Proposed by Uschold and King[1], It defines four main phases

1. Identifying a purpose and scope
2. Building the ontology
   (a) Ontology capture
   (b) Ontology coding
   (c) Integrating existing Ontologies
3. Evaluation
4. Documentation

3.1.1 Purpose

It is important to be clear about the purpose of ontology and the intended users of the particular ontology. Some ontologies were developed to structure a knowledge base and some other ontologies are used as a part of a knowledge base.
For Example: Let us consider that we have to develop and ontology of operating system. The purpose for development may be to find out the dependencies between the course ware repositories for operating system object.

3.1.2 Building the Ontology

1. Capture:
   - Identification of the key concepts and relationships in the domain of interest (scoping).
   - Production of unambiguous text definitions for the concepts and relationships.

- Identification of terms to refer to such concepts and relationships.

  For Example: In operating system onotlogy the keywords may be *Types of Computing, Types of Systems, Process Management, Memory Management, File Management*

2. Coding:

   Coding is nothing but explicit representation of the conceptualization capture in the above stage in some formal language.
   For Example: If we arrange the keywords gathered above using freemind[7] the resultant is shown in Figure - 2

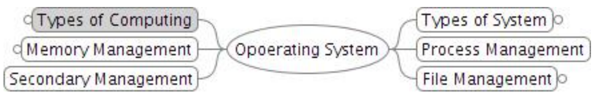


Figure 2: Operating System dependency graph

3. Integrating existing ontologies:
   In order to agree on ontologies that can be shared among multiple user communities, much work must be done to achieve agreement. One way forward is to make explicit all assumptions underlying the ontology.

3.1.3 Evaluation

Gomez-Perez's definition of evaluation in the context of knowledge sharing technology:

"to make a technical judgement of the ontologies, their associated software environment, and documentation with respect to a frame of reference The frame of reference may be requirements specifications, competency questions, and/or the real world."

Evaluation mainly dals with verification and validation that is validating the relations and verifying the purpose.

3.1.4 Documentation

All important assumptions should be documented, both about the main concepts defined in the ontology, as well as the primitives used to express the definitions in the ontology.

3.2 Seven-Step Method:

It is proposed by Noy and Deborah[2]. It describes the process of developing ontologies in following steps:

1. Determine the domain and scope of the ontology.

2. Consider reusing existing ontologies.

3. Enumerate important terms in the ontology.

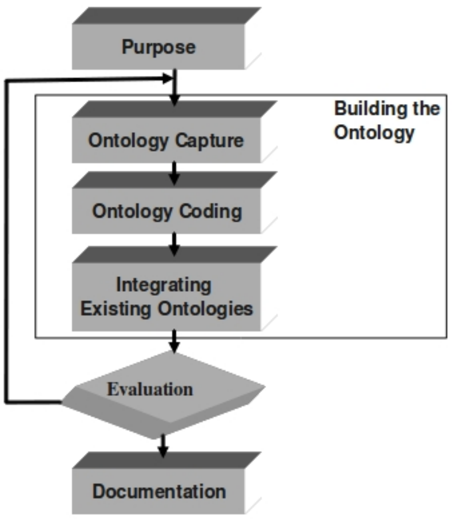


Figure 3: Skeletal Ontology Approach

4. Define the classes and the terms in the ontology.
5. Define the properties of classes slots.
6. Define the faces of the slots.
7. Create instances.

3.2.1 Determine Scope

Scope is nothing but the purpose of ontology. It should answer the following questions.

- What is domain that the ontology will cover?
- For what we are going to use the ontology?
- For what type of questions the ontology should provide answers?
- Who will use and maintain the ontology?

One of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer called competence questions. Some examples of competence questions are

- What resources does someone have authority to assign?
- Is someone allowed to perform some activity in any situation?
- What goal is some person committed to achieving?
- Is a goal achievable by an agent given its current commitments and the commitments of other agents?

- What goals are solitarily unachievable?

For Example: Let us consider that we have to develop and ontology of operating system. The purpose for development may be to find out the dependencies between the course ware repositories for operating system object.

3.2.2 Consider Reuse

Check if we can refine and extend existing sources for our particular domain and task. There are reusable ontologies on the web and in the literature.
Ex:

- www.ksl.stanford.edu/software/ontolingua
- www.daml.org/ontologies/
- www.unspc.org
- www.roselternet.org
- www.dmoz.org

3.2.3 Enumerate Terms

Write down a list of all terms we would like either to make statements about or to explain to a user. It is important to get comprehensive list of terms without worrying about concepts they represent relation among the terms, or any property of concepts or whether concepts or classes or slots.
For Example: In operating system onotlogy the keywords may be
Types of Computing, Types of Systems, Process Management, Memory Management, File Management

3.2.4 Define Classes

There are several approaches in developing a class hierarchy

Top-down approach: This process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts.

Bottom-up approach: This type of development process starts with the definition of the most specific classes the leaves of the hierarchy with subsequent grouping of these classes into more general concepts.

Combination Approach: This is a combination of the top-down and bottom-up approaches. We define the more salient concepts first and then generalize and specialize them appropriately. We might start with a few top-level concepts and a few specific concepts. We can then relate them to a middle-level concept.

None of these three methods is inherently better than any of the others the approach to take depends strongly on the personal view of the domain. The combination approach is often the easiest for many ontology developers, since the concepts in the middle tend to be more descriptive concepts in the domain. Which ever approach it is we start by defining classes. From the list which is derived from step-3 select the terms that describe objects having independent existence rather than terms that describe these objects. These terms will be classes in the ontology and will become anchors in the class hierarchy. If a class A is a superclass of class B, then every instance of B is also an instance of A. I.e Class B represents a concept that is a "kind of" A.
For Example: If we arrange the keywords gathered above using freemind[7] the resultant is shown in Figure - 2

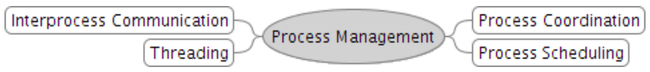


Figure 4: Defining classes of Process

3.2.5 Define Properties

Once we have defined some of the classes, we must describe internal structures of concepts. After selecting the classes from the list created by step-3 most of the remaining terms are likely to be properties of these classes. For each property in the list we must determine which class it describes. These properties becomes slots attached to classes. In general, there are several type of object properties that can become slots in an ontology.

- "intrinsic" properties such as flavor of a wine
- "extrinsic" properties such as wines name and "area" it comes from.,
- parts, if the object is structured: these can be both physical and abstract "parts".
- Relationship to other individuals: relationship between individual member of the class and other items.

All subclasses of a class inherit the slot of that class . We can add additional slots tenn in level (low, moderate (or) high) to the inherited classes. A slot should be attached at the most general class that can have that property.
For Example: The properties for threads may be user level or kernel level threads.

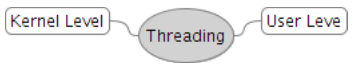


Figure 5: Properties of Threads

3.2.6 Define Constraints

slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take. Example:

- value of a name slot is one string i.e. name is a slot with value type string.
- A slot produces can have multiple values and the values are instances of the given class. i.e. produces in a slot with value type Instance with name of the class as allowed class.

Several common facets:

Slot cardinality: Slot cardinality defines how many values slot can have, For example single cardinality (allowing at most one value) and multiple cardinality (allowing any number of values). Some systems allow specification of a minimum and maximum cardinality to describe the number of slot values more precisely. i.e. minimum cardinality of N means that a slot must have at least N values. And similarly Maximum cardinality of M means that a slot can have at most M values.

Slot-value type: A value type facet describes what type of values can fill in the slot.
String: the value is a simple string used for slots such as name.
Number: describes slots with numeric values , more precisely can have integer and float .

Ex: price

Boolean: Slots simply Yes-No (True-False) flags.
Enumerated: It specify a list of specific allowed values for the slot

Ex: flavor slot can take strong, moderate, delicate

Instance: These type of slots allow definition of relationships between individuals. Slots with value type instance must also define a list of allowed classes from which the instances can come.
Domain and Range of a Class
Allowed classes for slots of type instance are often called a range of a slot. The classes to which a slot is attached or a classes which property, a slot describes are called the domain of the slot. Basic rules for determining a domain and a range of a slot are similar. When defining a domain or a range for a slot find the most general classes or class that can be respectively the domain or the range for the slots. On the otherhand do not define a domain and range that is overly general. All the classes in the domain of a slot should be described by the slot and instances of all the classes in the range of a slot should be potential fillers for the slot. Do not close an overlay general class for range (i.e. one would not want to make the range THING) but one would want to chose a class that will cover all filters.
Some import rules in defining range and domain:

- If a list of classes defining a range or a domain of a slot includes a class and its subclasses remove the subclass.

- If a list of classes defining a range or a domain of a slot contains all subclasses of a class A, but not the class A itself the range should contain only the class A and not the subclass.
- If a list of classes defining a range or a domain of a slot contains all but a few subclasses of a class A consider if the class A make a more appropriate range definition.

3.2.7 Create instance

Define an individual instance of a class requires

1. Choosing a class.
2. Creating an individual instance of the class.
3. Filling in the slot values.

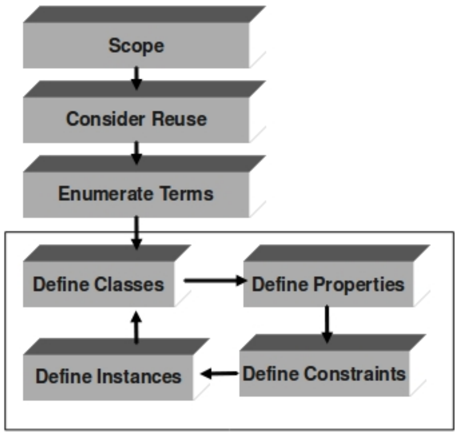


Figure 6: Seven-Step Ontology Approach

3.3 Practical Approach:

proposed by Gavrilova[3], It consists of 5-steps recipe for creating ontology:

1. Glossary development
2. Laddering
3. Disintegration
4. Categorization
5. Refinement

3.3.1 Glossary development

Gather all the information relevant to the described domain. The main goal of the step is selecting and verbalizing all the essential objects and concepts in the domain. **For Example:** In operating system onotlogy the keywords may be *Types of Computing, Types of Systems, Process Management, Memory Management, File Management*

3.3.2 Laddering

Define the main levels of abstraction. Specify the type of Ontology classification such as taxonomy, partonomy, and genealogy.
For Example: If we arrange the keywords gathered above using freemind[7] the resultant is shown in Figure - 2

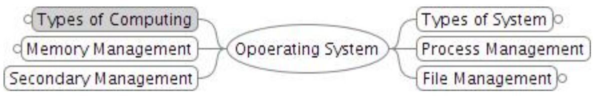


Figure 7: Operating System dependency graph

3.3.3 Disintegration

Break high level concepts, built in the previous step, into a set of detailed ones where it is needed. This could be done via a top-down strategy trying to break the high level concept from the root of previously built hierarchy.

3.3.4 Categorization

Detailed concepts are revealed in a structured hierarchy and the main goal at this stage is generalization via bottom-up structuring strategy. This could be done by association similar concepts to create meta-concepts from leaves of the aforementioned hierarchy.
For Example: If we arrange the keywords gathered above using freemind[7] the resultant is shown in Figure - 8

3.3.5 Refinement

The final step is to updating the visual structure by excluding the excessiveness, synonymy, and contradictions. As mentioned before, the main goal is harmony and clarity.

3.4 Knowledge Engineering Approach:

It was better described in *"Development of Domain ontology for e-learning courses"*[4] which was appeared in ITIME - 09 IEEE international symposium.

1. Identify purpose and requirement specification:
2. Ontology acquisition:
3. Ontology implementation:

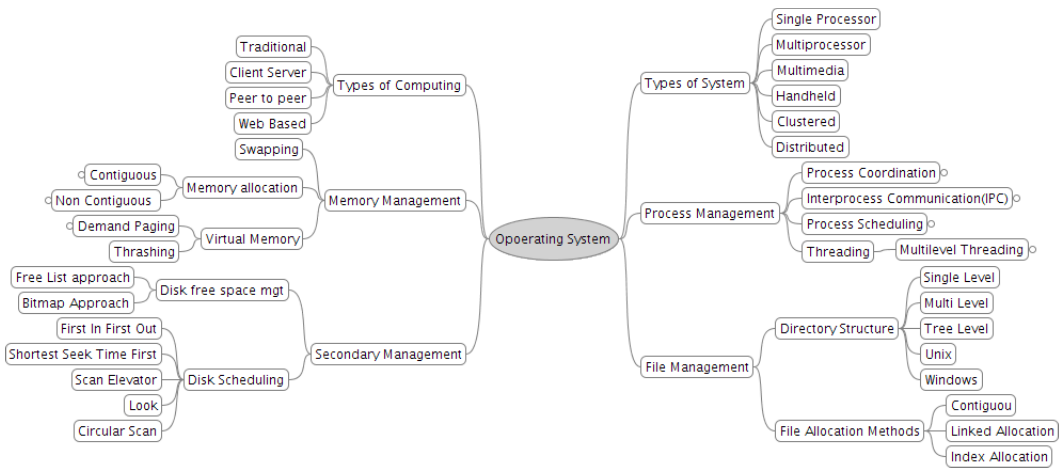


Figure 8: Complete Operating System ontology

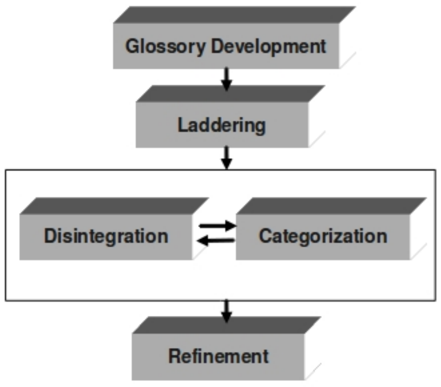


Figure 9: Practical Ontology Approach

4. Evaluation/Check:
5. Documentation:

3.4.1 Identify purpose and requirement specification

Ontology purpose, scope and its intended use, i.e. the competence of the ontology. **For Example:** Let us consider that we have to develop and ontology of operating system. The purpose for development may be to find out the dependencies between the course ware repositories for operating system object.

3.4.2 Ontology acquisition

Capture the domain concepts based on the ontology competence. It envolves

1. Enumerate important concepts and terms in this domain

2. Define concepts, properties and relations of concepts, and organize them into hierarchy structure.

3. Consider reusing existing ontology.

For Example: In operating system onotlogy the keywords may be *Types of Computing, Types of Systems, Process Management, Memory Management, File Management.* If we arrange the keywords gathered above using freemind[7] the resultant is shown in Figure - 10

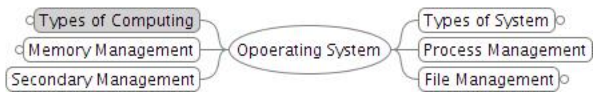


Figure 10: Operating System dependency graph

3.4.3 Ontology implementation

Explicitly represent the conceptualization captured in a formal language

3.4.4 Evaluation/Check

The ontology must be evaluated to check whether it satisfies the specification requirements.

3.4.5 Documentation

All the ontology development must be documented, including purposes, requirements, textual descriptions of the conceptualization, and the formal ontology.

4 Ontology languages

ontology languages are formal languages used to construct ontologies. It can formalloy describe the meaning of terminology used in web documents.

Need of Ontology languages Ontologies can be viewed as Database Schema but we cannot utilize the database schema. because database schema is more rigid and can fit into set of tables but where as ontology will not fit into tables. Fensel [8] points out the following differences between ontologies and schema definitions:

- A language for defining ontologies is syntactically and semantically richer than common approaches for databases.
- The information that is described by an ontology consists of semi-structured natural language texts and not tabular information.
- An ontology must be a shared and legal terminology because it is used for information sharing and exchange.

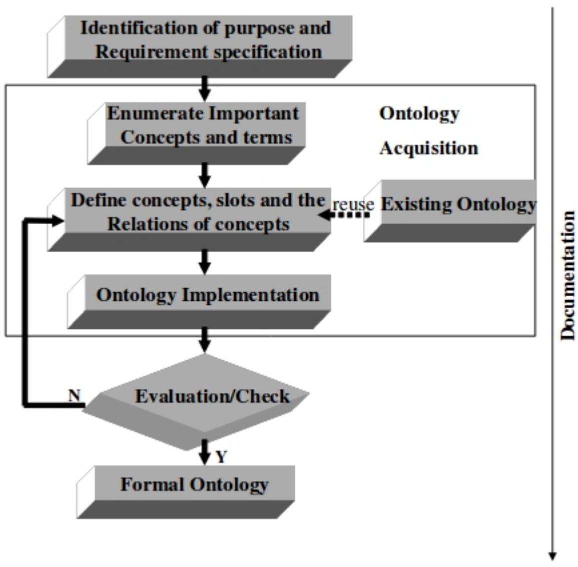

Identification of purpose and Requirement specification
Enumerate Important Concepts and terms
Ontology Acquisition
Define concepts, slots and the Relations of concepts
reuse
Existing Ontology
Ontology Implementation
Evaluation/Check
N
Y
Formal Ontology
Documentation

Figure 11: Knowledge Engineering Approach

4.1 History of Ontology Languages

At the beginning of the 1990's, a set of AI-based ontology implementation languages were created. Figure - 12 describes various ontology languages.

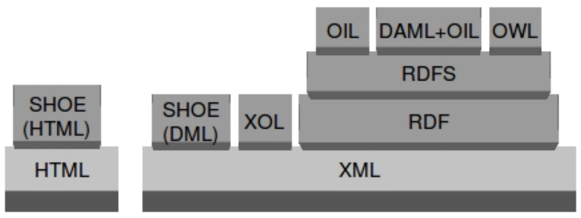


Figure 12: Stack of Ontology Markup Languages taken from [9]

SHOE was built in 1996 as an extension of HTML, in the University of Maryland. It uses set of tags which are different form the HTML specification thus it allows insertion of ontologies in HTML documents. SHOE just allows representing concepts, their taxonomies, n-ary relations, instances and deduction rules.

Then XML was created and widely used as a standard language for exchanging information on the web. Then SHOE syntax was modified to includes XML, and some other ontology languages are also built on XML.

XOL was developed by the AI center of SRI international, in 1999. It is a very restricted langauge where only concepts, concept taxonomies and binary relations can be specified. No inference mechanisms are attached to it. It is mainly designed for the exchange of ontologies in the biomedical domain.

Then RDF was developed by the W3C (The world wide web consortium) as a semantic-network based language to describe Web resources. RDFSchema was built by the W3C as an extensioin to RDF with frame-based primitives. The combination of both RDF and RDFSchema is normally known as RDF(S). RDF(S) is not very expressive. It just allows the concepts, concept taxonomies and binary relations.
Three more languages have been developed as extensions to RDF(S): OIL, DAML + OIL and OWL

OIL was developed in the framework of the European IST project On-To-Knowledge. It adds frame-based Knowledge Representation primitives to RDF(S), and its formal semantics is based on description logics.

DAML + OIL was created by a joint committee from the US and the EU in the context of the DARPA project DAML. DAML + OIL also adds DL-based KR primitives to RDF(S). Both OIL and DAML + OIL allow representing concepts, taxonomies, binary relations, functions and instances. Many efforts are being put to provide reasoning mechanisms for DAML + OIL.

Finally, in 2001, the W3C formed a working group called Web-Ontology (WebOnt) Working Group. The aim of this group was to make a new ontology markup language for the Semantic Web, called OWL (Web Ontology Language). Brief Description of the Languages

4.2 XML (Extended Markup Language)

XML[10] is a markup language for delivery of documents containing structured information over the web. structured information contains both content and some indication of what role that content plays. In HTML the tag semantics and the tag set are fixed. It doesnot provide arbitrary structure. XML specifies neither semantics nor a tag set. i.e. there is no fixed tags in XML, and XML provides a facility to define tags and the structural relationships between them. XML is created so that richly structured documents can be used over the web.

XML documents are composed of markup and content. Following kinds of markups that can occur in an XML document: elements, comments, processing instructions etc.
Elements: elements identify the nature of the content they surround, some elements may be empty or non empty if an element is not empty, it begins with a start-tag,

```
<element>
```

and ends with an end-tag,

```
</element>
```

attributes, are name-value pairs that occur inside start-tags after the element name. For example

```
<div class="preface">
```

is a div element with the attribute class having the value preface. In XML all attribute values must be quoted.
Comments: Comments begin with <!- - and end with - ->. Comments can contain any data except the literal string - -. We can place comments between markup anywhere in the document.
Processing Instructions: Processing instructions are an escape sequences to provide information to an application. Like comments, they are not textually part of the XML document, but the XML processor is required to pass them to an application. Processing instructions have the form:

```
<?name pidata?>
```

Basic XML code looks like

Table 1: Book Store

Book Id	Title	Author	Year	Price
059600	XML	John	2005	30
059601	Javascript	David	2003	29.99

For the above text the XML code is

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="book.css"?>
<bookstore>
  <book Id="059600">
    <title lang="en">XML</title>
    <author>John</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book Id="059601">
    <title lang="en">Javascript</title>
    <author>David</author>
    <year>2003</year>
    <price>29.99</price>
  </book>
</bookstore>
```

The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).

The next line describes the root element of the document (like saying: ”this document is for bookstore”) i.e. <bookstore>, The next 4 lines describe 4 child elements of the root (title, author, year, price), And finally the last line defines the end of the root element <bookstore>

we can attach cascaded style sheet can be attached to the XML document by adding the following line

```
<?xml-stylesheet type="text/css" href="book.css"?>
```

The above XML code can be explained graphically as follows

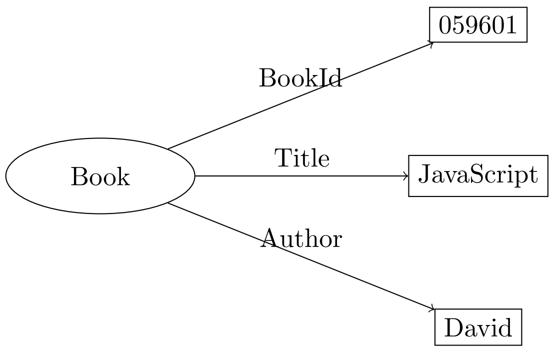


Figure 13: graphical representation

4.2.1 XMLS(XML Schema)

XML Schema[8] is a means for defining constraints on well formed XML documents. It provides basic vocabulary and predefined structuring mechanisms for providing information in XML. XML Schemas are extensible, because they are written in XML.

That is we can Reuse our Schema in other Schemas, we can Create our own data types derived from the standard types and we can Reference multiple schemas in the same document. XML Schema provides seveal significant improvements:

- XML Schema definitions are themselves XML documents. The clear advantage is that all tools developed for XML (e.g., validation or rendering tools) can be immediately applied to XML schema definitions, too.
- XML Schemas provide a rich set off datatypes that can be used to define the values of elementary tags.
- XML Schemas provide a much richer means for defining nested tags (tags with subtags)
- XML Schemas provide the namespace mechanism to combine XML documents with heterogeneous vocabulary.

With XML Schemas, the sender can describe the data in a way that the receiver will understand. A date like: ”03-11-2004” will, in some countries, be interpreted as 3.November and in other countries as 11.March.

However, an XML element with a data type like this:

```
<date type="date">2004-03-11</date>
```

Let us consider an example

for the above XML code the XML Schema will be

Table 2: Message

To	From	Heading	Body
John	David	Reminder	Meeting is cancelled.

```
 <?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The note element is a **complex** type because it contains other elements. The other elements (to, from, heading, body) are **simple types** (string) because they do not contain other elements. Final XML code with reference to the XML Schema

```
 <?xml version="1.0"?>

<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
  <to>John</to>
  <from>David</from>
  <heading>Reminder</heading>
  <body>Meeting is cancelled.</body>
</note>
```

ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".

It provides syntax for structured documents but no semantic constraints on the meaning of these documents.

A language for restricting the structure of XML documents and extends XML with data type.

4.3 RDF (Resource Description Framework)

Resource Description Framework(RDF)[11] is a graphical language used for representing information about resources on the web. It is a basic ontology language. RDF is written in XML, By using XML, RDF information can easily be exchanged between different types of computers using diffferent types of operating systems and application languages. RDF was designed to provide a common way to describe information so it can be read and understood by computer applications. RDF descriptions are not designed to be displayed to the web. Data model for objects and relations between them. provides a simple semantics for datamodel. Data models can be represented in an XML syntax. **Basic RDF code looks like**

Table 3: Student Information

Student Id	Name	Subject	Marks	Percentage
059600	John	Networks	40	80
059601	David	Networks	45	85

For the above data the RDF code is

```
 <?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:st="http://www.cse.iitb.ac.in/st#">

<rdf:Description
rdf:about="http://www.cse.iitb.ac.in/st/059600">
  <st:name>John</st:name>
  <st:subject>Networks</st:subject>
  <st:marks>40</st:marks>
  <st:percentage>80</st:percentage>
</rdf:Description>

<rdf:Description
rdf:about="http://www.recshop.fake/st/059601">
  <st:name>David</st:name>
  <st:subject>Networks</st:subject>
  <st:marks>45</st:marks>
  <st:percentage>90</st:percentage>
</rdf:Description>
</rdf:RDF>
```

The first line of the RDF document is the XML declaration. The XML declaration is followed by the root element of RDF documents: ¡rdf:RDF¿.

The xmlns:rdf namespace, specifies that elements with the rdf prefix are from the namespace

```
"http://www.w3.org/1999/02/22-rdf-syntax-ns#".
```

The xmlns:cd namespace, specifies that elements with the cd prefix are from the namespace

`"http://www.iitb.ac.in/st#".`

The <rdf:Description>element contains the description of the resource identified by the rdf:about attribute.

The elements: <st:name>, <st:subject>, <st:marks>, etc. are properties of the resource.

RDF identifies with Uniform Resource Identifiers (URI) These are called resources. The base element of the RDF model is the triple: a subject linked through a predicate to object. We will say that <subject>has a property <predicate>valued by <object>

The RDF triple (S,P,O) can be viewed as a labelled edge in graph.

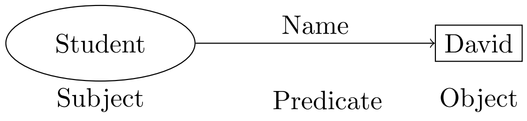


Figure 14: Predicate

4.3.1 RDFS (RDFSchema)

Vocabulary for describing properties and classes of RDF resources[12], with a semantics for generalization - hierarchies of such properties and classes. It defines a simple ontology that particular RDF documents may be checked against to determine consistency.Many RDFS components are included in the more expressive language Web Ontology Language (OWL).

4.4 OIL (Ontology Interchange Language)

OIL is also known as Ontology Inference Layer[13]. OIL is derived from RDFS. OIL is based on descriptive logic. Descriptive logic describes knowledge in terms of concepts and role restrictions that can automatically derive classification taxonomies.

OIL is better then its ancestors in the following ways. it has rich set of modelling primitives and nice ways to define concepts and attributes. The definitions of a fomal semantics were included in OIL. Customized editors and interface engines for OIL are also exist. Every RDFS ontology is an valid ontology in OIL and vice versa. Much of the work in OIL was subsequently incorporated into DAML+OIL and the Web Ontology Language (OWL).

4.5 OWL (Web Ontology Language)

In 2001, the W3C formed a working group called Web-Ontology (WebOnt) Working Group. The aim of this group was to make a new ontology markup language for the Semantic Web, called OWL (Web Ontology Language)[14].

OWL is used when the information contained in documents needs to be processed by application. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between the terms. It is a revised version of the DAML + OIL web ontology language. OWL adds more vocabulary for describing properties and classes.
Sibilings of OWL are

1. OWL Lite
2. OWL DL
3. OWL Full

4.5.1 OWL Lite

OWL lite supports classification hierarchy and simple constraints. OWL Lite provides a quick migration path for thesauri and other taxonomies. OWL Lite has a lower formal complexity then OWL DL.

4.5.2 OWL DL

Maximum expressiveness while reatining computational completeness and decidable i.e. all computations will be finished in time. OWL DL is named due to its correspondence with Description Logic. and it includes all the OWL language constructs.

4.5.3 OWL DL

OWL Full gives syntactic freedom of RDF with no computational guarantees. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary

The following set of relations hold. Their inverses do not.

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

The choice between OWL-Lite and OWL-DL may be based upon whether the simple constructs of OWL-Lite are sufficient or not. Where as the choice between OWL-DL and OWL-Full may be based upon whether it is important to be able to carry out automated reasoning on the ontology or whether it is important to be able to use highly expressive and powerful modelling facilities such as meta-classes (classes of classes).

OWL Full can be views as an extension to RDF. where as OWL Lite and OWL DL can be viewed as and extension of a restricted view of RDF.

Every OWL (Lite, DL, Full) document is an RDF document and every RDF document is an OWL Full document. Only some RDF documents can be OWL lite or OWL DL.

5 Ontology Editors

Ontology editors are designed to assist in the creation or modification of ontologies and for the subsequent ontology usage. These editors are the applications which support one or more ontology languages. And some editors also have the facility to export from one to another ontology language.

5.1 Ontolingua

The Ontolingua Server was the first ontology tool created[15]. It was developed in the Knowledge Systems Laboratory (KSL) at Stanford University. The public server is available at http://www-ksl-svc.stanford.edu:5915/, we must have an account to use the system. All work takes place with in a session, which has a duration and a description. Ontolingua uses Object-Oriented presentation, and full logical representation. The ontology is stored on the server. we can download our work to a local file system or we can email our work to any system. Uses of Ontolingua are:

1. Runtime query
2. Translation

Runtime Query, remote applications may query and ontology server

- To determine if a term is defined
- To determine the relationship between terms
- To manipulate the contents of an ontology

Translation, from one language to another language. It includes the following challenges

- **Semantics -** ensure that the meaning is preserved.
- **Syntax -** ensure that target syntax is correct.
- **Style -** ensure that target idioms a re preserved.

5.2 Protégé

Protégé is a free, open source ontology editor and a knowledge acquisition system[16]. It was developed by the Stanford Medical Informatics(SMI) at Stanford University. It is an opensource, standalone application with an extensible architecture. It is written in Java and heavily uses Swing to create the complex user interface. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. The Protégé platform supports two main ways of modeling ontologies via

1. Protégé-OWL
2. Protégé-Frames

5.2.1 Protégé-OWL

The Protégé-OWL editor is an extension of Protégé that supports the Web Ontology Language (OWL). An OWL ontology may include descriptions of classes, properties and their instances.
The Protégé-OWL editor enables users to:

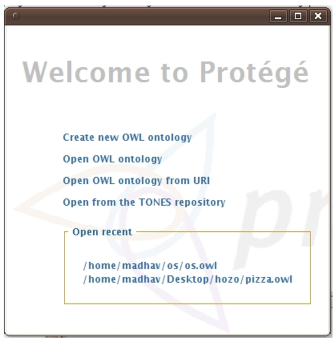
1. Load and save OWL and RDF ontologies.
2. Edit and visualize classes, properties, and rules.
3. Define logical class characteristics as OWL expressions.

5.2.2 protégé-frames

The Protégé-Frames editor provides a full-fledged user interface and knowledge server to support users in constructing and storing domain ontologies. In this model, an ontology consists of a set of classes organized in hierarchical order to represent a domain's concepts. classes are associated with a set of slots to describe their properties and relationships, and a set of instances of those classes.

5.2.3 Developing Ontology

For creating ontologies using protégé install protégé in the system and run from the command prompt then a dialoug box similar to as shown in the system will appear. We can create our own Ontology using protégé. A sample operating



Welcome to Protégé

Create new OWL ontology
Open OWL ontology
Open OWL ontology from URI
Open from the TONES repository

Open recent
/home/madhav/os/os.owl
/home/madhav/Desktop/hozo/pizza.owl

Figure 15: protégé welcome screen

system ontology developed using protege is shown in the below figure.

5.3 WebODE

WebODE is a tool for building ontologies in the World Wide Web[18]. It was developed in the Artificial Intelligence Lab from the Technical University of Madrid(UPM). Web ODE is not used as a standalone application, but we can

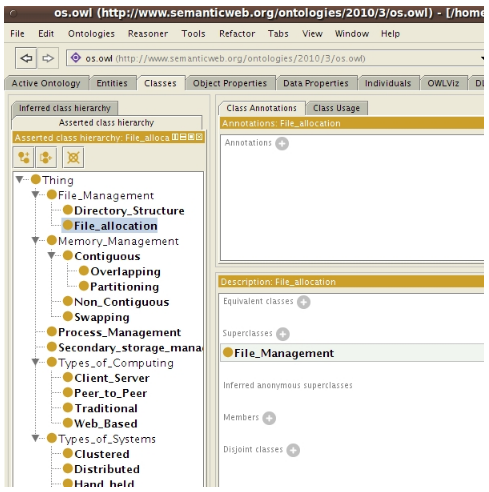


Figure 16: Operating system ontology developed using protégé

use it as a Web server with a Web interface. Multiple concurrent users are allowed to work at the same time, and the proper synchronisation and blocking mechanisms are provided so that ontology designers can make their best with this tool either working alone or in a team. WebODE is based on a central ontology repository implemented using a relational database. A very simple and powerful mechanism to export and import ontologies using the XML standard is also supplied by default with the tool. Technical requirements

1. Browser (Internet Explorer 5.0 or above)
2. Java plug-in version 1.2.x must be installed in the browser
3. configure the Web browser so that the pages are retrieved from the server everytime they are visited. this can be done by
   Tools / Internet Options / Internet Temporal Files / Configuration / Every Time the page is Visited.
4. Do not use the proxy for the WebODE URL. This option can be configured selecting:
   Tools / Internet Options/Connection / LAN Configuration.

Usage Login in this website http://www.oeg-upm.net/webode

New Ontology: give a name and description for the ontology that is being created. where name is compulsary and description is optional.

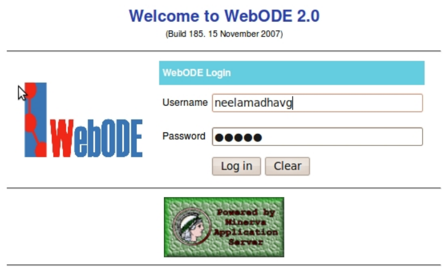
List of Ontologies: we can list the ontologies that we are able to access (and modify) by clicking on the list ontologies tab from the main menu.

Open Ontology: we can open ontologies, to insert new components, to update them and to simply remove some of them.

Export Ontology: , We can export the ontology into several langauages like UML, XML, RDF(S), OIL, DAML+OIL, OWL and so on, and we can get the target code.

5.3.1 Developing Ontology

First we have to register for WebODE, after registration we will get a username and password. Log on to the webode website `http://www.oeg-upm.net/webode` using the username and the password. We can create new ontology us-



Welcome to WebODE 2.0
(Build 185. 15 November 2007)

WebODE Login

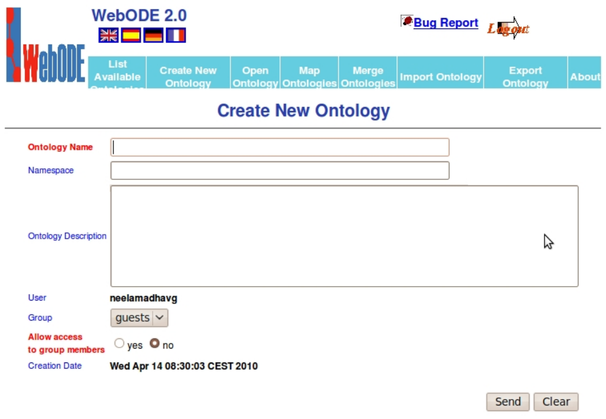
Username	neelamadhavg
Password	●●●●●

Log in Clear

Powered by Minerva Application Server

Figure 17: Operating system ontology developed using protégé

ing webode the new ontology scree will appear as shown in the following figure. After creating the ontology it will look like the following figure. Here we have



Create New Ontology

Ontology Name	
Namespace	
Ontology Description	
User	neelamadhavg
Group	guests
Allow access to group members	yes [✓ no]
Creation Date	Wed Apr 14 08:30:03 CEST 2010

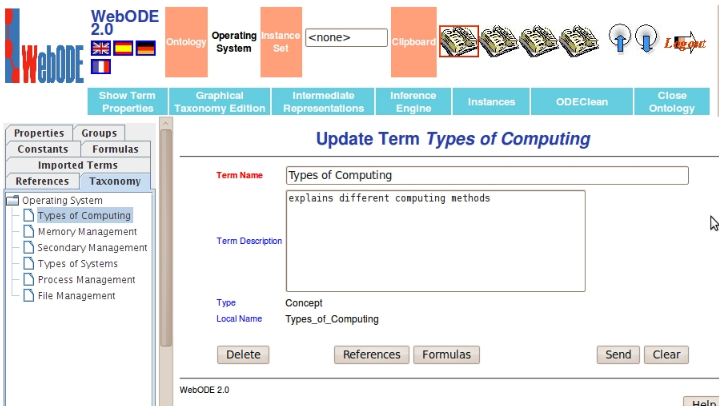
Send Clear

Figure 18: Operating system ontology developed using protégé

created the Ontology for Operating system course.

5.4 OntoStudio

OntoStudio is an modeling environment to create and maintain ontologies[17], with particular emphasis on rule-based modelling. It was originally developed for F-Logic but now also includes some support for OWL, RDF, and OXML. It also includes functions such as the OntoStudio Evaluator. The Evaluator is used for the implementation of rules during modelling. It is the successor of OntoEdit.



Update Term *Types of Computing*

Term Name: Types of Computing

Term Description: explains different computing methods

Type: Concept

Local Name: Types_of_Computing

Figure 19: Operating system ontology developed using protégé

5.4.1 Data Models

OntoStudio can be operated with a RAM-based- as well as database-data-model; therefore it is scalable and suitable for the modeling of large ontologies.

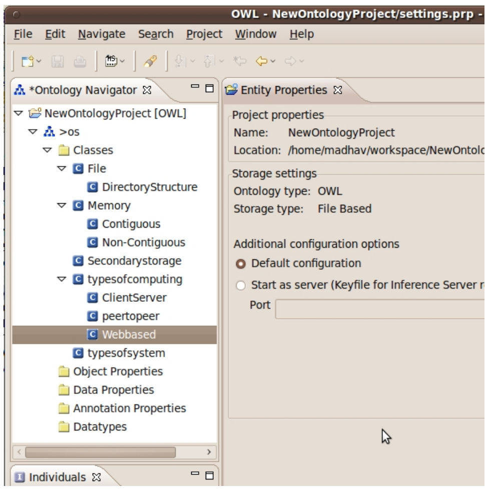
1. **RAM Data Model:**
   The complete ontology data is loaded into the main memory of the workstation. All changes made in the ontology are written in files when you save the new ontologies. It is appropriate for small and medium-size ontologies (dependent on main memory size)

2. **Database Data Model**
   Only the actual presented parts of the ontologies are loaded from a database into main memory and will be written back to the database immediately when changes happens in the ontology. It is appropriate for large ontologies.

OntoStudio supports the ontology languages F-Logic, RDF/RDFS and OWL.

5.4.2 developing Ontology

For creating ontologies using OntoStudio install OntoStudio in the system and run from the command prompt. After creating the ontology it will look like the following figure. Here we have created the Ontology for Operating system course.



Project properties

Name: NewOntologyProject

Location: /home/madhav/workspace/NewOntolo

Storage settings

Ontology type: OWL

Storage type: File Based

Additional configuration options

[✓ Default configuration]

Start as server (Keyfile for Inference Server r

Port

Figure 20: protégé welcome screen

6 Future Work

A model can be developed which will use Natural Language Processing tools to find out the keywords and their dependencies. According to the relationship between the keywords we can derive a heirarichy and finally we can construct an ontology which not only provides the user with the most relevant learning module for his query, but also provides him/her with the relevant information. For example we can construct the ontology from the lecture notes provided in NPTEL repository which can help users in getting the pre-requisite and follow-up modules for a particular module.

7 Conclusion

There is no one correct way to model a domain. There are always alternatives. Ontology development is necessarily an iterative process. There is no correspondence between ontology building methodologies and tools. since there is no technological support for most of the existing methodologies, they cannot be easily applied in the ontology construction task. Most of the tools just focus on few activities of the ontologyh lifecycle (Design and implementation). There are many similar tools are available for building ontology but they do not inter-operate. That means we can use the ontology developed by one tool in another tool.

Ontology development languages are still in development phases, and they are continuously evolving, which makes it difficult to have up-to-date technology for managing them. The following table is the summary of

Table 4: Comparision of Ontology development tools

	Ontolingua	**Protégé**	**WebODE**	**OntoStudio**
Developers	KSL (Stanford University)	SMI (Stanford University)	UPM	Ontoprise
Current Release and Tools	0.1.45(Aug 2003)	4.1Alpha(Mar 2010)	2.0(Dec 2002)	2.3.3(Dec 2009)
Pricing Policy	Free Web Access	Free Ware	Lincences	Freeware & Licences
Mode of Access	Web Access	Stand Alone	Web Access	Stand Alone
Export to Languages	CLIPS CML, ATP CML	XML, RDF(S), XMLSchema, OWL	XML, RDF(S), OIL, DAML+OIL	XML, RDF(S), F-Logic, OWL
Import from Languages	IDL KIF	XML, RDF(S), XMLSchema, OWL	XML, RDF(S)	XML, RDF(S), F-Logic, OWL
Ontology Library	Yes	Yes	No	Yes

References

[1] Gruninger, M. and Fox, M.S, *"Methodology for the Design and Evaluation of Ontologies"*, Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, 1995.

[2] Natalya F. Noy and Deborah L. McGuinness, *"Ontology Development 101: A Guide to Creating Your First Ontology"*, Available from Internet:`<http://protege.standford.edu/publications/ontology_development/ontolgogy101.html>`

[3] Gavrilova, T., Farzan, R. and Brusilovsky, P., *"One Practical Algorithm of Creating Teaching Ontologies"*, Proceedings of Network Based Education, 2005.

[4] YUN Hong-yan, XU Jian-liang, WEI Mo-ji, XIONG Jing, *"Development of Domain Ontology for E-learning Course"*, ITIME-09IEEE international symposium, 2009.

[5] Nidhi Malik, under the Guidance of Prof.Sridha Iyer *"Discovering Dependencies in Courseware Repositories"*, M.Tech. Dissertation, 2008.

[6] T.R.Guber *"Towards principles for the design of ontologies used for knowledge sharing,"*, *Int..J.Human-Computer Studies,*, 43(5-6), p.p 9.7-928, 1993.

[7] Freemind `http://freemind.sourceforge.net/wiki/index.php/Main_Page`

[8] Michel Klein, Dieter Fensel, Frank van Harmelen, and Ian Horrocks, *"The relation between ontologies and XML schemas"*.

[9] Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez, *"Methodologies, tools and languages for building ontologies. Where is their meeting point"*, Data & Knowledge Engineering 46(2003) 41-64.

[10] XML: `http://en.wikipedia.org/wiki/XML`

[11] Pierre-Antoine Champin `''RDFTutorial''` April 5, 2001.

[12] RDF Schema: `http://en.wikipedia.org/wiki/RDF_schema`

[13] Dieter Fensel and Frank van Harmelen *"OIL: An Ontology INfrastructure for the semantic web"* IEEE Intelligent Systems, The semantic web magazine, 2001.

[14] Deborah L. McGuinness, Frank van Harmelen *"OWL Web Ontology Language Overview"*, `http://www.w3.org/TR/owl-features/`, W3C Recommendation 10 February 2004.

[15] *Ontolingua:*, `http://ksl.stanford.edu/software/ontolingua/`

[16] Protégé: `http://protege.stanford.edu/`

[17] *OntoPrise:* `http://www.ontoprise.de/en/home/products/ontostudio/`

[18] *WebODE:* `http://webode.dia.fi.upm.es/WebODEWeb/Documents/usermanual.pdf`