

UNIVERSITÉ DE MONTRÉAL

IMPLEMENTATION OF A SEMANTIC WEB SITE
FOR A COMPUTER ENGINEERING DEPARTMENT

CHANGSHAN SUN

DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLOME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)

AVRIL 2005



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-494-03918-3

Our file Notre référence

ISBN: 0-494-03918-3

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

IMPLEMENTATION OF A SEMANTIC WEB SITE
FOR A COMPUTER ENGINEERING DEPARTMENT

présenté par: ChangShan Sun
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment acceptée par le jury d'examen constitué de:

M. DAGENAI Michel , Ph.D., président

M. GAGNON Michel, Ph.D., membre et directeur de recherche

M. DESMARAI Michel, Ph.D., membre

Acknowledgment

I would like to thank my supervisor, professor Michel Gagnon for his guidance and constructive criticisms throughout this project and the writing of this paper, he brought me opportunities to develop my interest in research and gave me invaluable comments in the research project. Also I would like to thank the jury members: professor Dagenais Michel R. and professor Desmarais Michel C. who have accepted to evaluate this thesis.

At last I want to thank my wife and my new born little baby, my wife has overcome so many difficulties to let me have more time working on this project, and my little baby let me know well what is the responsibility of a father.

Résumé

Le web classique se compose de documents et de textes dont le contenu sémantique n'est pas explicite. Les ordinateurs ne peuvent pas comprendre et traiter ces données automatiquement. Les technologies classiques du web fournissent une capacité limitée pour les moteurs de recherche basés essentiellement sur des mots-clé. Dans le web sémantique, des ontologies sont employées pour stocker les données et les moteurs de recherche peuvent faire une recherche basée sur ces concepts, et de ce fait réaliser la tâche entière et fournir une réponse plus exacte à la requête de l'utilisateur. Ce mémoire de maîtrise présente une architecture d'un site web sémantique basé sur le langage d'ontologie OWL. Elle repose sur un moteur de recherche sémantique qui peut accomplir une recherche plus précise et plus intelligente que celle réalisée par d'autres moteurs de recherche sémantiques actuellement disponibles.

Abstract

Classical web consists of documents and text which have no real meaning. Computers could not understand and process this data automatically. Classical web technologies provide limited capacity for search engine based essentially on keyword matches. Quite differently, semantic web is the web in which the data is understandable to both human and machine. It makes the classical “web of links” replaced by a “web of meaning”, where the web resources are described in an explicit language. Semantic web is not a web of documents or texts, but a web of knowledge and concepts that are shared by different resources. In semantic web, ontology is used to conceptualize different objects in a certain (specific or common) domain and also to provide computer with machine-readable knowledge to process.

This work introduces an three layers approach for the architecture of a semantic web site which is based on the OWL ontology language. It relies on a flexible and powerful semantic search agent system which is called SWSAS (semantic web search agent system). We easily get information from SWSAS that would not be obtained so easily in a traditional web site. As a semantic web search agent, it can offer instant and accurate query, it is not only superior to the search engine of classical website, but also some advantage which other semantic search engines do not have. SWSAS has “simple semantic search” and “advanced semantic search” available to use. In “advanced semantic search” user can choose “triple (subject, predicate and object) clause search” as well as the multiple clauses search. This search agent support both English and French language. We set up an ontology by modeling the information of the computer engineering department including information about the professors, courses and researches. The ACM computer science taxonomy is creatively used in developing the ontology. SWSAS makes inference and get potential information from OWL ontology, meanwhile, it is also compatible for the ontology language DAML and RDF(S).

The prototype has been implemented and evaluated, and we conclude that the techniques applied are valuable and usable for the proposed domain.

Condensé en français

1. Introduction

Les données de WWW (world wide web) peuvent être exprimées par le langage HTML (hypertext markup language) sous forme de pages du Web. Ces pages, ce que nous avons appelé Web Classique, consistent en du texte pour la lecture et des liens à d'autres pages. Les gens obtiennent l'information ou la connaissance du Web en passant en revue les pages du Web ou à l'aide d'un moteur de recherche. Un moteur de recherche est censé donner à l'utilisateur une réponse précise à sa question. Puisque les données du web sont conçues surtout pour les humains, l'ordinateur ne sait pas bien accéder et traiter ces données. En outre, le moteur de recherche n'a pu ni intégrer l'information de différentes sources ni les extraire. Soit, par exemple, un utilisateur qui veut obtenir certaines informations du site web du département de génie informatique de l'École Polytechnique de Montréal (www.polymtl.ca). Il doit premièrement essayer de trouver le bouton ou le lien sur les pages web qui sont liées à sa requête. Il est tout à fait possible que le bouton ou le lien choisi ait beaucoup de sous-liens, forçant l'utilisateur à vérifier chaque couche pour confirmer si elle contient la réponse exacte. Si l'utilisateur choisit plutôt d'utiliser le moteur de recherche du site web, le résultat sera donné par un ensemble de liens. Alors l'utilisateur doit identifier quel lien est celui désiré? Il doit les consulter un après l'autre jusqu'à ce qu'il en trouve quelques uns parmi eux qui répondent à sa question. Ainsi, les utilisateurs deviennent confus ou impatients lorsqu'ils n'arrivent pas à trouver les liens corrects pour leurs questions.

Le web sémantique vise à résoudre ces problèmes. Dans un web sémantique, les données sont compréhensibles par l'homme et la machine. Le "web classique des liens" est remplacé par un "web de signification", où les ressources du Web sont décrites dans un langage explicite. Le web sémantique n'est pas un web de documents ou de textes, mais plutôt un web de connaissances et de concepts qui sont partagés par des ressources différentes. En laissant à l'ordinateur accomplir la plupart des applications du site web et automatiser les applications sur le web, sans aide humaine, le web sémantique exprime

les données en format compréhensible par une machine, en utilisant des ontologies.

2. Ontologies

Le web sémantique doit ajouter une sémantique compréhensible pour la machine, en employant des ontologies pour définir et organiser des données. Une ontologie est une base sémantique qui est employée pour conceptualiser différents objets tels que des choses, des événements et des relations dans un certain domaine (spécifique ou commun) et pour fournir également à l'ordinateur la connaissance compréhensible par une machine. Une ontologie abstrait la connaissance d'un certain domaine, offre une compréhension commune de cette connaissance en fournissant un vocabulaire, et donne une définition formalisée de la terminologie et de ce vocabulaire. Une ontologie est composée de la définition de concepts et de contraintes sur ces concepts. Ces concepts et ces contraintes sont formels afin de rendre les connaissances lisibles par la machine et facilite le partage de ces connaissances. L'ontologie peut également être réutilisée par d'autres programmes d'application, ou être employée dans une autre ontologie comme ressource.

La modélisation d'ontologie est une tâche très importante, car le but d'une ontologie est de faire un modèle d'une certaine réalité du monde[8]. Tous les concepts dans une ontologie correspondent aux entités. Les langages d'ontologie ont leur propre définition de sémantique ou de théorie de modèle[10]. Un modèle définit les rapports entre la syntaxe et les interprétations. Il y a deux familles populaires d'ontologies qui sont largement répandues dans la représentation de la connaissance: la représentation de connaissance [11] et la logique description [12]. Il y a plusieurs langages d'ontologie, parmi lesquels on retrouve RDF(s)[2][5], DAML+oil[7] et OWL[6]. RDF fournit un langage de base qui permet l'interopérabilité entre les applications qui échangent l'information sur le web. RDFS fournit les primitives de base pour définir des propriétés et des types des ressources du web. DAML peut en outre exprimer des restrictions à la définition de classes et des restrictions aux domaines et images des propriétés. Le plus récent langage d'ontologie du web, OWL, étend DAML+OIL avec trois différentes

variantes: OWL Lite, OWL DL et OWL Full. OWL Lite et DL sont décidable tandis que OWL Full ne l'est pas complètement. OWL Full peut être considéré comme une extension de RDF, alors que OWL Lite et OWL DL peuvent être considérés comme des extensions d'une vue restreinte de RDF.

3. Technologie de sémantique

Le web sémantique est une extension du web existant, qui donne au contenu du web une signification bien définie. Il se base sur quelques normes, vocabulaires et ontologies afin d'associer et de relier beaucoup de ressources de données qui pourraient être liées à d'autres documents et ressources d'information. Il y a beaucoup de différences entre web classical et le web sémantique. Le web classical est basé sur HTML, le web sémantique est basé sur XML et RDF(s); Web classical est le web des documents, web sémantique est web des connaissances; le moteur de recherche de web classical effectue ses recherches par des mots-clés, tandis que le moteur de recherche du web sémantique effectue ses recherches par du concepts; le web classical est compréhensible pour les humains mais difficiles pour le traitement de machine (ambiguïté, formats de données sans contrainte), l'information compréhensible pour une machine est exprimée dans un format qui est non ambigu et favorable au traitement par la machine; la base de données à l'intérieur du CW(Web Classique) est séparée, la base de données dans le SW(Web Semantique) est une base de données liée globalement.

Les données du web sémantique sont basées sur les ontologies spécifiques: des entités telles que des ressources, des classes, des propriétés, des images et des domaines peuvent être créés pour décrire les significations et les relations des concepts et de tous les objets relatifs. Une des méthodes pour représenter l'information sémantique dans un web sémantique est en définissant des triplets de URI(Uniform Resource Identifier). Ces triplets se composent d'un sujet, d'un attribut et d'un objet. Ces URIs peuvent être édités dans un document d'ontologie et les utilisateurs peuvent se référer à eux librement. Non seulement ils identifient des sources d'information (telles que des pages du Web) mais réfèrent également indirectement à des ressources (telles que des articles et des objets).

La technologie de recherche d'information [23] dans un web sémantique est censée être une manière facile et précise pour obtenir de l'information de manière exacte et immédiate. Puisque les données d'ontologie traitées automatiquement par ordinateur, les données sont les concepts réels avec la vraie signification. Les agents de web sémantique sont des applications, qui communiquent avec d'autres agents de web sémantique et leurs données, et naviguent sur l'Internet, pour rassembler, filtrer et traiter des données, afin de fournir à l'utilisateur de l'information utile et appropriée.

Plusieurs API peuvent être employés pour développer une application du web sémantique. Certaines APIs d'ontologie, tels que Jena, OWL Genie, DAML API et Pellet..., peuvent être employés comme analyseur et reasonneur d'ontologie dans le développement du SW. Il y a également des architectures de SW sur l'Internet qui peuvent être prises comme référence, comme par exemple Mindswap, qui fournit des informations et des programmes d'application pour le SW et est développé par l'Université du Maryland.

4. Vers un système SW

Étant donné l'imperfection du site web traditionnel du Département de génie informatique et de son moteur de recherche, un site web sémantique a été développé afin de démontrer son avantage en fiabilité et intelligence. Comme illustré à la figure 1, l'architecture entière comprend trois couches principales: le client, le serveur et des serveurs principaux d'ontologie (la connaissance). Dans ce site web sémantique, couche du client consiste en un fureteur, la couche du serveur accomplit l'interaction avec le moteur de recherche, et comprend un visualisateur adapté aux besoins du client, et l'outil d'annotation. Enfin, la couche de stockage peut être un système de fichiers local ou à distance qui accueille les bases de connaissance. Il y a un portail de la trousse d'outils d'annotation dans le serveur par lequel le système peut sélectionner l'ontologie utilisée. Dans le module d'agent de recherche il y a un module d'exécution de requête qui analyse l'information et effectue quelques inférences sur ces données. Enfin, le résultat de recherche est présenté par le module adapté aux besoins du client et l'utilisateur peut alors obtenir le résultat par le fureteur. Un composant important de ce site s'appelle

SWSAS (système sémantique d'agent de recherche du Web), qui comprend l'agent sémantique de recherche et une Ontologie du département de génie informatique, et constitue notre principale contribution.

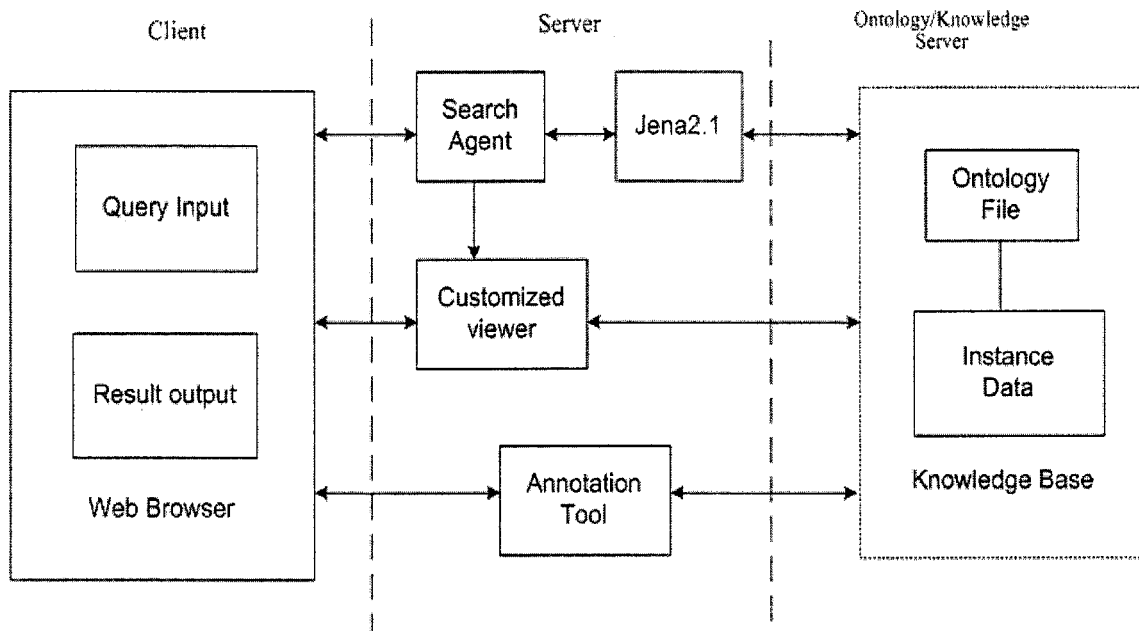


Figure I Architecture d'un SW

Développement de l'ontologie

En général, avant de concevoir une ontologie, des spécifications pour l'ontologie doivent être recueillies et analysées. Il faut concevoir le modèle conceptuel, qui comprend des définitions de la classe et la hiérarchie de classe et leurs propriétés. Eventuellement, d'autres ontologies peuvent être réutilisées et intégrées. La dernière étape est la création des différents instances des classes dans la hiérarchie. Il y a beaucoup d'outils disponibles pour développer l'ontologie, dont Protégé (avec le OWL plugin).

L'ontologie créée dans ce projet a été conçue pour la gestion du département de génie informatique. Elle est basée sur l'information sur les professeurs, les cours, la recherche et toutes autres informations relatives. L'ontologie, de ACM [18] a été adoptée pour la

classification et définition de la recherche et des cours. Dans cette ontologie, il y a 1515 classes, 62 propriétés et 1440 instances pour SWSAS qui couvrent tous les aspects des cours, recherches, informations sur les professeurs, l'information sur les matières scolaires, etc. En particulier, l'ontologie contient 1473 items pour représenter le domaine de l'informatique.

Utilisation de Jena comme ontologie

Jena 2.1 a été employé comme raisonneur et API de l'ontologie dans le système. Jena 2.1 est une API mûre pour l'ontologie de base de RDF, de DAML+OIL, de OWL et le traitement de données. Elle est compatible avec toutes les normes recommandées par le consortium de W3C. Selon l'évaluation de SemWebCentral [33], nous avons la table suivante qui fait la comparaison avec d'autres API ou raisonneurs :

Table I Comparaison d'APIs ou de Reasoners

subject	Jena 2	OWL Genie	DAML API	RDF API	Pellet	Racer
Language Support	OWL& DAML,RDF	OWL	DAML+OIL	RDF	OWL	OWL
Based on	Java	XSL	Java	Java	Java	Lisp
Complete consistency checker	Yes	No	No	No	Yes	Yes
Interface	Java,DIG[34], Command Line	xslt	Java,DIG	Java,DIG	Java,DIG	Java, DIG
Query language	RDQL	XSL	RDQL	RDQL	RDQL	Racer Query Language
Reasoner	Yes	No	No	No	Yes	Yes
Parser/validator	Yes	No	No	No	Yes	No
Java API	Yes	No	Yes	Yes	No	No
Open sources	Yes	Yes	Yes	Yes	Yes	No

Algorithme de recherche en SWSAS

Il y a deux modes de requête dans SWSAS: la recherche sémantique simple et la recherche sémantique avancée. Pour répondre à une requête sémantique simple, l'algorithme SimpleSearch illustré à la figure 3 est utilisé. Afin de mettre en application la recherche sémantique avancée, l'algorithme InstanceSearch et l'algorithme AdvancedSemanticSearch sont utilisés. L'idée principale de ces algorithmes est d'obtenir le résultat exact du processus sémantique de recherche, lequel doit rechercher si des concepts sémantiques reliés avec la description D dans l'ontologie existent et, dans le cas affirmatif, renvoyer leurs classes équivalentes, les instances de classe, les superclasses et les sous-classes par le raisonneur Jena.

Algorithm SimpleSemanticSearch(Description D)

```

1: if reasoner.isInconsistent(D) then
2:   raise Exception
3: end if
4: result  $\leftarrow \emptyset$ 
5: bclass  $\leftarrow$  Reasoner.listClasses(D);
6: if bclass  $\neq \emptyset$  then
7:   equivalents  $\leftarrow$  reasoner.getEquivalents(D)
8:   if equivalents  $\neq \emptyset$  then
9:     for each c  $\in$  equivalents do
10:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
11:    end for
12:   End if
13:   instances  $\leftarrow$  Reasoner.getInstances(D);
14:   if instances  $\neq \emptyset$  then
15:     for each c  $\in$  instances do
16:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
17:    end for
18:   end if
19:   subclasses  $\leftarrow$  Reasoner.getSubClasses(D);
20:   If subClasses  $\neq \emptyset$  then
21:     for each c  $\in$  subClasses do
22:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
23:    end for
24:   end if
25: else
26:   bproperty  $\leftarrow$  Reasoner.getProperties(D)
27:   if bproperty  $\neq \emptyset$  then
28:     for each c  $\in$  Instance do

```

```

29:         result ← result U allIndividuals(c)
30:     end for
31: end if
32: end if
33: return result.

```

Figure II Algorithme SimpleSemanticSearch

Les algorithmes AdvancedSemanticSearch et InstanceSearch sont utilisés pour la recherches triplet (sujet, attribut, objet) qui est employée dans la recherche sémantique de SWSAS. Elle décompose d'abord l'exemple d'ontologie en ensemble de triplets, identifie alors l'instance en trouvant les triplets qui correspondent à la requête d'utilisateur.

Algorithm InstanceSearch (class, property, D)

```

1: Result ← ∅;
2: inst ← class.listInstance(class)
3: if inst ≠ ∅ then
4:   for each c ∈ inst do
5:     iValue ← c.listPropertyValues(property);
6:     if iValue ≠ ∅ then
7:       for each cc ∈ iValue do
8:         if cc == D then
9:           result ← result U (cc)
10:        end if
11:      end for
12:    end if
13:  end for
14: end if

```

Figure III Algorithme InstanceSearch

Algorithm AdvancedSemanticSearch (class, property, D)

```

1: if reasoner.isInconsistent(D) then
2:   raise Exception
3: end if
4: result ← ∅
5: result = InstanceSearch(class,property,D)
6: iEqClass ← Reasoner.listEquivalentClasses(D);
7: if iEqclass ≠ ∅ then
8:   for each c ∈ iEqclass do
9:     result ← result U InstanceSearch(c, property, D)

```

```

10:     end for
11: end if
12: iSubClass ← Reasoner.listSubClasses(D);
13: if iSubClass ≠ ∅ then
14:     for each c ∈ iSubClass do
15:         result ← result U InstanceSearch(c, property, D)
16:     end for
17: end if
18: iSubProperty = dp.listSubProperties(D);
19: if iSubProperty ≠ ∅ then
20:     for each c ∈ iSubProperty do
21:         result ← result U InstanceSearch(class, c, D)
22:     end for
23: end if

```

Figure IV Algorithme AdvancedSemanticSearch

Caractéristiques de SWSAS

SWSAS a plusieurs caractéristiques intéressantes. Premièrement, SWSAS supporte non seulement des ontologies de OWL, mais également des ontologies de DAML et RDF. Deuxièmement, dans SWSAS, deux modèles de recherche sont à la disposition des utilisateurs. Un est le moteur de recherche sémantique simple et l'autre est le moteur de recherche sémantique avancée. Les deux peuvent fonctionner indépendamment l'un de l'autre. Avec la recherche sémantique simple, l'utilisateur n'a qu'à entrer un mot dans son requête, puis SWSAS peut renvoyer beaucoup d'information sémantique générale reliée avec ce mot. Dans le moteur de recherche sémantique avancé, l'utilisateur peut faire la recherche en construisant une clause qui comprend un sujet, un attribut et un objet. Troisièmement, il supporte des requête à clauses multiples, contrairement aux autres moteurs de recherche développés pour le web sémantique. Il supporte également la combinaison booléenne des clauses. Quatrièmement, il comprend plusieurs langues telles l'anglais et le français. Ceci est très important parce qu'il peut satisfaire différents utilisateurs avec différentes langues. Cinquièmement, l'utilisation de l'ontologies de ACM pour la définition de sujets des cours du département de génie informatique de l'université, fait de SWSAS un outil facilement adaptable à un autre département de génie informatique.

Défis

Plusieurs défis ont été relevés dans ce travail. Premièrement, le système demande beaucoup de programmation; dans le cas de ce projet, quatre milles lignes de code de Java étaient nécessaires et ont exigé beaucoup de technologies et d'habilité. Deuxièmement, c'est un défi d'utiliser Jena, parce qu'en ce moment il y a un manque de documentation de référence pour développer un projet avec Jena. Le troisième défi est la construction de l'ontologie: pour la rendre plus raisonnable et plus acceptable. En outre, il a fallu un effort pour adapter l'ontologie de ACM à notre problèmes. Le quatrième défi est le support de langues multiples de ce système, particulièrement pour la langue française.

5. Évaluation de SWSAS

Ce système de recherche documentaire a une compatibilité élevée. Il est écrit en Java afin de pouvoir fonctionner dans plusieurs systèmes d'exploitation différents comme Windows, Linux ou Unix. Il peut également fonctionner avec différentes ontologies en différentes langues OWL, DAML ou RDF. Dans ce système, la puissance de déduction et de raisonnement est fournie par le raisonneur de Jena 2.1 qui est aussi écrit en Java.

La taxonomie d'ACM est employée comme ontologie de base. Dans ce projet, 80 pour cent des concepts de la taxonomie d'ACM pourraient être employés directement dans notre ontologie. L'ontologie de ACM constituant déjà une norme, son utilisation dans notre système offre un avantage certain.

Afin de rechercher quelques informations sur les professeurs, les cours, la recherche et toutes autres informations relatives, cela prend à des utilisateurs plusieurs mesures et beaucoup de temps, de ce réaliser dans le moteur de recherche ou du webpage de École Polytechnique de Montréal. Dans SWSAS, une mesure suffit pour obtenir le résultat.

Il y a actuellement beaucoup d'autres moteur de recherche pour le web sémantiques.

Quel est l'avantage de SWSAS sur ces moteurs de recherche? Le table en suivant montre en détail la comparaison de SWSAS avec ces moteurs de recherche:

Table II Comparaison de SWSAS avec l'autre moteur de recherche

Subject	OWL Semantic Search	Swoogle	Search Engine for the Semantic Web (BETA)	The SHOE Search Engine	SWSAS
Support OWL	No	Yes	No	No	Yes
Support DAML	Yes	Yes	No	No	Yes
Support RDF	Yes	Yes	Yes	No	Yes
Triple clause support	Yes	No	No	No	Yes
Mulitple clauses support	Yes	No	No	No	Yes
Simple words Semantic Search	No	Yes	Yes	Yes	Yes
Mulitple languages Support	No	No	No	No	Yes
Semantic Statistic Support	No	Yes	No	No	No

6. Conclusion et travaux futurs

Le but du web sémantique est de structurer les meta-données afin de traiter, ou de rechercher par la signification, les contenus dans le site web. Il peut surmonter l'imperfection du web actuel et utiliser pleinement les ressources de l'Internet. L'ontologie constitue un outil central permettant la flexibilité, la réutilisabilité et l'expressivité des possibilités de communication exigés par le web sémantique. Le langage OWL est la technologie la plus récente recommandée par W3C et devient un standard largement accepté pour le développement d'ontologies. OWL est non seulement un langage d'ontologie mais également une langage de logique descriptive.

Dans ce travail, je propose SWSAS, un moteur de recherche sémantique qui peut

répondre à une requête rapidement et précisément. Il est non seulement supérieur au moteur de site web classique, mais aussi à d'autres moteurs de recherche du web sémantique. Il a deux modes de recherche qui aident l'utilisateur à définir exactement sa requête afin d'obtenir le résultat désiré, dont un est une mode sémantique simple de recherche et l'autre est une mode sémantique avancé de recherche. Ce système SWSAS est conçu pour extraire des données non seulement à partir des pages spécifiques, mais le Web entier sur l'Internet. Il peut également supporter plusieurs langues, telles que le français et l'anglais. Jéna 2.1 est employé et joue un rôle important dans le développement de ce système.

Dans l'avenir, j'ai l'intention d'améliorer SWSAS dans les aspects suivants: approfondir la gestion de la connaissance du web sémantique pour la rendre capable de résumer et classifier automatiquement un document par concepts; approfondir la fonction du 'e-learning' pour les étudiants et les professeurs, qui est basé sur le web sémantique et fournir tous les moyens pour le 'e-learning', ce qui comprend le développement d'ontologie et l'annotation basée sur l'ontologie des sujets d'étude, de leur composition dans des cours. Un autre travail qui doit être effectué est l'évaluation des ontologies. Par exemple, s'il y a deux ontologies qui emploient les mêmes concepts mais en différents termes, comment les distinguer ou évaluer? Tout cela devrait être considéré dans les recherches futures.

Table of Contents

Acknowledgment.....	IV
Résumé.....	V
Abstract.....	VI
Condensé en français.....	VII
Table of Contents.....	XIX
List of Tables	XXII
List of Figures.....	XXIII
List of abbreviation.....	XXV
 CHAPTER 1 Introduction.....	 1
1.1 Search engines for Classical Web (CW) are limited.....	2
1.2 Examples of the search engine problems in CW.....	3
1.3 Semantic Web (SW) could probably solve the problem.....	6
1.4 My research project and organization of thesis.....	9
 CHAPTER 2 Ontologies.....	 10
2.1 Definition of ontology.....	10
2.2 Ontology Modeling[7].....	12
2.2.1 Frame Oriented Knowledge Representation.....	12
2.2.2 Knowledge Representation in Description Logic(DL).....	17
2.2.3 Inference in DL ontology.....	21
2.3 Description languages.....	21
2.3.1 XML and XML Schema.....	22
2.3.2 RDF and RDFS.....	25
2.3.3 DAML+OIL and OWL.....	31
2.3.4 OWL Ontology Example.....	35
2.3.5 Relation of Description Languages.....	41
2.4 Reusability of Ontologies.....	42
2.5 Ontology Develop Tools.....	43
2.5.1 Protégé with OWL plugin.....	43
2.5.2 Other Ontology tools.....	44

CHAPTER 3 Technology of Semantic Web.....	47
3.1 From CW to SW.....	47
3.2 Ontology Languages for Approaching SW.....	49
3.3 Ontologies in Semantic Web.....	51
3.4 IR(Information Retrieval) in SW.....	54
3.4.1 IR technology in CW and SW.....	54
3.4.2 SW Agent.....	55
3.5 Ontology API.....	56
3.5.1 API Jena2.1.....	56
3.5.2 Several Other Ontology APIs or Reasoners.....	58
3.5.3 Why Jena2.1 Is Appropriate For Our Project.....	59
3.6 Knowledge Management and E-learning In SW.....	61
3.6.1 Knowledge Management.....	61
3.6.2 e-learning [32].....	62
3.7 Trust between SW sites [33].....	63
3.8 SW architectures and applications.....	64
3.8.1 Examples of SW architectures.....	64
3.8.2 SW Search Engine Examples.....	66
3.8.3 W3C (World Wide Web Consortium).....	69
CHAPTER 4 Toward a System of SW.....	71
4.1 Structure of the SW System.....	71
4.2 Development of the Ontology.....	73
4.2.1 Protégé as tool for developing the ontology.....	73
4.2.2 Procedure used to develop an ontology.....	74
4.2.3 Using ACM Computing Taxonomy.....	79
4.2.4 Metrics of the ontology.....	80
4.2.5 Why database is not used to store the Ontology file	81
4.3 Use Jena as API.....	81
4.4 Characteristic of SWSAS.....	83
4.4.1 Support both OWL ontology and DAML ontology.....	84
4.4.2 Two Search Models.....	86
4.4.3 Support Multiple Clauses Search.....	88

4.4.4 Support multiple languages.....	90
4.5 Algorithm of Search in SWSAS.....	91
CHAPTER 5 Evaluation of SWSAS.....	94
5.1 General evaluation.....	94
5.2 Comparison with other SW Search Engines.....	102
5.3 Conclusion about SWSAS.....	104
5.4 Challenges in developing SWSAS.....	106
CHAPTER 6 Conclusion and future work.....	108
REFERENCE	110

List of Tables

Table I Comparaison d'APIs ou de Reasoners.....	XII
Table II Comparaison de SWSAS avec l'autre moteur de recherche.....	XVII
Table 3.1 Comparison of APIs or Reasoners.....	60
Table 3.2 Tools in Mindswap.....	65
Table 5.1 Evaluation Questionnaire.....	96
Table 5.2 Evaluation Result	97
Table 5.3 Evaluation of SWSAS	99
Table 5.4 Comparison of SWSAS with other Search Engine.....	104

List of Figures

Figure I Architecture d'un SW.....	XI
Figure II Algorithme SimpleSemanticSearch.....	XIV
Figure III Algorithme InstanceSearch.....	XIV
Figure IV Algorithme AdvancedSemanticSearch.....	XV
Figure 2.1 Abox Inference.....	21
Figure 2.2 XML based RDF.....	26
Figure 2.3 RDF Eexample.....	29
Figure 2.4 Another RDFS example.....	29
Figure 2.5 Three versions of OWL	34
Figure 2.6 Example of Simple class definition.....	38
Figure 2.7 Ontology Language Level.....	41
Figure 2.8 Snapshot of Protégé 2.1.....	44
Figure 2.9 Snapshot of OntoEdit	45
Figure 2.10 Snapshot of OilEd	46
Figure 3.1 Comparison of CW and SW.....	48
Figure 3.2 Layers of Description Languages [22].....	49
Figure 4.1 Overall Architecture of a SW.....	72
Figure 4.2 The Definition of An Instance of Professor Class.....	79
Figure 4.3 A fraction of Computer Science Taxonomy.....	83
Figure 4.4 Upload Ontology.....	85
Figure 4.5 Semantic Search	87
Figure 4.6 Advanced Semantic Search.....	88
Figure 4.7 Multiple Clause Search of SWSAS.....	89
Figure 4.8 Multiple Languages.....	90
Figure 4.9 Simple Semantic Search Algorithm	92

Figure 4.10 InstanceSearch Algorithm.....	92
Figure 4.11 Advanced Semantic Search Algorithm.....	93

List of Abbreviation

WWW	world wide web
HTML	hypertext markup language
CW	Classical Web
SW	Semantic Web
RDF	Resource Description Framework
RDFS	RDF Schema
OWL	Web Ontology Language
DAML	DARPA Agent Markup Language
OIL	Ontology Interface Layer
W3C	World Wide Web Consortium
MT	Model Theory
DL	Description Logic
XML	eXtensible Markup Language
XMLS	XML Schemas
URIs	Uniform Resource Identifier
RDQL	RDF Data Query Language
DIG	DL Implementation Group
AI	Artificial intelligence
IR	Information Retrieval

CHAPTER 1

Introduction

Almost everyone can sense the changes that internet brings to the human's life. People use WWW(world wide web) for searching information, for providing information and electronic business. Once a website has its own host address and content settled, all users from different places of the world can browse the website with this address as well as use a search engine for searching some information. Data can be expressed in HTML language (hypertext markup language) and web pages are always available to people for reading or linking. This is what we called Classical Web (CW) .

The number of websites and the quantity of information are growing very fast. There are millions of websites now. People begin to find some shortcomings and problems with the Classical Web which are limiting the progression of this technology.

People get information or knowledge from Web through two ways. One is browsing the web pages, clicking on the button or links; the other one is getting information from a search engine. A search engine is supposed to give us a precise and accurate answer to our question. But in CW all the web pages use format languages such as HTML for representation, and Web data are still mainly presented in natural language. This makes more difficult the task of automatically accessing and processing the available Web data. Since the Web data is human understandable but not machine-understandable, the computer does not know well how to access, process and make use of the data. Then, when the search engine is used to search something, what it could do is just search for keywords, which is not exactly the function of "answering questions".

For example, on the website of the Department of Computer Engineering of École Polytechnique de Montréal (www.gi.polymtl.ca), where a user wants to get some information from its website, if he does it by browsing the webpages, he must find the

button or link which is related to the desired content. Meanwhile, it is quite possible that the chosen button or link has many sub-layers, forcing him to check each layer to confirm whether it contains the exact answer or not. If the user rather chooses to use the search engine in that website, the result would be given as a set of links. Then the user must identify which link is the expected one. He has to try them one by one until he finds some results among them which fit his question. So users often feel confused or lose patience when they cannot find the correct links for their question.

At this moment, if a user wants to get some research domain information about the department of computer engineering at the website of www.gi.polymtl.ca, the computer cannot give him an exact answer quickly and directly. This procedure of information searching is far from an intelligent search where the computer is supposed to do all the work. Moreover, if the user wants to get another more specific information, like “which professor teaches INF3300 and research in Artificial Intelligence? ”, this will be very difficult to achieve. The website cannot understand such a complicated question, neither can execute such a difficult search. The user has to achieve a distinct search for each question and then combine the results by himself.

1.1 Search engines for Classical Web (CW) are limited

The following list summarizes several shortcomings of CW search engine. Solving them would become the key to broaden the use of Web technology, to improve the work efficiency, and to make full use of the web resources :

1. Because CW contains only static information composed of words or literature that machine could not read and understand, the search engines of CW work in a low automation degree. They must accomplish the job with the aid of analysis and understanding of users' queries. For example, it is the user's task to find the right link

among those “search results” returned by a search engine. In these “search results”, there may be thousands of links.

2. A Classical Web search engine could not integrate the data from different sources. That means it can only search in a certain database source, other database sources could not be integrated or involved in the searching task. The search of classical web may become very time-consuming and inefficient because we have to search and analyze data in different database sources one by one.
3. Classical Web search engines are essentially based on keyword extraction and do not achieve an intelligent search. How to get exactly the specific information we want is still a problem, because the engine can return thousands of result links for the keywords. In the query, maybe few of them are expected; there is still the possibility that some input keyword misses the relevant information because that information is represented by other keywords in the target document.

1.2 Examples of the search engine problems in CW

We can search information by Google (<http://www.google.com>) or Yahoo (<http://www.yahoo.com>) and get the website links result as our query result. Then what is the general principle of the classical website search engine?

In classical website search engine, software programs and databases can “crawl” the web by following links in pages already stored. The new web sites could be added to database by indexing program of the search engine. The database can be searched by keyword. That is the reason why search engines always give us keyword or keyword alike result. When we search something with the classical website search engine, actually we are searching the stored pages in its database. Using the most appropriate keywords and website text to search is very important, because according to the statistic

from search engine different keyword or website text can make the website rank differently.

Generally, the procedure for searching information in the CW is the following one: firstly input some keywords, then activate the search engine. After that we get a list of links which have the content related to the keywords. But this kind of search engine is not as efficient and effective as we expected. It is obvious that a classical web search engine could not efficiently automate the search process, since the search engine does not understand the semantic of the document. This search engine could not give us directly the right answer and it could not accomplish all the search without users' checking and filtering. Among the links returned by the search engine, there must be a lot of links that are useless and irrelevant. That means users still have to filter and extract the result links themselves after the search engine has done its searching job and the results have been given. So it is an ambiguous search where the computer does not know the exact concept meaning of the keywords, and it is also a time consuming process.

The main disadvantage of a keyword-based search is that it does not know the meaning of the words. For example, in Classical Web, let us suppose I want to identify "the doctor whose first name is John?". We should input a keyword "doctor" as the search keyword. But when I input "doctor", it may mean the "medical doctor", it may mean the academic degree "doctor", and maybe it is a brand of something. So the search engine does not know exactly which meaning of word to search. It cannot accomplish a more complicated search, either. For example, if John wants to leave Montreal for Toronto, by bus, by car, or by plane, and he is not sure which one he should choose, he will probably query some information about how he can get there and when he can get there from the search engine. He can separately input keywords "bus", "car", "plane", then filter the search result links to select the exact one according to this question. Three keywords

searches and he still needs to spend more time on extracting the searching results. If he inputs the three keywords “bus”, “car”, “plane” together, the search engine returns some links that contain these three words together or separate. He needs to check among those links whether there is the expected information or not, because there are many circumstances and possibilities in which these three words appear in one website or documents but has no information about “when and how he can go to Toronto”. But in Semantic Web computer can distinguish the concept meaning of every word and give the user exact answer of what he wants to know.

In CW, Web data from different resources cannot be integrated together, because the data have no real meaning, they are just characters and text and have no relation with each other. For example, every website has its own database and this database can only be used by this website. The classical website databases are isolated from each other; there is not one international database which can contain all the information of different data resources which are available to all existing websites. Google (www.google.com) has a powerful searching function, since it has many powerful servers scanning the information from all over the world on the internet. It can create its own database, but compared with the whole information resources on internet it is still very limited. In a query of CW, people are not able to search more than its own database resource. If people are not satisfied with the searching results, they have to use another search engine since search engines have different website resource in their database, maybe one of them can return the answer correspond to user's expectation.

Distributed databases is a database that is stored on different computers in different locations connected by a client-server network. It is aiming at sharing the data. One of the problems of distributed database is that the quality of information derived from numerous distributed databases is relevant with their synchronization. If one of these distributed databases contains incomplete or outdated information relative to other databases, the quality of the whole database would badly affected. Another problem is

different definition of indexes among member database cause the information inconsistency. In distributed database, the different databases can be accessed simultaneously and the data of distributed database is dynamic. These several databases construct one database group, actually the databases can be united together only in this group, on the internet this database group is still isolated from other database. Distributed Databases cannot offer us a practical solution for the database sharing in a large scale.

Considering the problems above, we can imagine that if there were another kind of search engine which can give us explicit and intelligent answer about our questions and related information, and the users need not participate too much in the search process, then it may solve the problems and satisfy our requirement. That is exactly what we hope to achieve.

1.3 Semantic Web (SW) could probably solve the problem

Actually the technology of Semantic Web (SW) could overcome the limits of the search engines of CW. What is the Semantic Web? Just like Tim Berners-Lee says, "the Semantic Web approach develops languages for expressing information in a machine processable form" [1] . Semantic Web is an extension of the current Web in which information is given well-defined meaning, enabling programs to understand it.

The Semantic Web is the Web in which the data is understandable to both human and machine. It makes the classical "web of links" replaced by a "web of meaning", where the web resources are described in an explicit language. Semantic Web is not a Web of documents or texts, but a Web of knowledge and concepts that are shared by different resources.

To let the computer accomplish most of the applications of the website and automate the applications on the web, we need to express the data in a machine-readable format. Then the machine deals with all the data without the aid of human. This can be achieved by adding more metadata in the Web data, Metadata is “data about data”. In the context of SW, it could be used to add meaning to the existent data. One interesting tool that can be used to add the metadata is the Resource Description Framework (RDF) [2]. It is a data model that has been created especially for the SW. Based on XML, it constitutes a simple way to express some information about resource that are part of the web. “Resource Description Framework (RDF) is thought to enable automated processing of Web resources. Research in RDF has identified a variety of application areas. In resource discovery, RDF may lead to better and more efficient search engines. RDF may facilitate the description of the content and content relationships available at a particular Web site or digital library, facilitate knowledge sharing and exchange, and many other appealing possibilities” (Berners-Lee , 1998) [3]

The requirement of machine-interpretable information led to the creation of languages that are necessary for us to set up an ontology [4] for the SW. Ontologies are used to share information on a specific subject or some knowledge domain, Ontologies define the computer understandable basic concepts and relationships among them which are used to describe and represent an area of knowledge. Ontologies can also make that knowledge reusable. Among the knowledge representation languages, there are RDF, RDFS(RDF Schema)[5] , OWL (Web Ontology Language) [6] , DAML (DARPA Agent Markup Language) and OIL (Ontology Interface Layer) [7].

Ontologies can enhance the functioning of the Web in many ways : firstly, they help to improve the accuracy of Web searches. With them, we can search for a precise concept instead of using ambiguous keywords. Ontologies contain structured information and enable the use of inference rules. The information is readily processed by a computer

and may come from different sources. Ontologies constitute a solution for accessing data that are stored using different models or languages.

Suppose someone wishes to find some information on Professor Michel, and she does not know his second name, but she remembers that he does some research in the field of Semantic Web. The search engine of SW could sift through all the web data to find the professor whose research is Semantic Web and whose name is "Michel". After the searching process, all information about this person would be given as a result by SW search engine.

To summarize, the SW technology could bring us technology advance and help us solve a number of problems in CWs. It offers us many basic advantages:

1. It delegates many tasks that are achieved by human to computers, because in SW the web data is computer understandable, and computer can process those data automatically. Based on this SW search engines will be quicker, more accurate and more intelligent.
2. Different data resources could have very complicated relationship with each other, all of the resources can be integrated together within ontology. In the search engines of SW, user can ask complicated questions which covers wide information and get the right answer immediately.
3. Since the web data of SW is open, well-defined and has the same public standard, any other search engine of SW can reuse the web data, so the SW data can be shared by others.
4. Computer and users also could easily work in a good cooperation. Because in SW computer can understand the real meaning of commands or input word from the user, it can execute the user's command exactly as user requires.

1.4 My research project and organization of thesis

My research project is based on the technology of ontologies and Semantic Web. I set up an ontology by modeling the information of the computer engineering department including information about the professors, courses and researches. Then I built a search engine in which different ontologies can be loaded. It is a powerful semantic web search agent. Users can search information of the computer engineering department conveniently. For example, an user can easily get the answer of following questions. What are the available courses or research resources in the department? Who in the department has some knowledge on some topic? Compared with classical search engines, it has several advantages which I will introduce in chapter 4.

In the chapter 2 of this thesis, I will explain what is an ontology and how an ontology can be represented in SW. I will also present the OWL language, which is the official recommendation of W3C for ontology representation. In Chapter 3, I will briefly describe the Semantic Web and present its main technologies. In Chapter 4, I present an architecture of a semantic web site for the Computer Engineering Department of Ecole Polytechnique. More specifically, I will present the ontology I have developed and a new search engine that have been designed especially for the SW. I will show that it has some interesting characteristics not found in the search engine available at this moment. Finally, in Chapter 5, I do some evaluation of this search engine and compare it with other SW search engine. In Chapter 6, I give some conclusions and identify interesting future works.

CHAPTER 2

Ontologies

The aim of Semantic Web is to add machine understandable semantics into the World Wide Web (WWW) by using an ontology technology to define and organize the web data. In this chapter I will discuss ontologies, including ontology definition, modeling ontology, description languages, reusability of ontology and ontology develop tools. I will also introduce OWL, an ontology language created by W3C (world wide web consortium) with an detailed example of ontology written in this language.

2.1 Definition of ontology

According to the definition of Stanford Knowledge Systems Lab, " an ontology is an explicit specification of some topic. It is a formal and declarative representation which includes the vocabulary (or names) for referring to the terms in that subject area and the logical statements that describe what the terms are, how they are related to each other, and how they can or cannot be related to each other. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some topic and a set of relationships that hold among the terms in that vocabulary." [4]

Ontology is a semantic foundation which is used to conceptualize different objects such as things, events, and relations in a certain (specific or common) domain and also to provide computer with machine-readable knowledge to process. An Ontology abstracts the knowledge of some domain, offers a common understanding about this knowledge

by making explicit vocabulary, and gives a formalized definition of the terminology and vocabulary.

On the symbolic level, an ontology is the representation of concepts and logical theory, it is the vocabulary used by a logical theory and also the specification of a logical theory. At the knowledge level, an ontology is a formal, explicit specification of a shared conceptualization. The formalism makes an ontology machine understandable as the explicit specification encompasses the concepts, properties of concepts, relations, constraints, and axioms in the target domain.

Ontologies can be highly informal if they are expressed in natural language; semi-informal if expressed in a restricted and structured form of natural language; semi-formal if expressed in an artificial and formally defined languages; and rigorously formal if they provide meticulously defined terms with formal semantics. A highly informal ontology would not be an ontology since it is not machine-readable.

There is an important concept is ontology commitment, it guarantee consistency, but not completeness of an ontology, and this involves making as few claims as possible about the world being modeled, this allows the participating parties freedom to specialize and instantiate the ontology as necessary.

An ontology consists of concept definitions and the constraints on these concepts, and could be shared by other softwares or application systems. The concepts and constraints are formal so as to realize the purpose of machine readability and knowledge sharing. Generally, an ontology has the following four important properties:

Conceptualization : We get the ontology model by abstracting the phenomenon of objective world, though its content is still independent from the environment reality.

Explicit : The concepts and the constraints have explicit definition in an ontology.

Formal : There are formal rules and common standards for the definition of ontology. That makes it machine readable.

Sharing : Ontology reflects some common knowledge that could be shared by any software or application system committed to it.

2.2 Ontology Modeling [8]

The aim of an ontology is modeling some reality of the world. Every term in an ontology corresponds to an entity. The meaning of the term may be given by mapping to another formalism (such as FOL-First Order Logic [9]). Thus the ontology languages have their own well-defined semantics or Model Theory (MT) [10]. A Model Theory defines relationships between syntax and interpretations. There may be many interpretations of one piece of syntax. Models are supposed to be analogue of reality. For example, elements of model correspond to objects in world. The relationship between syntax and models and structure of models reflects the specified relationships in syntax. Inference is defined in terms of MT. There are two popular ontology frameworks that are used in knowledge representation: the Frame Oriented Knowledge Representation[11] and Description Logic [12].

2.2.1 Frame Oriented Knowledge Representation

Frame-based systems rely on a data structure called "frame". Frames are often used as a popular method of implementing knowledge base systems. The general format of a frame is the following:

(<frame-name> (<slot1> (<facet1> (<value1> <value2> . . .)))

```

                (<facet2> (<value1> <value2> . ...))
                                ... )
    (<slot2> (<facet1> (<value1> <value2> . ...))
              (<facet2> (<value1> <value2> . ...))
                                ... )
    .....
)

```

An example frame that represents the details of a person named 'Joe' is the following:

```

(Joe (isa (value (man)))
      (hobbies (value (skiing fishing)))
      (height (default (6")))
      (we (if-needed (-today birth-date)))
)

```

A frame contains a number of slots that are filled with the descriptions of the object that it represents. A facet indicates what kind of information is filled in the slot that it is associated with. There are a variety of facets used in frame-based representations. A value facet contains actual values of the property described by the slot (e.g. skiing is a value of facet hobby). Default facets are used to indicate values of a slot when the actual value of the property described is not known (e.g. height). In addition to actual and default values, facets also permit the possibility of attaching procedures to a slot in a frame. Such procedures are known as demons. An *if-needed* demon is a procedure that computes the value of the property when the slot value is retrieved (e.g. age). Similarly, the demons *if-added* and *if-removed* monitor the slot to which they are attached and act when a value is added or removed from the slot. *If-added* and *if-removed* demons are useful in enforcing integrity constraints during updating of properties.

Frame : it can be a Class Frame (entity), or an Individual Frame (instance).

Slots : slots are relationships (sets of characteristics or data elements) of Frames. A frame may have many slots, but a slot can belong to only one Frame.

Slot Values : slot values are assertions about the relationship, and a value may be another frame.

Facets: domains that apply to slot values, and domains could be other frames.

A slot can be attached to a frame in one of two ways: as a *template slot* or as an *own slot*. An *own slot* attached to a frame describes properties of an object represented by that frame (an individual or a class). *Own slots* attached to a class do not get inherited by the class' subclasses or propagated to its instances. *Template slots* can be attached only to class frames. A *template slot* describes properties of the class' instances. A *template slot* attached to a class is inherited by its subclasses. In addition, a *template slot* of a class becomes an *own slot* of the instances of that class.

Generally, in a frame oriented knowledge representation, there are classes, relations, instances, functions and axioms. In the following introduction examples I selected KIF as the knowledge representation language.

Classes A class is a description of objects in natural language. It could be the name of an object or concept, or a relation between objects or concepts. For example, the relations "subclass" means that some class is the subclass of another. A subclass can have several superclasses. It possibly has many specific properties, restrictions, or relationships inherited from other classes. They can be different from the ones of its superclasses. Here is an Example:

```
(define-class Travel (? travel)
: axiom-def
(and (Superclass-Of Travel Flight)
(Subclass-Of Travel Thing)
(Template-Facet-Value Cardianlity arrivalDate Travel 1)
(Template-Facet-Value Cardianlity departureDate Travel 1)
:iff-def
(and (arriavlDate ?travel Date)
(departureDate ?travel Date))
```

In this example, *Travel* is the *Superclass-Of Flight*, it is the *Subclass-Of Thing*. It has an unique *arrivalDate* and *departureDate*. *Iff-def* is for defining that *arrivalDate* and *departureDate* are the necessary and sufficient conditions for being a trip occurs.

Properties (Slots) Properties describe the attributes of classes. They can be used to describe relationships between classes. A property's range lists the classes that are allowed to fill in the property value, Cardinality is the number of property values that a property may have.

Here is an example to model a binary relation *arrivalPlace*, where we specify that it is a relation between a travel and a location. In this example *:def* is to define that *travel* and *location* are necessary condition for an *arrivalPlace*:

```
(define-relation arrivalPlace (?travel ?arrivePlace)
: def
( and ( Travel ? travel)
      (Location ? arrivePlace))
```

Functions A function is a certain relation between the elements, such that one entity is related to exactly one another entity. Here is an example which obtains the price of a room after applying a discount, the relations between input entities and output entity is expressed within *body*:

```
(define-function Pays (?room ?discount) : -> ?finalPrice
: def (and (Room ? room) ( Number ? discount)
          ( Number ?finalPrice)
          ( Price ?room ?price )
: body
      (-?price (/ (* ?price ?discount) 100)))
```

Axioms. Axiom is the statement part of an ontology, It is an ontological assumption that cannot be described using only properties and property values.

Axioms can describe classes as “disjoint”, “subclass of”, or “same class as”. The consistency of axiom can be checked with a reasoner. A pair of disjoint classes has no instances in common; The subClassOf axiom describes one combination of classes and properties; the sameClassAs axiom sets one combination of classes and properties equivalent to other classes or classes combination and properties.

Here is an example for the axiom in travelling domain representing the fact that it is not possible to travel from the USA to Europe by train.

```
(define-axiom No-Train-between-USA-and Europe
:= (forall (?travel)
    (forall (?city1)
        (forall (?city2)
            (=> (and (Travel ?travel)
                    (arrivalPlace ?travel ?city1)
                    (departurePlace ?travel ?city2)
                    (or (and (EuropeanLocation ?city1)
                            (USALocation ?city2))
                        (and (EuropeanLocation ?city2)
                            (USALocation ?city1))))
                (not (TrainTravel ?travel ))))))))
```

Instances. Instances are objects which are abstracted from the reality into ontology, in other word they are individual manifestations of a class. An instance of a subclass is also an instance of that subclass’s superclasses. Once a class instance is declared, properties associated with this class are also filled in with specific properties values.

Here is an example of instance of the concept AA7462 which is an occurrence of flight AA7462 that arrives at Montreal on February 8 of 2004, and whose cost is 300 dollars.

```
(define-instance AA7462-Feb-08-2004 (AA7462)
: def (( singleFare AA7462-Feb-08-2004 300)
      (departureDate AA7462-Feb-08-2004 Feb-08-2004)
      (arrivalPlace AA7462-Feb-08-2004 Montreal)))
```

Its Frame oriented object structure is nice but its semantics is not very clear. In particular, there is confusion between assertions and descriptions. That is the weakness of Frame oriented knowledge representation.

2.2.2 Knowledge Representation in Description Logic(DL)

Description Logic (DL) is a family of knowledge-representation formalisms. The knowledge base in Description Logic(DL) provides a precise characterization of the type of the knowledge. One of its components Tbox, is the terminological knowledge (concepts and roles). Roles represent relations between pairs of individuals. Its other component ABox is for assertional Knowledge (facts). It is object-centered and has roles, features and attributes. DL also has inference services and other properties: guaranteed correctness, completeness, decidability and good complexity properties.

Guaranteed correctness means the conclusion we infer is a logical consequence of the statements contained in the knowledge base. For example, let us suppose the knowledge base states that every human has exactly one mother who is a woman, and Mary is the mother of John. It is correct to infer that Mary is a woman, incorrect to infer she is a man. Description Logic guarantees that we will never deduce such incorrect conclusion .

Completeness means that if some fact is a logical consequence of the statement contained in the knowledge base, the inference engine will be able to infer it. Returning to our example, it would be able to infer that Mary is a woman, because this is a logical consequence of the asserted facts.

Decidability means that there is some inference procedure that will always be able to verify whether some fact is a logical consequence of the statements contained in the knowledge base.

Finally, good complexity properties means that inference can be achieved in reasonable time.

Description Logic (DL) has the following characteristics : a knowledge base is modeled as a pair (Terminological information, Assertional information). Terminological information is to constructs in concept and role expressions, it is a set of definitions of the basic and derived notions and concept of the ways they are inter-related. This information is “global,” being true in every model of the situation and of every individual in the situation. Assertional information is assertions in the terminological information and assertional information, which records “specific” or “local” information, being true of certain particular individuals in the situation. There is inference mechanisms for reasoning on both these two information.

DL comes equipped with a very expressive language and allows the most general form of TBox assertions. DL takes into account instance assertions on both concepts and roles in the ABox.

Tbox

A TBox stores the conceptual knowledge of an application domain. Because it defines the knowledge in the form of a terminology it has the term “TBox”. The terminology consists of concepts, which has individuals and their relations. The DL model systems can build both atomic and complex concepts and relations. Different DL systems are distinguished by the description language which is used for building complex concepts and relations.

Here is an example of a Tbox of a family definition:

*(signature :atomic-concepts (person human female male woman man parent mother
father grandmother)
:roles ((has-child :parent has-descendant*

```

        :domain parent
        :range person)
    (has-descendant :transitive t)
    (has-gender :feature t))
: individuals (alice betty charles doris eve))
)

```

the code above shows *signature* of Tbox, *signature* helps avoiding typos. *Atomic concepts* are the names of primitive (undefined) concepts, signature of *roles* includes *has-child*, *has-descendant* and *has-gender* and signature of *individuals* has several names of persons. Continue with the code above, the following lines are the concepts definition of an Axiom, where *roles* *has-child*, *has-desendant* and *has-gender* as we constraint.

```

(implies person (and human (some has-gender (or female male))))
    ; person is the human who has the gender of female or male.
(disjoint female male)
    ; female and male are disjoint.
(implies woman (and person (some has-gender female)))
    ; woman are the person who has female gender.
(implies man (and person (some has-gender male)))
    ; man are the person who has male gender.
(equivalent parent (and person (some has-child person)))
    ; parent are the persons who has child.
(equivalent mother (and woman parent))
    ; mother is the woman in the parent.
(equivalent father (and man parent))
    ; father is the man in the parent.
(equivalent grandmother (and mother (some has-child (some has-child person))))
    ; grandmother is the mother of the person who has child.

```

Just as the above, the :some operator guarantees that there will be at least one value for the attribute. Concepts represent classes, i.e., sets of individuals.

Abox

An ABox contains extensional knowledge about the domain. Whereas TBoxes restrict the set of possible words, ABoxes describe specific objects by introducing individuals (or instances) together with their properties. In the Abox, knowledge can be seen as concept assertions. Assertions are individual or particular members of concepts. ABox can infer facts about ABox individuals. ABox consistency checking is in general more complicated than Tbox consistency checking. If ABox is consistent, there exists a "model" for both ABox and Tbox, and all ABox inferences are based on the ABox consistency check.

Specifically, ABox contains concept assertions and role assertions. An ABox always refers to a particular TBox. It requires unique names and the inferences are restricted, its facts are assumed to be incomplete, that is, new facts may be added under the restricted inference.

Continuing with the codes in TBox, here is an example of ABox :

<i>(instance alice mother)</i>	<i>; alice is a instance of mother</i>
<i>(related alice betty has-child)</i>	<i>; alice has-child betty</i>
<i>(related alice charles has-child)</i>	<i>; alice has-child charles</i>
<i>(instance charles father)</i>	<i>; charles is a instance of father</i>
<i>(related charles doris has-child)</i>	<i>; charles has-child doris</i>
<i>(related charles eve has-child)</i>	<i>; charles has-child eve</i>

2.2.3 Inference in DL ontology

We can do inference in the DL ontology above, in the example above there are the following people: Alice, Charles, Betty, Doris and Eve. Charles is a mother of Doris and Eve. Alice is the mother of Betty and Charles. Then according to the definition of axiom in Tbox before, we can get the inference that Alice is the grandmother of both Doris and Eve.

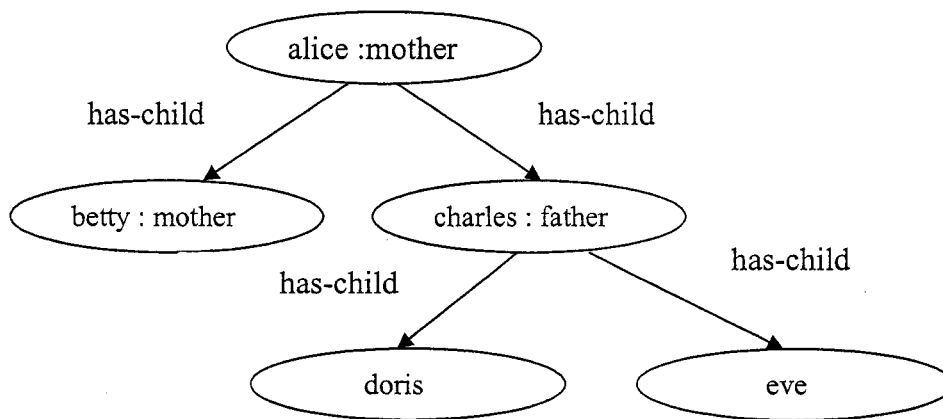


Figure 2.1 Abox Inference

2.3 Description languages

When we talk about ontology application in the field of computer engineering, naturally we should know which kind of expressive languages we should adopt to express the knowledge and setup certain ontology. Currently, the most popular description languages

are XML(S), RDF(S), DAML+OIL and OWL. Among these description languages, RDF(S), DAML+OIL and OWL could be called ontology languages.

2.3.1 XML and XML Schema

XML (eXtensible Markup Language)

XML (eXtensible Markup Language) is a metalanguage: this language is used for representing information in a standardized and structured format. XML is a human readable and machine understandable language. When we represent something with XML, we need specify the vocabulary for representing the information, because it has a certain unambiguous syntax for describing hierarchical data. But XML does not specify the data's use and semantics of elements. That means it is not an ontology language. Consider the following example:

```
<?xml version = "1.0" encoding = "UTF - 16">
<?xml:stylesheet type="text/css2" href="style.css"?>
<student>
    <name>Changshan Sun</name>
    <gender>male</gender>
</student>
```

The example above defines the concept of a student. The first clause `<?xml version="1.0"?>` specifies the standard version of XML. It continues with the definition of name, "*Changshan Sun*", and gender, "male". Normally XML provides seven different means for presenting information [13] :

Prolog : the prolog consists of an XML declaration and an optional reference to external structuring documents. Here is an example of an XML declaration:

```
<?xml version = "1.0" encoding = "UTF - 16">
```

It specifies that the current document is XML document and define the version and the character encoding used in the particular system.

Elements: Elements are items bounded by markup tags. For example:

```
<name>Changshan Sun</name>
```

Attributes: Attributes are name-value pairs defined within a tag , the following example defines the attribute of name and gender of a student:

```
<student name= "Changshan Sun" gender="male"/>
```

Comments: Comments begin with <!--and end with --> XML processors could ignore comments. For example:

```
<!-- This is a comment -->
```

Processing instructions: Processing Instructions are procedural elements in an otherwise declarative approach. For example:

```
<?xml:stylesheet type="text/css2" href="style.css"?>
```

XMLS (XML Schemas) [14]

XML Schema is a language being proposed by the W3C to describe the content and structure of document types in XML. It provides a powerful method for describing and constraining the content of XML-documents. XML Schema meets the requirements of a wide-range of data-oriented applications that will use XML. XML Schemas are used for defining the vocabulary, specify the names and attributes of elements, specify the

structure of documents but cannot specify the data's meaning. In particular, XML Schema provides the following features:

XML Syntax:

XML-Schema uses an XML syntax, which means that existing XML parsers can be used to process XML-Schema files.

Richer Data Typing:

XML Schema extends the range of primitive types from SQL and Java, such as numeric, date/time, binary, boolean, URIs, etc. Furthermore, complex types can be built from the composition of other types (either primitive or complex). XML Schema uses a single inheritance model that allows the restriction or extension of type definitions.

Support for Name Spaces:

XML Schema is namespace-aware, enabling elements with the same name to be used in different contexts. Additionally, schema types and elements can be included (or imported) from a separate XML schema using the same (or different) namespace.

Constraints:

XML Schema provides an assortment of constraint types including format-based 'pattern' constraints and uniqueness constraints, key references (foreign keys), enumerated types (value constraints), cardinality (or frequency) constraints and 'nullability'.

Other Features:

A number of other features, including anonymous type definitions, element content types, elements and attributes, annotations, groupings and the use of derived types in instance documents, are also provided by XML Schema.

For example:

```
<complexType name="WatchPerceptualProcess">
  <complexContent>
```

```

        <extension base="ns:PerceptualProcess"/>
    </complexContent>
</complexType>

```

In this example, *WatchPerceptualProcess* has *complexContent* as a *complexType*, its *complexContent* has *extension* which is expressed by *namespace*. XML schema does not allow to express multiple inheritance, only unary inheritance can be handled. This is a constraint on the modeling side which should be taken into account.

2.3.2 RDF and RDFS

RDF (Resource Description Framework)

RDF (Resource Description Framework) is a W3C (world wide web consortium) [15] technology. It is a data model for describing machine understandable semantics of information. Different applications can easily exchange the data on the web. This model is for describing interrelationships between resources, properties and values. RDF Data model consists of three object types: resources, properties and statements. A resource can be a web page, a document, and any entity that is not a web resource, but that is referred to in the web, such as people. Properties are used to describe the attributes or aspects of a resource. An RDF statement identifies a resource, a property, and value of this property for that resource. There are two kinds of entities in RDF: resources and literals. A resource is an abstract concept or concrete object of interest. Resources are named by uniform resource identifiers (URIs). Most people are familiar with one large subclass of URIs, namely uniform resource locators (URLs). The other major class of URIs is called uniform resource names (URNs). A literal is a string or fragment of datatype. Literals are used to express basic properties of resources, such as names, ages, or anything that requires a human-readable description. Literals will be enclosed in double quotation marks (").

Using resources and literals, we can express information in RDF by composing statements. An RDF statement consists of three parts: subject, predicate, and object. Because of their tripartite construction, statements are sometimes referred to as triples. RDF is a data model for representing metadata as a graph, for describing the semantics of information in a machine-accessible way.

Figure 2.2 illustrates a simple example of XML-based syntax of RDF. It presents a person whose name is Michel Gagnon, who works at the department of computer engineering of Ecole Polytechnique, and whose homepage is <http://www.dgi.polymtl.ca/profs/~michel.gagnon>

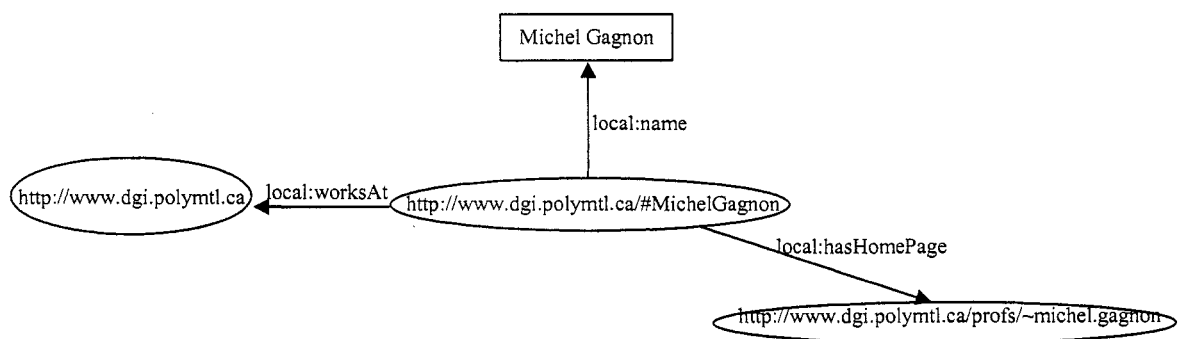


Figure 2.2 XML Based RDF

Here is a XML serialization of the RDF model of Figure 2.2:

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:local="http://www.dgi.polymtl.ca/local">
  <rdf:Description rdf:about="http://www.dgi.polymtl.ca/#MichelGagnon">
    <local:hasHomePage resource="http://www.dgi.polymtl.ca/profs/~michel.gagnon"/>
    <local:worksAt resource="http://www.dgi.polymtl.ca"/>
    <local:name>Michel Gagnon</local:name>
  </rdf:Description>
</rdf:RDF>

```

One constant in all RDF serializations is the use of the element `rdf:RDF` to wrap the RDF statements. RDF relies heavily on XML namespaces for disambiguating names. There are several elements and attribute names that must be in the namespace defined by RDF. All RDF predicates must use a namespace to clarify their meaning, as the example shows. This example uses *`rdf:Description`* tag to identify the resource that is described. Here, the “about” attribute means that it is an external resource that has already been defined. Note that all three statements have been joined in the same description element.

We expect that a resource is defined in some file, when it will be identified with a `rdf:ID` attribute, instead of `rdf:About`. According to the definition of RDF, the actual predicate is formed by joining the namespace URI and local name of the predicate element. So the full predicate of the statement is like the following form: *`http://www.dgi.polymtl.ca/local/hasHomePage`*. In this example, the resource representing the name is actually represented by the embedded Description element with a Name attribute.

To summarize, The resource " *`http://www.dgi.polymtl.ca/#MichelGagnon`* " itself is the subject of the three statements, with predicates represented by the child elements *`local:workAt`*, *`local:hasHomepage`* and *`local:name`*. The object of the last statement is a literal "Michel Gagnon". Its URI is *`http://www.dgi.polymtl.ca/profs/~michel.gagnon`*.

RDFS (RDF Schema)

RDFS (RDF Schema) is used to define further RDF vocabulary, to give external semantics to the resources. Compared with RDFS, RDF model primitives are very limited. In RDFS we can define class hierarchies and some restriction to properties [5] .

RDFS extends the RDF vocabulary with the following elements `Class`, `subClassOf`, `subPropertyOf`, `range`, `domain`. RDF provides a formalism for metadata annotation, and

a way to write it down in XML, but it does not give any special meaning to vocabulary such as `subClassOf`. Interpretation is an arbitrary binary relation, DF Schema allows us to define vocabulary terms and the relations between those terms, it gives “extra meaning” to particular RDF predicates and resources.

Classes. They are represented by `rdfs:Class` tag. We can use `rdfs:Resources` to describe , use `rdfs:property` to predicate, use `rdfs:type` to show a class instance.

Properties. They are `rdfs: type`, `rdfs:subClassOf`, and `rdfs:subPropertyOf`. The `rdfs:type` is to express the relation “type” between resources and classes. `Rdfs:subClassOf` and `rdfs:subPropertyOf` could help in subsumption(specializes general concepts in more specific concepts).

Constraints. They are `rdfs:range` and `rdfs:domain`. `rdfs: range` and `rdfs:domain` are used to restrict the range and domain.

The RDF Schema languages defines standard vocabulary for talking about RDF properties. An RDF schema is a specification of a set of RDF classes and properties. The RDFS ontology is defined within the `<http://www.w3.org/2000/01/rdf-schema#>` namespace and is commonly referred to by the `rdfs: prefix`. In RDFS language, other properties are available to improve readability. The `rdfs:label` property defines a human-readable name for a property or a class. The `rdfs:comment` property gives a human-readable description of how to use a property or a class. Finally, `rdfs:isDefinedBy` indicates the schema resource that defines the property or class in question.

We will illustrate RDFS with an example schema. For clarity reason, the properties are represented without namespace :

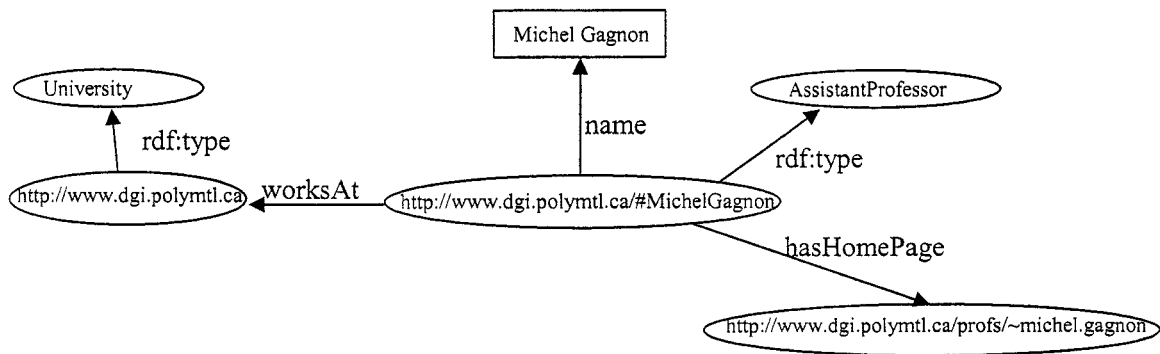


Figure 2.3 RDFS Example

Figure 2.3 shows the element already presented in Figure 2.2, but with some class information that has been added. It is still a RDF model, but it states explicitly that Michel Gagnon is an assistant professor and his work place is an university.

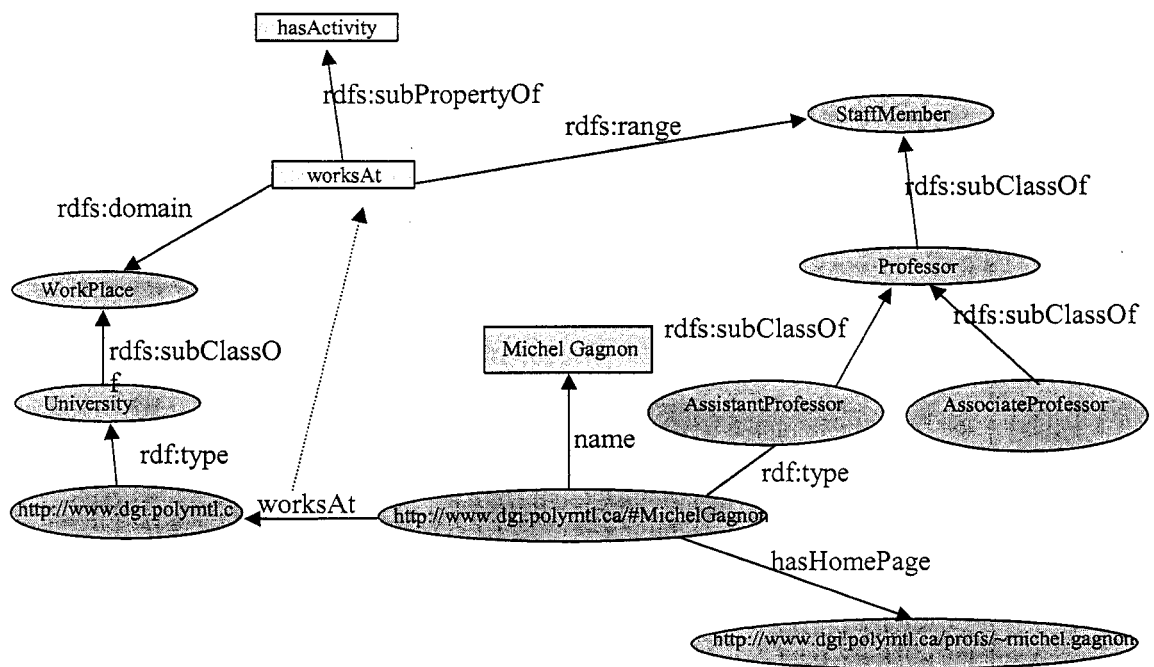


Figure 2.4 Another RDFS example

Considering Figure 2.4, we see that RDFS provides a way to specify more clearly the two classes that have been added, Michel Gagnon works at a place whose URI is

http://www.dgi.polymtl.ca, and whose type is University. Class university is the subclass of class *Workplace*. Property *WorkAt* is the *subPropertyOf* property *hasActivity*. Michel Gagnon is an *AssistantProfessor*, and class *AssistantProfessor* is the *SubClassOf* class Professor, class professor is the *subClassOf* class *StaffMember*. *StaffMemeber* worksAt the *WorkPlace*.

Let us now see how this example may be expressed in that XML serialization of RDFS:

```
<?xml version="1.0"?>
<RDF:RDF xml:lang="en"
  xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:RDFS="http://www.w3.org/2000/01/rdf-schema#">
  <RDFS:comment>A simple example schema.</RDFS:comment>
  <RDFS:Class ID="StaffMember"/>
  <RDFS:Class ID="Professor">
    <RDFS:subClassOf resource="#StaffMember"/>
    <RDFS:allowedPropertyType>
      <RDF:PropertyType ID="workAt">
        <RDFS:comment>The preposition "workAt".</RDFS:comment>
        <RDFS:range resource="# WorkPlace "/>
      </RDF:PropertyType>
    </RDFS:allowedPropertyType>
  </RDFS:Class>
  <RDFS:Class ID=" AssistantProfessor">
    <RDFS:subClassOf resource="#Professor"/>
  </RDFS:Class>
  <RDFS:Class ID=" AssociateProfessor">
    <RDFS:subClassOf resource="#Professor"/>
  </RDFS:Class>
  <RDFS:Class ID=" University">
    <RDFS:subClassOf resource="#WorkPlace"/>
  </RDFS:Class>
```

</RDF:RDF>

In this example, class *Professor* is the *subClassOf* the class *StaffMember*, it has property *workAt*, the range of *workAt* is *WorkPlace*. Class *AssistantProfessor* and *AssociateProfessor* are both the *subClassOf* *Professor*. Class *University* is the *subClassOf* *Workplace*.

2.3.3 DAML+OIL and OWL

From the introduction above, we can see RDFS has its own shortcomings. It is too weak to describe resources in sufficient detail. There is no way to explain range and domain constraints are local to some classes; For example in RDFS it is not possible to say the range of *hasChild* is a person when applied to a persons and elephant when applied to an elephant, there is no existence/cardinality constraints; it cannot say that all instances of person have a mother that is also a person, or that a person has exactly four parents. It does not point to specify that a property is transitive, inverse or symmetrical properties; For example, we cannot say that *isPartOf* is a transitive property, that *hasPart* is the inverse of *isPartOf* or that is symmetrical. It is not clear how to provide reasoning support with RDFS. Based on RDF and RDFS there are two other popular more advanced semantic web language DAML(+OIL) and OWL.

DAML+ OIL

DAML(Darpa Agent Markup Language combined Ontology Inference Layer) is an extension of RDF, adding language constructors from object- oriented and frame-based knowledge representation languages. OIL is a Web oriented ontology language which has RDFS based syntax and a set of language constructors taken from frame-based language. DAML+OIL is an merged language of DAML and OIL, and builds on RDF by supporting richer, but still decidable, modeling primitives.

In a DAML+OIL ontology there are classes, properties, including DatatypeProperty and ObjectProperty. We can take the ontology as statements (subject, predicate and object), usually we use Domain to define the domain of the subject class. Subject has property that can be treated as predicate. We use Range to specify the value or domain of the object.

OWL (Ontology Web Language)

OWL (Ontology Web Language) has been developed by W3C. Based on DAML+OIL, it also incorporates concepts from RDF/XML. It is a semantic mark-up language used to describe ontologies on the web. OWL defines two kinds of domains for properties: datatype domain and object domain, the datatype domain contains XML schema datatype values, and an object domain contains objects that belong to OWL classes. OWL also describes two types of properties. Object properties pertain to other objects, while datatype properties associate datatype values with objects. OWL is the more recent Web Ontology Language, it has three different versions with different levels of expressiveness. In increasing expressiveness, these are OWL Lite, OWL DL and OWL Full.

OWL Lite. Used by those who need only one category level and simple property constraint. Supports cardinality that can be only 0 or 1.

The following example claims the individual "http://www.polymtl.ca#ChangshanSun" is sameIndividualAs "http://www.polymtl.ca#Chang_shan_Sun".

```
<Student rdf:about="http://www.polymtl.ca#ChangshanSun">
  <owl:sameIndividualAs rdf:resource="http://www.polymtl.ca#Chang_shan_Sun"/>
</Student>
```

OWL DL. Supports the users who want the most expressive language based on reasoning

system that could guarantee the computational completeness and decidability. It includes all the constraints of OWL language, in OWL DL a property can only relate an individual with another one, a class can not be member of another class. but it is possible to use some certain constraints of them when a class is a subclass of another one.

Here is an example, shows the class *peopleAtUniversity* has the property of *unionOf*, and *unionOf* property contains a collection of two nodes elements *staffMember* and *student*, this is definition of the class by union:

```
<owl:Class rdf:ID="peopleAtUniversity">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember">
    <owl:Class rdf:about="#student">
  </owl:unionOf>
</owl:Class>
```

OWL Full. Supports those user who need free RDF expressive requirement, without the computational completeness guarantee. It allows an ontology to add words in the vocabulary. Any reasoner could not totally support all features of OWL FULL. A class could be member of another class. In OWL DL, a class cannot be made of another class. This is allowed in OWL Full, here is an example where the class *Human* is made of the class *Species*:

```
<owl:Class rdf:ID="#Species">
<owl:Class rdf:ID="#Human">
<Species rdf: ID = "#Human">
```

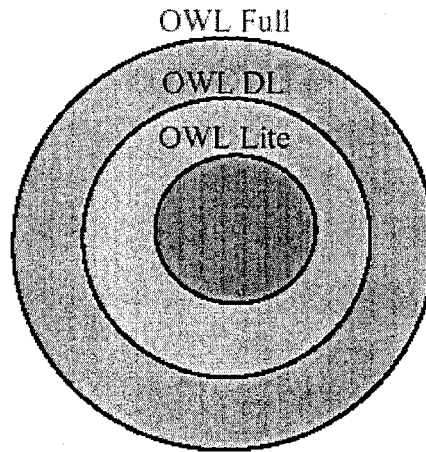


Figure 2.5 Three Versions of OWL [16]

OWL Lite is well suited for defining a classification hierarchy and simple constraints, OWL DL has maximum expressiveness while retaining computational completeness and decidability, whereas OWL Full has maximum expressiveness and syntactic freedom with no computational guarantee. In the three species of OWL, OWL Full is union of OWL syntax and RDF; OWL DL is restricted to FOL fragment (DAML+OIL); OWL Lite is to take an “easier to implement” subset of description logic than OWL DL .

In semantic layering there is OWL DL and OWL full within DL fragment. OWL DL semantics are officially definitive; OWL DL based on SHIQ [17] Description Logic; OWL DL benefits from well defined semantics, formal properties well understood (complexity, decidability), known reasoning algorithms, implemented systems (highly optimised).

OWL language has substituted DAML +OIL. OWL has world-wide support. Tools exist that do inferencing with OWL. In the next section, I give a concrete example of OWL ontology.

2.3.4 OWL Ontology Example

In my research project, an ontology is created for the management of the teaching and research work of Computer Engineering Department. It is based on the information of professors, courses, researches and other related information. In this Ontology, I use ACM Computing Classification System [18] for the definition of concepts used to index the department documents. I selected Protégé 2.1 as the tool to create this Ontology.

In the following I will explain concretely the construct of this OWL ontology .

Namespace

The beginning of an OWL Ontology is a `rdf:RDF` tag, in which the namespaces used in the ontology are defined.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
  xmlns:owl=http://www.w3.org/2002/07/owl#
  xmlns:daml=http://www.daml.org/2001/03/daml+oil#
  xmlns:rss="http://purl.org/rss/1.0/"
</rdf:RDF>
```

In this fragment, `xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#` defines the Namespace of prefix RDFS, it is to use the vocabulary of RDFS. `xmlns:owl=http://www.w3.org/2002/07/owl#` defines the Namespace of prefix OWL, this is to use the vocabulary of OWL. `xmlns:daml=http://www.daml.org/2001/03/daml+oil#` defines the Namespace of prefix DAML, this is to use the vocabulary of DAML.

We can also define some entities in DocType before we define the ontology. For example:

```
<!DOCTYPE owl [ <!ENTITY professor "http://www.dgi.polymtl.ca/professor#" >
<!ENTITY course " http://www.dgi.polymtl.ca/course#" > ]>
```

Ontology head

```
<owl:Ontology rdf:about="http://www.dgi.polymtl.ca/finala.owl">
  <rdfs:comment>OWL ontology of academic department </rdfs:comment>
  <owl:versionInfo>
    $Id: Overview.html,v 1.2 2004/08/08 16:42:25 connolly Exp $
  </owl:versionInfo>
  <owl:imports rdf:resource="http://www.dgi.polymtl.ca/finala.owl"/>
</owl:Ontology>
```

the first line `<owl:Ontology rdf:about="http://www.dgi.polymtl.ca/finala.owl">` defines the main function of ontology, the second line give the comment or explanation for this ontology. Then continue to define the version information of this ontology. After that, `<owl:imports rdf:resource="http://www.dgi.polymtl.ca/finala.owl"/>` is to offer import mechanism with `"http://www.dgi.polymtl.ca/finala.owl"`. At last it is the indicator of end `</owl:Ontology>`.

Simple Classes and individuals

Every class defined by the user is default subclass of owl:Thing. For example, the following is an example for definition of simple classes:

```
<owl:Ontology rdf:about="">
  <owl:Class rdf:ID="computer_systems_organization">
```

```

<rdfs:subClassOf>
  <owl:Class rdf:ID="knowledge_category"/>
</rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Software">
  <rdfs:subClassOf rdf:resource="# knowledge_category "/>
  <rdfs:label xml:lang="en">software</rdfs:label>
  <rdfs:label xml:lang="fr">logiciel</rdfs:label>
</owl:Class>

```

In this fragment, `rdf:ID` is to give the class a name, then this class can be quote in other ontology or in this ontology. *SubClassOf* is used to define one class is the subclass of another. `rdfs:label` offer a readable class name and `xml:lang` express in multi-languages. In this example, *computer_systems_organization* is the subClass of *knowledge_category*, class *Software* is also the subClass of *knowledge_category*. Class *Software* has two labels, one is in language English 'software', the other is in language French 'logiciel'.

The relationship can be described by the figure 2.6.

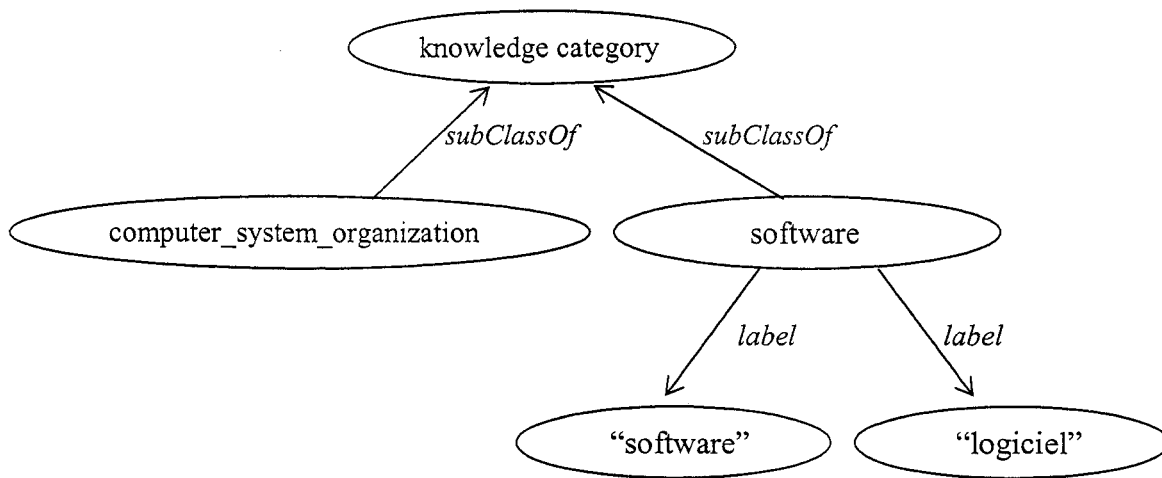


Figure 2.6 Example of Simple class definition

Individuals

The following fragment define an individual which is an instance of “professor” class and where ID is “Michel_Dagenais”, furthermore, there is a property where stating that this individual teaches another class individual name “algorithm_design_and_analysis”, this is to show an individual can be expressed by declaring it is the member of a class.

```

<professor rdf:ID="Michel_Dagenais">
  <teach rdf:resource="#algorithm_design_and_analysis"/>
</teach>
</professor>

```

Simple Property

In OWL, to give restriction on some property, we first define an anonymous class by specifying these restrictions, and then state the property as a subClass of this anonymous

class. Properties can be defined for every instance of a class, or for some specific individual. There are two kinds of properties:

datatypeproperty: relation of class elements with XML datatype;

objectproperty: relation between two class instances.

The definition of property also has domain and range. In the following fragment which extracted from our department ontology, it describes the *class Professor* with *objectproperty* and *Datatypeproperty*, in the properties of *Professor*, we used RDF Schema for constraining range and domain :

```
<!-- ****Professor**** -->
<owl:Class rdf:ID="Professor">
</owl:Class>
<owl:DatatypeProperty rdf:ID="name">
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="title">
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasEducationDegree">
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range rdf:resource="#Degree"/>
  <owl:minCardinality>1</owl:minCardinality>
  <owl:maxCardinality>n</owl:maxCardinality>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasOfficeAddress">
  <rdfs:domain rdf:resource="#Professor" />
```

```

    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasTelephoneNumber">
    <rdfs:domain rdf:resource="#Professor" />
    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasEmailAddress">
    <rdfs:domain rdf:resource="#Professor" />
    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="hasResearchArea">
    <rdfs:domain rdf:resource="#Professor"/>
    <rdfs:range rdf:resource="#Research"/>
    <owl:minCardinality>0</owl:minCardinality>
    <owl:maxCardinality>n</owl:maxCardinality>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="teachCourse">
    <rdfs:domain rdf:resource="#Professor"/>
    <rdfs:range rdf:resource="#Course"/>
    <owl:minCardinality>1</owl:minCardinality>
    <owl:maxCardinality>n</owl:maxCardinality>
  </owl:ObjectProperty>
</rdf:RDF>

```

In this ontology, class *Professor* has various properties of type *DatatypeProperty*. For some of them, the range type is a string: name, title, hasOfficeAddress, hasTelephoneNumber, hasEmailAddress. It also has *ObjectProperty*, for example the property *hasEducationDegree* relates individual of class *Professor* to one or more individual of class *Degree*. Class

Professor also has the *ObjectProperty* of *hasResearchArea* and *teachCourse*. *Professor* is the domain of all the *DatatypeProperty* and *ObjectProperty*.

2.3.5 Relation of Description Languages

I will now summarize the description languages presented in this chapter, and give a introduction for each level:

	Level	Name	Description
	Level 1	UNICODE and URI	The base of the whole semantic web . Unicode for processing the resource,URI for identification of resource.
	Level 2	XML+NameSpace +xmlschema	For expressing the Data content and Data construct.
	Level 3	RDF+rdfschema	Used to describe the resource of the web and their type.
	Level 4	Ontology vocabulary	Used to describe the relations between all resources.
	Level 5	Logic	Based on level 1 to level 4, it has logic and reasoning function.
	Level 6	Proof	
	Level 7	Trust	

Figure 2.7 Ontology Language Level

XML provides a syntax transport layer, RDF provides a basic relational language, RDFS provides basic ontological primitives, and OWL provides a (decidable) logical layer. OWL has substituted DAML+OIL. But DAML+OIL and OWL are at the same level.

2.4 Reusability of Ontologies

The successful use of ontology in Semantic Web has enhanced the development of ontologies. People put many efforts on domain modeling in the field of both software engineering and ontology engineering. Ontologies may become very important and necessary in general software systems. Many mature examples of ontologies can be found in the domain of business integrated e-business, in information retrieval systems, digital libraries, and natural language processing.

Some example of ontologies:

CYC: A general ontology for common sense knowledge to facilitate reasoning.

Address: www.cyc.com.

UMLS(Unified Medical Language System): An ontology of medical concepts.

Address: www.nlm.nih.gov/research/umls

WORDNET : Most well-developed lexical ontology.

Address: www.cogsci.princeton.edu/~wn

FRAMENET : English dictionary. Use Frame Semantics for description frame, with strong semantic analysis capacity .

Address: <http://www.icsi.berkeley.edu/~framenet/>

Reusability of Ontologies:

These ontologies can be used by other application programs. Developers can define their own ontology by extending these standard ontologies. These ontologies are excellent resources for us to share, to learn from them and to find their shortcomings. If it is possible to use in our ontology we just use it or put it as a resource in our own ontology. It is very convenient and can save us a lot of time.

In my research subject, unfortunately, I could not find the related ontology on the Internet, so there is no ontology available for reusability. We have to create it from scratch. But we can imagine in the future, if somebody needs an ontology about an academic department of university, he can take our ontology and reuse it if he thinks it is good. With the technology of ontology and SW becoming more and more popular, there will be plenty of ontologies on internet which will be available for us to reuse.

2.5 Ontology Develop Tools

2.5.1 Protégé with OWL plugin

In this section, I will introduce Protégé, the most popular free ontology development software. It is used for the creation and definition of an Ontology. It also has some specifications that user make it easy to use. Right now the new edition of Protégé is Protégé 2.1.

It is also an open source software and convenient for developer to use and do references with its code. Protégé is a tool which allows a user to construct an ontology, customizing data entry forms and entering data. The tool can be easily extended to access other knowledge base embedded applications. For example, graphical widgets can be added for tables and diagrams. Protégé can also be used by other applications to access the data. The main assumption of Protégé is that knowledge-based systems are usually very expensive to build and maintain.

Protégé can import and export the file in format of XML, RDF(S) and XML Schema, after installing a plug-in. It also supports DAML+OIL and OWL. Via plug-ins (PAL and FaCT) Protégé can make Consistency check [19].

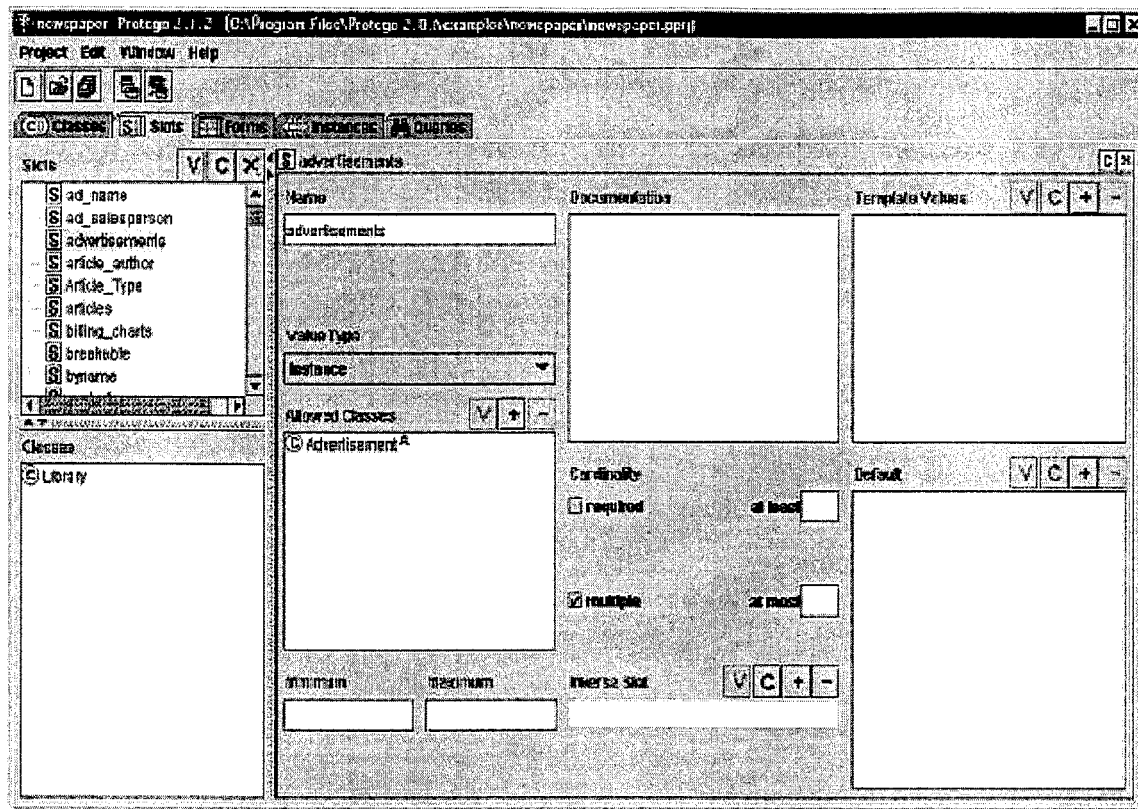


Figure 2.8 Snapshot of Protégé 2.1

2.5.2 Other Ontology tools

OntoEdit

OntoEdit is another tool for doing the ontology development or ontology maintenance. It supports ontology modeling, easy for users to create or revise concepts, relations and axioms. Its structures contains several graphical views which are used to create ontology, refine ontology and evaluate ontology.

The ontology engineering environment OntoEdit uses a powerful ontology model to store an ontology. With OntoEdit users can transform the conceptual representation in all

major ontology representation languages including RDF(S), XML, DAML+OIL or Frame-Logic. It can import/export format XML, RDF(S), Frame-Logic and DAML+OIL for free version , XML, RDF(S), Frame-Logic, DAML+OIL and SQL-3 (export only) for professional version, also with Graph view.

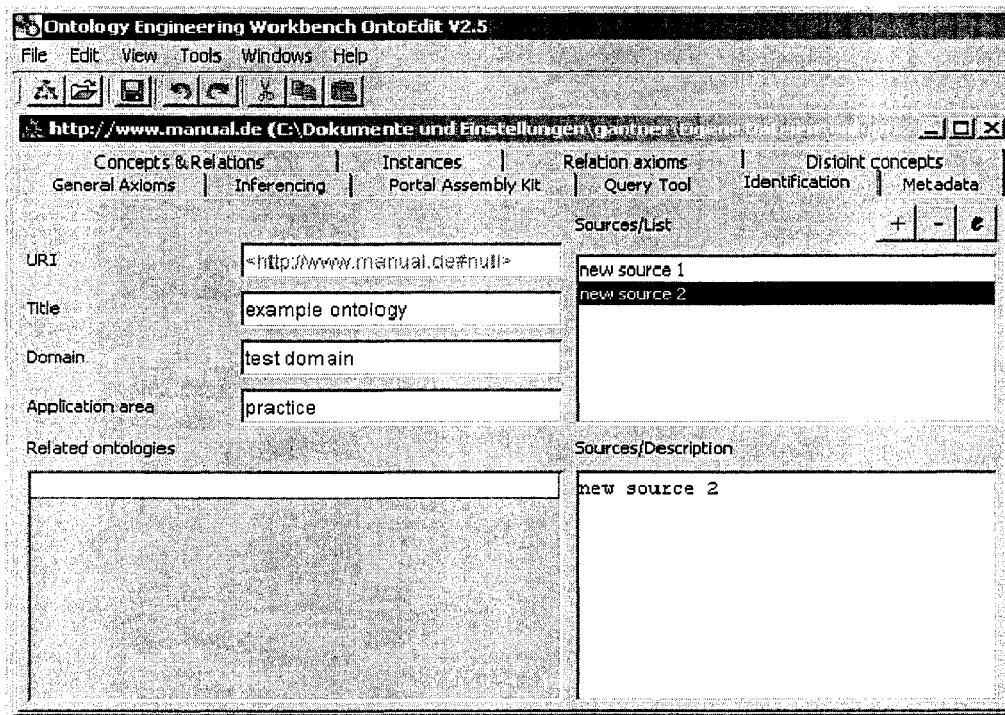


Figure 2.9 Snapshot of OntoEdit

OilEd

OilEd is a simple editor that allows the user to create and edit OIL ontologies. The main intention behind OilEd is to provide a simple, freeware editor that demonstrates the use of, and stimulates interest in DAML+OIL. It imports format RDF(S), OIL and DAML+OIL, export format RDF(S), OIL, DAML+OIL, SHIQ, dotted and HTML. It does not support graph view, consistency check via built-in FaCT.

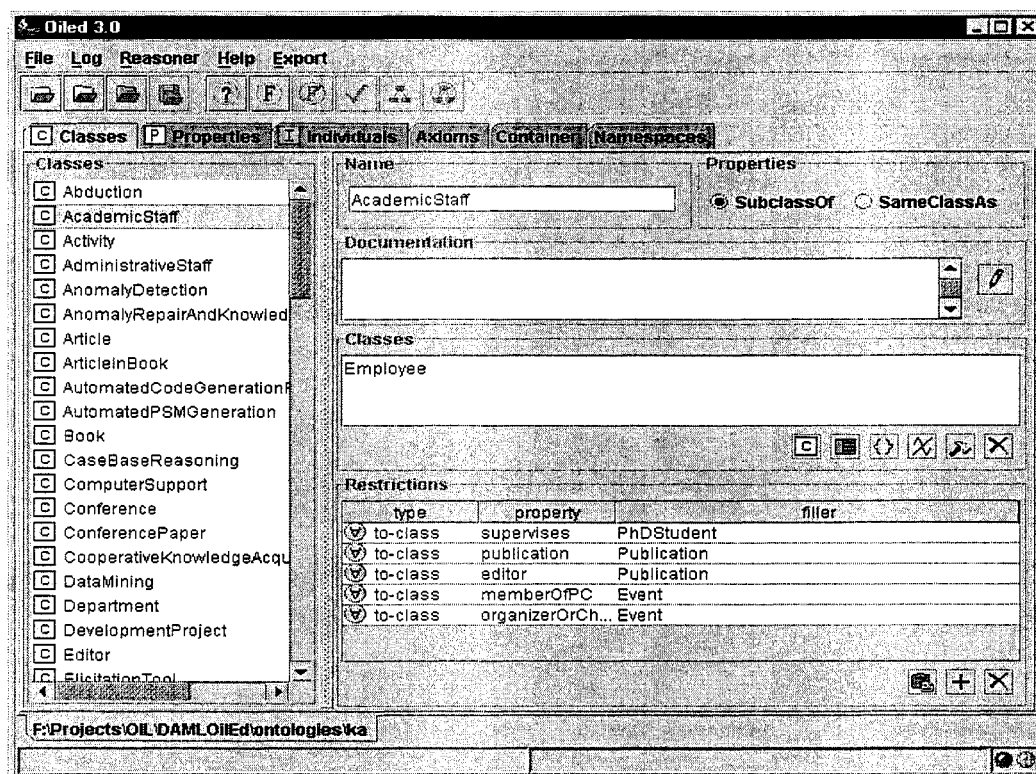


Figure 2.10 Snapshot of OilEd

CHAPTER 3

Technology of Semantic Web

As I mentioned in the first chapter, the Semantic Web (SW) is an extension of the existing Web (Classical Web, CW). It relies on some standards, vocabularies and ontologies to associate and connect many data resources which could be linked to other documents and information resources. Then computer could process the data and infer some conclusion by using Description Logic. In this chapter, I will compare CW and SW, introduce some essential technologies for approaching SW, including using ontology in SW, information retrieval and the use of ontology API, I will also introduce application of SW such as knowledge management [20] and e-learning [21], and the trust relationship between SWs. At last, I will give some examples of SW applications which are popular on internet.

3.1 From CW to SW

Currently, almost every web site has its own database which is separated from other websites. The information in the database is static, and is not machine understandable. The work of data extracting and interpreting cannot be done by computers. It is hard for users to get some exact information meaning through browsing web pages or search engine. Search engines just give us the information by using the key word occurrences in the data document and have no capacity to answer a question or infer some information from the data. SW is a solution for this problem.

Figure 3.1 gives a comparison of CW and SW.

- | | |
|--|--|
| <ul style="list-style-type: none"> • Classical Web <ul style="list-style-type: none"> — Based on HTML — Web of Documents — Keyword Search — Humans understandable but difficult for machine processing (ambiguity, unconstrained data formats) — Separated Database. | <ul style="list-style-type: none"> • Semantic Web <ul style="list-style-type: none"> — Based on XML and RDF(s) — Web of Knowledge — Concept Search — Machine understandable and information expressed in a format that is unambiguous and suitable for machine processing — A globally linked database |
|--|--|

Figure 3.1 Comparison of CW and SW

SW technology helps to restructure the information data such that computers may access the information more easily. The computer processes the data and gives users the answer to their query. Because information data of SW is organized by using metadata, the data is not only human understandable but also machine understandable. Computers could process the data and do a better and more reliable job than humans. Besides, SW allows manipulation across multiple and heterogeneous databases. This capability of SW means that different databases could be processed at the same time by one SW application.

As ontology language is not only a machine understandable language but also human understandable language, machine and human can work in a good cooperation in SW. The resource on the internet would be mined and linked to other one, because in SW the databases is a global linked database. Step by step the potentiality of the global SW network development will be achieved with the increasing number of SW users. Soon SW will be widely used in our daily life, for information retrieval, e-business, knowledge management and e-learning, etc.

3.2 Ontology Languages for Approaching SW

Ontologies, in the context of the SW, are formal specifications of the concepts of some domain. Put simply, an ontology specifies the vocabulary of a domain. The semantics on the web will completely rely on explicit ontologies, and different SW applications will communicate by sharing their ontologies. The SW gives each term or concept a globally unique URI, so a concept could be used by different application in an unambiguous way.

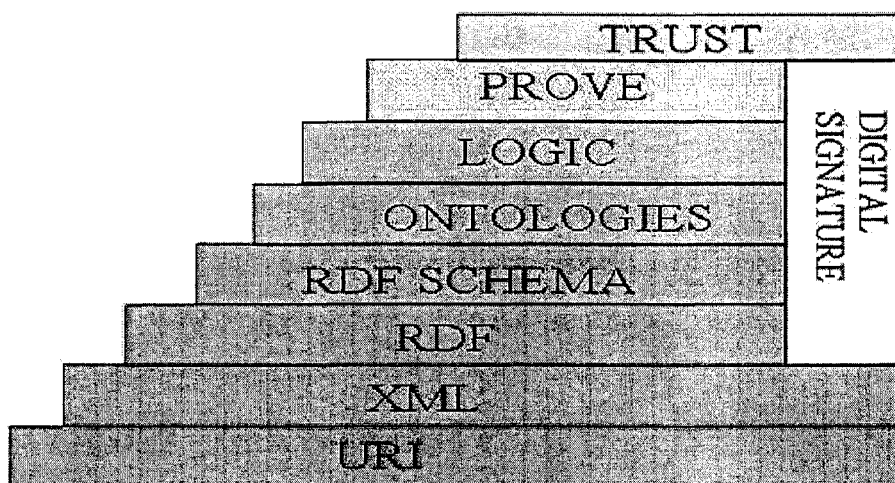


Figure 3.2 Layers of Description Languages [22]

The layered technologies of the Semantic Web, according to Tim Berners-Lee and the W3C, it is illustrated in Figure 3.2. Each layer is seen as building on—and requiring—the ones below it. The W3C has developed, or is in the process of developing, standards and recommendations for all but the top three layers. W3C recommendations for digital signatures and managing encryption keys will also play roles in the Trust layer.

XML (eXtensible Markup Language) offers a new syntax for us to structure the data. It has a tree-like data model and allows developers to add an arbitrary structure to a

document, but it does not introduce or deal with the semantics. The language framework XML has been used to define nearly all new languages that are used to interchange data over the Web. XML Schema is a language used to define the structure of specific XML languages.

RDF (Resource Description Framework) expresses the terms and concepts in a language that is understandable to computers. RDF comprises a set of triples in a way $\langle s, p, o \rangle$ where s is the subject, p is the predicate and o is the object. RDF provides a data model but not the actual meaning of the properties that are supposed to describe the data. It is a flexible language capable of describing all sorts of information and meta data.

RDF Schema (RDFS) is a framework that provides a means to specify basic vocabularies for specific RDF application languages to use. More specifically, RDF Schema is well suited for defining classes and property hierarchies. We can see it as a very primitive way of expressing ontologies. That are much more limited than ontology defined by using description logic. It is still sufficiently powerful for many application of the SW.

An ontology is a formal specification used to define vocabularies and establish the usage of words and terms in the context of some domain. OWL is an ontology language designed for the Semantic Web. DAML was the official starting point for the Web Ontology Language (OWL), an emerging standard from the World Wide Web Consortium. DAML supports the representation of ontologies (which include taxonomies of terms and semantic relations) via extensions to XML. XML alone is not sufficient for agents because it provides only syntactic interoperability that depends on implicit semantic agreements.

In Logic and Proof layers, logical reasoning is used to establish the consistency and correctness of data sets and to infer conclusions that are not explicitly stated but are

required by or consistent with a known set of data. Proofs trace or explain the steps of a logical reasoning.

In Trust layer, we should find means of providing authentication of identity and evidence of the trustworthiness of data, services, and agents.

Currently, there are several ontology languages available for defining the information: RDF(S), DAML+OIL, and OWL. OWL is recommended by W3C and used widely in building semantic website and ontologies. It has a standard vocabulary and common rules to allow the user state logical descriptions. Then the computer can infer new knowledge from the data. People can make the data more intelligent than any type of data before, because inferences and new knowledge can be derived from SW data. But before we realize this function, a reasoning engine is needed. It is used to read and parse the data of an ontology and reason with those data.

3.3 Ontologies in Semantic Web

The data of SW are based on specific ontologies. Different SW applications can communicate with each others and make full use of the information by sharing their ontologies. Ontologies are a key enabling technology for the SW and useful tools to enable flexibility, reusability and expressive communication between information resources. Ontologies play a very important role in SW. Usually the data syntax is important to a computer, because it defines the structure of strings and data in a certain language, like OWL and DAML. In SW the semantics on the other hand is to define the meaning of a string in a certain language. This meaning is understandable both by the human and the computer.

Unlike what people usually thought, SW is not artificial intelligence. It is just based on semantic description. Computers do not understand our own language, but they can give us the information context through the semantics and vocabulary. In the process of

building SW, publishing ontologies on the Internet and making them accessible for all the SW is a very important step, and how to define the concepts in a way similar to others and share them with others should be another important principle which is worthy of our attention.

Ontology and triples [8]

Ontologies are human understandable and machine processable data resource. Ontologies are documents that formally define relationships among terms in ontology languages like RDF(S), DAML and OWL. In SW, concepts such as resources, classes, properties, ranges and domains can be created to describe the meanings and relationships of terms and all the related objects. Ontology technology is capable of describing relationships between all types of things. At present time, many ontologies have already been built and shared on the internet. When we plan to build an ontology, we can learn model method, abstract skills, class and property definition from those ontologies which have been done by others. If it is exactly what we want to build and were designed, we just use it because it is open and free to everybody. We think that in the future, many ontologies should be available on the web.

One of those methods of representing semantic information in SW is by defining triples of URIs (Uniform Resource Identifier). These URIs can then be published in an ontology document and users can refer to them freely. These triples(subject, predicate, object), are much like ordinary natural language sentences. URIs identify not only information resources (such as web pages) but also indirectly refer to resources (such as items and things) that are not on the web. Providing persistent URIs for the concepts and making them available constitute an important foundation for enabling the SW.

Creating an ontology for SW

Generally there are four kinds of ontologies which can be used in SW:

1. content ontologies for reusing knowledge. These ontologies include other subcategories: task ontologies, domain ontologies and general or common ontologies.
2. Communication ontologies for sharing knowledge.
3. indexing ontologies for case retrieval
4. Meta-ontologies are equivalent to what other author refer to as a knowledge representation ontology.

To build a SW, the creation of ontologies becomes an absolutely necessary aspect. The process of creating ontologies requires efforts in developing common vocabularies which systems will use to recognize the content of a SW, but actually creating ontologies doesn't require a coordinated effort. The interoperability of SW allows words to be used differently as soon as they are related to a common concept in an ontology.

Generally, the creation of an ontology is regarded as the creation of formal specifications of the terms in a domain and the relations among them. If some ontology databases suitable for our needs already exist on the internet we can use them directly. If not, we can develop a new one by analysing the domain knowledge and taking the following steps:

1. determine the domain and scope of the ontology and enumerate important terms in the ontology;
2. define the classes and the class hierarchy;
3. define the properties of classes;
4. define the facets (also called properties) of these classes;
5. create instances of these classes.

By using this method, unstructured information can be restructured into an ontology which generates knowledge that the users can interpret unambiguously.

3.4 IR(Information Retrieval) in SW

Information Retrieval [23] technology is a kind of technology which is supposed to be an easy and accurate way for people to search some information and get the exact result immediately.

3.4.1 IR technology in CW and SW

As I mentioned above, with the application of Information Retrieval (IR) in CW, the search information is achieved through keywords. The results are given in a set of links which might be useful to users. In a word, the search in CW is based on the occurrence of words in documents.

Google is not a common CW search engine. It crawls and indexes Web pages in much the same way that the traditional search engines do, but when it puts together a list of Web pages that contain your search words, it ranks them according to their popularity or importance. So Google search does not limit in the keyword search, it use a special technology of ranking Web pages.

Google determines the popularity of a site by how many other “important” Web pages link to it. Then what are the “important Web pages”, those sites that many other Web pages link to, the pages with best information are those the other important or popular sties know about and have linked to. Google then further refines the rankings using a variety of different schemes for making the listings more relevant.

Technology LSI (Latent Semantic Indexing) means search engine tries to associate certain terms with concepts when indexing web pages. Latent semantic indexing helps search engines to find out what a web page is all about. Google has been using this concept to improve the quality of its search results in determining suitable ads for its

AdSense service (<https://www.google.com/adsense>). For example, if user do a semantic search for “phone”, Google returns “Nokia” as the first result.

But we need more sophisticated retrieval systems to deal with the fast expanding website on internet, and to help users to obtain the information conveniently and accurately. With the ontology technology, SW can make the data processable by computer. The ontology data are machine understandable and structured with semantic. The traditional web search engine will be greatly improved by SW search engine because of the addition of explicit semantics to the information data.

3.4.2 SW Agent

SW Agents[24] are applications of SW that communicate and share their data with other SW agents. A SW agent roams in the internet to collect, filter and process information data, in order to give the user some useful and relevant information. The results returned by the SW agent are quite different from the one obtained with traditional IR tools, because in a SW agent, a search typically uses the real-world concepts. The text retrieval part of the search engine can understand these concepts, and can understand the context of the search and the real meaning of the search activity. It then gives the accurate and correct information to user. A SW agent has no artificial intelligence, it is a program that uses semantics encoded within webpages and data. Rather than doing everything for a user, the agents would find possible ways to meet user needs, and offer the user choices for their achievement.

when people begin to create many programs collecting information from diverse sources, process the information and exchange the results with other programs, the power of the SW will be realized. The software agents can increase the machine-readable information and automated services.

For example, we could specify our needs, constraints and preferences to an Internet

travel agent. It would then explore some possibilities and ask our approval for its choices. Upon our agreement, the agent would download for us the data which is supposed to satisfy our requirement. The agent task will be greatly facilitated by semantic content on the Web. It will help us to avoid the increasing complexities caused by the accelerating and virtually uncontrolled growth of the WWW(world wide web).

3.5 Ontology API

When we are setting up a SW website, a “reasoner” is needed for building the new application. The reasoner could process the information resources that was built using ontology languages, and then abstract, infer and provide analysis from these information data.

Jena2.1[25] does not only function as a “reasoner” but also as a convenient and standard interface for developing an ontology-based application. Jena2.1 is a famous Semantic Web programming toolkit from HP company and is a mature API for basic RDF/OWL ontology and data processing. I will now describe it.

3.5.1 API Jena2.1

Jena2.1 is a very important tool for the onstruction of a SW. It provides all the standards which from W3C SW recommendations. With the API , an application can get and infer information. The Jena2.1 architecture has clear development definition about whatever the application programmer wish to do with the RDF graph. Jena2.1 could support ontology in RDFS, DAML+OIL and OWL. It can read, store and parse them; I/O is provided at the model layer for computer to read and write models in SW languages. Integrated parsers and writers are available for RDF/XML, N3 (a language which is a compact and readable alternative to RDF's XML syntax, but also is extended to allow greater expressiveness) and N-Triple (N-Triples is a line-based, plain text format for encoding an RDF graph, it was designed to be a fixed subset of N3). It support RDQL

(RDF Data Query Language). RDQL consists of a graph pattern which is expressed as a list of triple patterns: an RDQL query can additionally have a set of constraints on the values of those variables. Each triple pattern contains named variables and RDF values (URIs and literals). An application that use the Jena2.1 API can be optimized by making the representation of the model in memory more efficient.

One very important property of Jena2.1 is that it has several useful reasoners: Transitive reasoner, RDFS rule reasoner, OWL reasoner. The transitive reasoner is the simplest reasoner provided in Jena 2.1. Its capabilities are limited to providing full transitive closure and symmetric inferencing for the properties of *rdfs:subPropertyOf* and *rdfs:subClassOf*. For example, if it is asserted that class B is a subclass of class A and class C is a subclass of B, the transitive reasoner will return classes B and C (the inferred indirect subclass) when subclasses of A is requested. The RDFS rule reasoner affords all the reasoning features provided by the transitive reasoner plus more additional entailments possibilities. As claimed by Jena 2.1, the RDFS reasoner in the current release “supports almost all of the RDFS entailments described by the RDF Core working group (<http://www.w3.org/TR/rdf-mt/>)”. Among other capabilities available in this reasoner not found in a transitive reasoner is the ability to do schema validation, for example checking that a class’ asserted *rdfs:domain* and *rdfs:range* are not violated. Jena 2.1’s documentation describes its OWL reasoner as an “instance-based” reasoner, applying rules to propagate if- and only-if- implications of the OWL constructs on instance data. It is an extension of the RDFS reasoner, supporting the capabilities of the RDFS reasoner at the same time supports entailment on properties found only in the OWL language such as *owl:minCardinality*, *owl:sameAs* and *owl:differentFrom*. Jena 2.1 however cautioned that this reasoner is still not fully developed and that some features are still buggy.

3.5.2 Several Other Ontology APIs or Reasoners

There are many APIs or reasoners available for development of Semantic Web, I will now introduce the most popular ones and compare their features:

OWL Genie[26]

A set of XSLT named-templates that extract information from OWL Ontologies. It is a read-only API for XSLT. This is a limited scope API for use with XSLT that provides read only access to locally stored OWL ontologies. Added functionality would be nice, as well as support for OWL data.

DAML API[27]

The DAML API is a collection of Java interfaces and utility classes that implements an Interface for manipulating DAML ontologies. The DAML API has reached its highest level of development as an advanced prototype, and no new versions of the DAML API will be released. The release by the W3C of the OWL Web Ontology Language and the DARPA DAML program transition to OWL have provided motivation to develop a new API. The new API is anticipated to use a paradigm analogous to the DAML API, so transition to it should be straight forward.

RDF API[28]

An RDF parser, it needs to store in memory model before it runs. API. Last updated in January of 2001, so is not up to date with the latest changes to RDF. It does not support the latest version of RDF.

Pellet[29]

Pellet is an OWL DL reasoner which based on the tableaux algorithms[30]. It supports the DL known as SHIN(D) [31] which corresponds to OWL DL without nominals. Therefore it supports all the OWL DL constructs except owl:oneOf and owl:hasValue. It

can be used in conjunction with either Jena or OWL API libraries. Pellet API, it provides some functionalities to test the species validation, check consistency of ontologies, classify the taxonomy, check entailments and answer a subset of RDQL queries (known as ABox queries in DL terminology). It can also be used as complete owl consistency checker. It is used in ontology development, web service composition, multi-ontology reasoning, etc.

Racer[32]

RACER (Renamed ABox and Concept Expression Reasoner) is a description logic reasoner for OWL DL that implements a highly optimized tableau calculus for very expressive description logics. It also offers reasoning services for multiple TBoxes and for multiple ABoxes encoded as OWL DL knowledge bases. RACER is used as the reasoner for provided OWL ontologies, so the seeker agents would have the ability of querying the ontologies about provided web services.

3.5.3 Why Jena2.1 Is Appropriate For Our Project

Jena2.1 is a mature API for OWL, RDF, DAML+OIL data and ontologies. Features include both memory and DB based persistence, RDQL support, inferencing capabilities, syntax checking, and schema generation. Some of these extra features are beta level, but it is supported by an active group of programmers.

According to the assessment of SemWebCentral [33] we have the following table. SemWebCentral is funded by the DARPA Agent Markup Language program. It is developed and maintained by InfoEther and BBN Technologies:

Table 3.1 comparison of APIs or reasoners

subject	Jena 2	OWL Genie	DAML API	RDF API	Pellet	Racer
Language Support	OWL& DAML,RDF	OWL	DAML+OIL	RDF	OWL	OWL
Based on language	Java	XSL	Java	Java	Java	Lisp
Complete consistency checker	Yes	No	No	No	Yes	Yes
Interface	Java,DIG[34], Command Line	xslt	Java,DIG	Java,DIG	Java,DIG	Java, DIG
Query language	RDQL	XSL	RDQL	RDQL	RDQL	Racer Query Language
Reasoner	Yes	No	No	No	Yes	Yes
Parser/validator	Yes	No	No	No	Yes	No
API	Yes	No	Yes	Yes	No	No
Open sources	Yes	Yes	Yes	Yes	Yes	No

The APIs and reasoners above support different ontology languages, Jena 2.1 support OWL , DAML and RDF, OWL Genie only support OWL, DAML API support DAML+ OIL, RDF API support only RDF, Pellet support only OWL; they are based on different languages, all of them are based on Java language except OWL Genie which is based on XSL. They also have different features of interface and Query language. From the table above, we can see that not every tool can work as reasoner, parser or API. All the APIs above are open source except Racer. Open sources program could provide a very good reference and precious experience for developers when they are developing their own application program, so the open sources programs are personally more valued.

Jena2.1 is a mature API for basic RDF/OWL ontology and data processing. The additional capabilities are helpful and should mature nicely. Jena is open source. Support is provided on a "best-effort" basis by volunteers through the jena-dev mailing list on Yahoo, the address: <http://groups.yahoo.com/group/jena-dev/>, we can

ask questions or problems solutions we met during the development directly to the Jena development team, because the Jena development team all read this list. I highly recommend the Jena2.1 API. Another reason for choosing Jena 2.1 is the fact that Protégé – OWL plugin was built on Jena 2.1. Using the same library minimizes the risk of incompatibility.

3.6 Knowledge Management and E-learning In SW

When we talk about SW, we have to mention Knowledge management and E-learning, because these two technologies has very close relation with SW, and they are used widely in the platform of SW.

3.6.1 Knowledge Management

In SW, all the information is based upon knowledge instead of documents. It aims to add a layer of intelligence to the information. For example, in SW, a concept does not mean the strings of letters of that word, but some relevant meaning of the word can be delivered.

An Knowledge Management tool can automatically search and retrieve information, organize and classify information. Concretely, it can extract relevant content from a document or page, summarize a document, automatically classify, cluster and match documents by concept. Document summarization and information extraction of Knowledge Management can present users with only the relevant information extracts from documents[35]. Usually we tend to express knowledge in natural language, but we also need a structured and formally defined expression for machine. Ontologies can eventually be the bridge to the significant gap between the two environments of human and machine. Some Artificial Intelligence (AI) technologies can offer some constructive solution methods in Knowledge Management.

3.6.2 e-learning [36]

E-learning does not aim only to offer knowledge for the users to learn, but does intelligently take all the content from the interaction of the participants and the website. Learning content changes constantly through user inputs, answers, new practices and heuristics.

E-Learning requirements are: efficient, just-in-time and task relevant learning. Learning material is annotated and the learning materials should be easily combined as a new learning content to user. According to user's preferences, it is very easy to find and combine the learning material which is useful or required for his learning objective. This process is based on semantic querying and navigation through the learning materials. For example, if a user wants to learn some knowledge about computer engineering, after a test or assessment of what he has learned before, the computer could get such a conclusion : he has a lot of knowledge in hardware but knows little in software. So the computer could continue giving him more materials in software, less but deep knowledge in the field of hardware, to make him know more about software and save his time on the deep knowledge of hardware. This will make his studying more efficient.

At present, there are some softwares for computer-based training which aim at automate education, but they are not really e-learning programs. E-learning tools do not only give user the service at anytime and at any places, but also give user the learning content they need.

In fact, the SW can be used as a very suitable platform for implementing an e-learning system, because SW technology provides all means or solution for e-learning system: ontology development, ontology-based annotation of learning materials, active delivery of the learning materials, etc.

3.7 Trust between SW sites [37]

We have argued that one of the main characteristics of SW is that ontology data can be used by different SW sites. With all this interconnectivity going on between different ontologies and different SW sites, the security should be considered. Generally, the successful usage of SW and its application depend on whether people trust the data or not. Therefore, it is very important to specify clearly the trust in SW sites. There are two possibilities:

1. The applications of the SW will use context to make sure whether the data are trusted or not. For example, if I get some information from a trustworthy friend, I know that I can use the information and safely trust the information from him.
2. The applications will contain proof checking mechanisms and digital signature. Digital signatures provide a proof that a certain person wrote (or agrees with) a document or a statement.

In ontology of SW, the metadata space is similar to a public bulletin board. Concepts are introduced by independent and distributed resource. This leads to the problem of logical consistency of different ontologies. Moreover, it is still necessary for the metadata space to guard against some malicious attempts in introducing false concepts.

The result of this will be a system that allows one to decide what or whom to trust on the SW. Then we can tell our computer who we trust and proceed different data according to different trust level. For example, web content and semantics of a SW allow the agents to produce intelligent answers to users' queries, but every result from the SW is checked with a digital signature checker which has been integrated into the SW Agent. Trust is one's confidence to a statement, the digital signatures permits one to associate with another in a level of trust. SW is interconnected source of machine processable

statements, statements can describe the trust-worthiness of other statements, those statements will be identified and authorized by agents.

3.8 SW architectures and applications

There are numerous SW applications that would be beneficial to users and developers, such as intelligent notification, information retrieval, decision support. I will introduce some organizations examples and their applications relevant to SW.

3.8.1 Examples of SW architectures

1. Mindswap [38]

Mindswap is the abbreviation of Maryland Information and Network Dynamics Lab Semantic Web Agents Project. It has plenty of resource, informations and open source application programs about SW. This site builds on a previous web site that used a toolkit based on a web ontology language called SHOE [39], which developed at the University of Maryland. This site uses many Web technologies (HTML, XHTML, XML, PHP, CSS, etc.) and couples them with Semantic Web languages (RDF, RDFS, DAML+OIL, OWL) and tools. Several useful open-sourced SW application tools can be found and downloaded from this website. In November of 2001, the W3C created the Web Ontology Language OWL, which is used in powering this web site and making it the first "Owl-compliant" web site to date.

The Semantic Web Agents Projects at the Maryland Information and Network Dynamics Laboratory (MIND SWAP; <http://www.mindswap.org>) has been developed many tools aimed at creating an integrated SW system for searching browsing and authoring. The

motivation of this MINDSWAP is creation of a “Semantic Web Portal”. It encourages people to add semantic markup to their documents, images and data, then more and more people can make use of the portal. The portal allows the integration of the markup process. It not only provide information but also allows user to create more links between documents or resources. Users can define terms which has extended ontological coverage or link web resources to terms from multiple ontologies. User can query various web back-ends that contain similar resources. MINDSWAP portal also includes the development of inference engines that can find relationships between entities, and the presentation technology which makes the information be appropriate for specific users.

In Mindswap website, there are a lot of valuable papers and some programs which are open sources codes and we can download for research and learning. There are many wonderful open source software projects which are very useful:

Table 3.2 Tools in Mindswap

OCRA	OCRA is a simple owl ontology crawler based on Jena.
Pellet	Pellet is an OWL DL reasoner based on the tableaux algorithms developed for expressive Description Logics.
PhotoStuff	PhotoStuff is a multimedia markup tool for describing images and movies in OWL
SWOOP	A hypermedia-based Web Ontology Browser and Editor
OWL Mindswap	Owl.Mindswap.org is the test homepage for the mindswap group. The goal of the owl page is to have all the content for a site generated from RDF, and to be able to change the presentation of this material with tweaks to the supporting ontologies.
DAML Ontology Search	A better way to search the ontologies in the DAML Ontology Library

2. CCSW [40]

CCSW is the abbreviation of Competence Center Semantic Web. It is a research and development center within the German Research Center for Artificial Intelligence, which also research in Semantic Web technologies such as XML, RDF/S, DAML+OIL/OWL, XSLT, RuleML, and (semantic) web services.

Its focus is on distributed information management, Web-based representations, ontologies and rule systems. CCSW considers the Semantic Web as the higher-level structure which emerge from a multitude of semantic sub-webs, each serving the needs of a specific Web community.

3. KAON [41]

KAON is an open-source ontology management infrastructure targeted for business applications. It includes a comprehensive tool suite allowing easy ontology creation and management and provides a framework for building ontology-based applications. An important focus of KAON is scalable and efficient reasoning with ontologies. Persistence mechanisms of KAON are based on relational databases.

KAON ontology language is based on RDF(S), but contains some proprietary extensions. KAON ontologies consists of concepts, properties and instances grouped in reusable units called OI-models (ontology-instance models). The division between concepts and instances isn't strict, an entity can be taken as a concept as well as an instance, this depends on the view of the observer. An OI-model may include other OI-models, thus having immediate access to all definitions from the included model.

3.8.2 SW Search Engine Examples

1. OWL Semantic Search [42]

This search engine is called OWL Semantic Search and its internet address is :
<http://plucky.teknowledge.com/daml/damlquery.jsp>

But after some experiments with this search engine, I found that its function is quite different from its name. It can not support OWL successfully and can only do the semantic search with the DAML ontology file. This search engine is in style of Triple Search (searched by construct a question phrase with subject, predicate and object) and defines the clause with subject, predicate and object. But it does not support multiple clauses search, neither could define the relations between the clauses. It can broaden queries with simple inference, such as equivalence, inversion, generalization and specialization.

In the following I will introduce the procedure of a search in OWL Semantic Search. Firstly, enter the desired Subject, Predicate, and Object in each of the entry boxes, then construct the clause, also user can enter the clause directly in the text box, next step is building the Query List, multiple clauses can be added to the query list. At last submitting the Query.

In this semantic search engine, the menu Search Options can be used to define some facts and relationship in the knowledge base. Use can select search method Standard, Extended with Equivalence, Extended with Inversion, Generalization, or Specialization.

2. Swoogle [43]

Swoogle is a crawler-based indexing and retrieval system for Semantic Web Documents (SWDs) written in RDF or OWL. Swoogle discovers, digests, analyzes and indexes online SWDs. Swoogle supports constraints on the URLs of SWDs, the classes/properties can be used or defined by them. An Ontology Dictionary indexes the classes and properties defined by the discovered SWDs. In Swoogle there are three kinds of functions available : document search, term search and Swoogle statistic.

In the middle of Swoogle home page, there are two search boxes titled with "Document Search" and "Term Search" respectively. The first one can be used to search semantic

web documents and the second one can be used to search Class or Property (in form of URIref) defined or used in the Semantic Web.

To use document Search, user need type the query string in the textbox in "Document Search" box, and then click "Swoogle Search" button below the textbox. "Term Search" locates at the home page. User can finding Classes and/or Properties defined in the Semantic Web. To use "Term Search", user need type query string in the textbox in "Term Search" box, and then click "Swoogle Search" button below the textbox. There are two checkboxes for filtering the type of term, i.e., users can search for only classes, only properties or both.

Swoogle Statistics is an extra function of Swoogle. Based on the discovered documents and ontologies, we collect several statistical results about the Swoogle performance and about the growth/structure of the Semantic Web. Currently the statistical results are visualized as figures and tables.

3. Search Engine for the Semantic Web (BETA) [44]

This site can be used like traditional Web search site by just typing a few keywords describing the information we are trying to find. However, if this produces a large number of irrelevant results we can then harness the real power of the Semantic Web. This search engine allows us to use the vocabularies of the Semantic Web to narrow our search by the specific type or properties of the resource (person, place, or thing) we want to locate.

Internet Address: <http://www.semanticwebsearch.com/>

By using the standard search engine interface user can just type one or more keywords describing the information we are trying to locate. This is no more complicated than a traditional Web search engine. However, like a traditional Web search engine this can lead to a large number of irrelevant results. To narrow our search we can restrict it to the

specific type of resource.

If the search is still producing a large number of irrelevant results than we can refine it further by specifying one or more specific property values that the resource must have. For example if we are trying to locate a person with a last name of 'Smith' and a first name of 'John' we would enter the search string '[foaf:surname]~smith [foaf:firstName]~john'.

4. The SHOE Search Engine [45]

It is also called the next generation in web search engine technology. This search engine is based on language SHOE. SHOE uses XML-like tags and advanced artificial intelligence technology to make keyword based search engines a thing of the past.

The SHOE solution associates a context with a web page; this context can be used to disambiguate terms and provide background knowledge that might help in interpreting content. In SHOE this context is an ontology, which is really just a fancy way of say "a vocabulary and what it means."

Internet Address <http://www.cs.umd.edu/projects/plus/SHOE/search/>

3.8.3 W3C (World Wide Web Consortium)

"The W3C develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. It is a forum for information, commerce, communication, and collective understanding." ERCIM, and Keio. World Wide Web Consortium, 2003. <http://www.w3.org/>.

W3C's long term goals for the Web are (<http://www.w3.org/Consortium/>):

1. *Universal Access*: To make the Web accessible by promoting technologies and make it used by different users on all the world;

2. *Semantic Web*: To develop a software environment that permits each user to make the best use of the resources available on the Web;

3. *Web of Trust*: To guide the Web's development with careful consideration for the novel legal, commercial, and social issues raised by this technology.

W3C concentrates its efforts on three principle tasks:

1. *Vision*: To identify the technical requirements that must be satisfied if the Web is to be a truly universal information space.

2. *Design*: W3C designs Web technologies to realize this vision, taking into account existing technologies as well as those of the future.

3. *Standardization*: W3C contributes to efforts to standardize Web technologies by producing specifications that describe the building blocks of the Web. W3C makes these Recommendations (and other technical reports) freely available to all.

CHAPTER 4

Toward a System of SW

Considering the shortcomings of traditional website and its search engines, I decided to develop a semantic website which is more reliable and more intelligent. In this chapter, I will introduce an approach for the construction of SW system, then particularly I will present one important component of this system, called SWSAS (Semantic Web Search Agent System). I will introduce the ontology development for SWSAS, using Jena API in developing SWSAS, search engine algorithm of SWSAS and characteristic of SWSAS.

4.1 Structure of the SW System

This SW system is an SW application website for the computer engineering department. By this system, user can do semantic search, ontology annotation, customize the viewer to browse the ontology information or query result. The whole architecture of the SW system consists of three main layers, Client part, Server part and Ontology (Knowledge) server part.

This architecture is illustrated in Figure 4.1. The web architecture is essentially a client-server architecture. In the Client layer, user can input query question and browse the query result, it is the web browser. In the Server layer there are a search engine, a customized viewer and annotation tool. Finally the storage layer can be either a local or a remote file system that hosts the knowledge base file which there are ontology and instances data in it. In this Semantic Website, the portal of annotation tool set in Server through which the system can manage the distributed ontology. In this system the annotation functions are front-end module that contains all annotation functions for

managing the individuals for the following classes: professor, project, courses, research, degree, etc. In the search agent module, there is a query execution module which can parse the information and draw some inferences from these data. Finally the search results are presented through the customized view module, through which the client can get the result on the browser or explore the information.

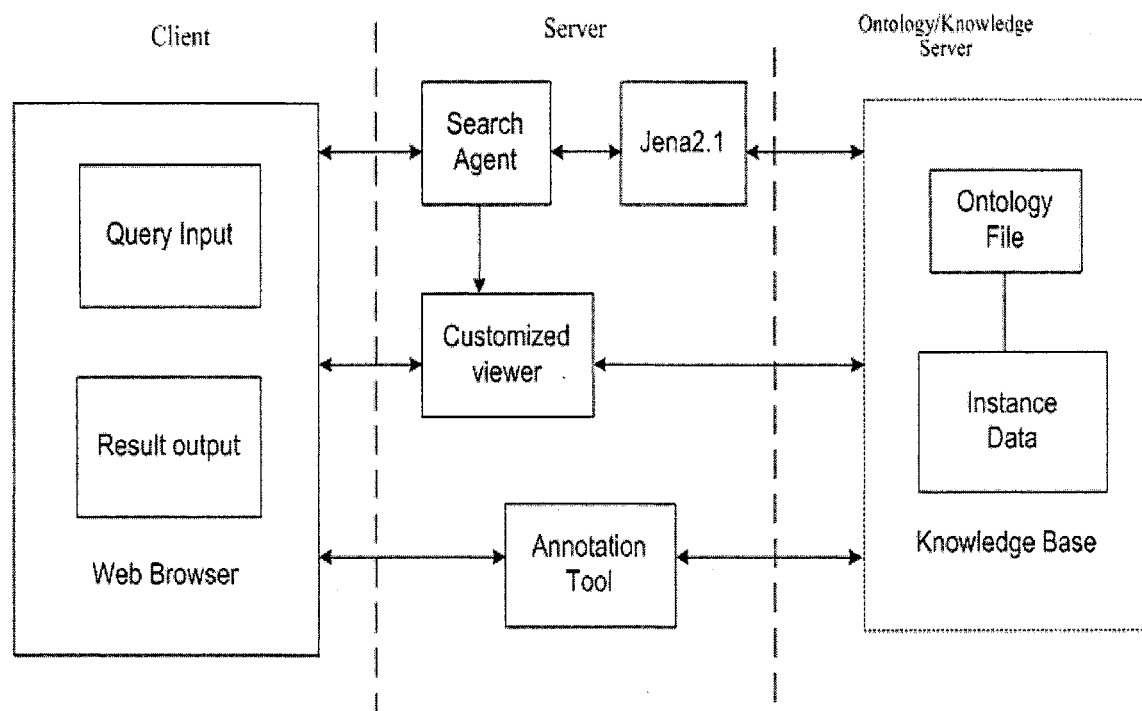


Figure 4.1 Overall Architecture of a SW

At present, I have developed SWSAS (Semantic Web Search Agent System) which consists of the Semantic search agent, and an ontology of the computer science domain which I contributed to the development of it.

4.2 Development of the Ontology

Here I will introduce how an ontology has been developed. It will include the procedure of the Ontology design, introduction of the ontology design tool, and the characteristic of the ontology.

4.2.1 Protégé as tool for developing the ontology

Many tools have been created by different companies and research institutes for using and developing ontologies, instances and the web application itself. One tool central to the development of ontology and instance creation is Protégé and its Protégé-OWL plugin.

Protégé's flexible open-source platform means that it is easy to integrate customized components to build user's applications. The OWL Plugin is a Semantic Web extension of the Protégé ontology development platform and it is now one of the world's most widely used OWL editors. It can be used to edit ontologies in OWL, to acquire instances and access description logic reasoners. OWL Plugin provides a complete, scalable and stable editor for most of the features of OWL. It has made ontology design more accessible to people without formal training in description logics. The OWL Plugin has created and facilitated new practices for building Semantic Web contents which often driven by the needs of and feedback from the users.

One of the major benefits of using Protégé is its open architecture. The system provides various mechanisms for external components like reasoners and web services can be integrated easily. Since the source code is open and freely available as well, existing base components can be used as templates for customized solutions. Projects don't need to spend time developing their own base infrastructure with standard features such as loading and managing OWL ontologies. Instead, they can start with the OWL Plugin then gradually adapt or remove the features that don't completely match their requirements. About more information of protégé please see 2.5.1.

4.2.2 Procedure used to develop an ontology

I accepted a certain procedure for designing the ontology. It covers all the major activities for creating a clear, objective, complete and consistent ontology. In modern ontology development, the activities of specification, knowledge acquisition, and conceptualization are often intertwined. Generally, before designing an ontology, we should gather requirements and specification for ontology, and analyze the requirements and specifications, then elicit knowledge using knowledge elicitation techniques and design the conceptual model. If we can reuse other ontologies, we try to integrate them into our design ontology. Based on different domains and specific projects, the process of constructing an ontology is quite different.

There are seven steps in developing an ontology according to the paper "Ontology Development 101" [46]. In our ontology development, the whole procedure of developing an ontology has only five steps. These five steps can help the developer to develop ontology efficiently and easily, this has been proved by ontology development in our project. The five steps are requirement analysis, considering reusing the ontology, define the class and properties, then define the value feature of the property, at last create instance.

Step 1. Requirement analysis

In this step, we should determine the domain and scope of the ontology, the concepts which are used to describe different professors, classes and research domains have been figured out in this step. The ontology also includes the relations between professors, classes and research domains.

In our ontology, there are several basic facts. The classes are taught by professors, the research domain are researched by the professors or the researchers. Professors have their personal information. A course has its own information or schedule. A research domain has its own content and researchers, etc. These issues need not be exhaustive at this step. At least we know the ontology can be used to assist the user in identifying the teaching and research activities of every professor, as well as the information on every course and every research domain.

Step 2. Consider reusing existing ontologies

It is very important to consider whether it is possible to reuse existing ontologies in our particular domain and task. We can think about what someone else has done and check if we can refine and extend some existing ontology or resource in our ontology.

On the web, there are many ontologies that already exist and available. The formalism in which an ontology is expressed is not important, since translating from one formalism to another is not very difficult.

Step 3. Define the classes (and the class hierarchy) and their properties

This step consists in developing the class hierarchy and defining properties of concepts. It is the most important step in ontology design. Generally, we create a few definitions of the concepts in the hierarchy, then continue by describing properties of these

concepts. It is important to find the exact and comprehensive terms for the classes and properties. We should avoid the overlap between concepts, we should think about the relations among the terms and any properties that the concepts may have, or whether the concepts are classes or properties.

When we define classes, we select the terms which can describe objects in independent existence instead of terms that describe these objects. We input these terms as the classes in the ontology. The classes should be organized into a hierarchical taxonomy by their relations. If a class A is a superclass of class B, then every instance of B is also an instance of A. Once we have defined some of the classes, we must get to know how to describe the internal structure of the concepts. For each property we must know which class it describes and which class it is attached to. Generally there are three most popular approaches in developing a class hierarchy: top-down development process, bottom-up development process and a combination process. A top-down development process starts with the definition of the most general concepts in the domain then subsequent specialization of the concepts. This is adopted in our ontology development. A bottom-up 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. A combination development process is a combination of the top-down and bottom-up approaches, this is adopted in our ontology development.

Different from the “Ontology Development 101”, we don’t have the step of “Enumerate important terms in the ontology”. This is because before we define the classes and their properties, it is natural and necessary to consider the ontology terms and decide which term which should use, we do not want to do the redundant job in two steps. Another difference from “Ontology Development 101” is that in our development we put defining the class and defining their property together in one step. This is because when we define classes or properties, we need to think about both of them, since classes and properties can not be really separated. For example, when we define a class, we should

think about how many properties it has. When we define a property, we should think about how many classes could have this property in the ontology. Moreover, we should consider clearly whether a term should be defined as a class or a property in our ontology.

Step 4. Define the value feature of properties

Value features of properties are value type, allowed values, cardinality and other features of the values. For example, *name* is a property with value type *String*, the value of a name property is one string. Property could have multiple values, these values are instances of the class which the property attached to. Cardinality defines how many values a property can have. Some ontologies allow specify a minimum and maximum cardinality to describe the number of property values and some ontologies define only between single cardinality and multiple cardinality,.

Value type describes what types of values the property can be. For example *Types:String* is the simplest value type which is used for properties such as *name*: the value is a simple string to describes property. Enumerated specify a list of values for the property. Instance type properties allow definition of relationships between individuals. When the value of property is the instance of some class, we must specify the allowed classes for this value. The domain and the range of a property are very important in the definition of an ontology. The range of a property can be restricted when the property is attached to a particular class. Instances are often called a range of a property. The classes to which the property is attached usually constitute the domain of that property. There is no need to specify the domain separately. When defining a domain or a range for a property, we should find the most general classes or class that can be part of the domain or the range for the properties respectively.

Step 5. Create instances

The last step is creating individual instances of classes in the hierarchy. Defining an individual instance of a class requires. First we choose a class, then we create an individual instance of that class, at last we fill in the property values.

For example, we can create an individual instance *Guy_Bois* to represent a specific platform. *Guy_Bois* is an instance of the class *professor* which represents all professors. This instance has the following property values defined :

Has Education Degree : Ph.D._University_Montreal

Has Email Address: guy.bois@polymtl.ca

Has Title: Assistant Professor

Has Fax Number: 514-340-3240

Has Firstname: Bois

Has Lastname: Guy

Has Office Address: D-6365

Research:

Microelectronics(performance optimization, system specialization and concept ...)

Numeric Circuits integration

Software

Computer Architecture

Teach:

INF6501 Specif. Et conception de systèmes embarqués

INF6600 Conception et analyse des systèmes temps réel

INF2501 Logique Electronique

INF4600 Systemes en temps réel

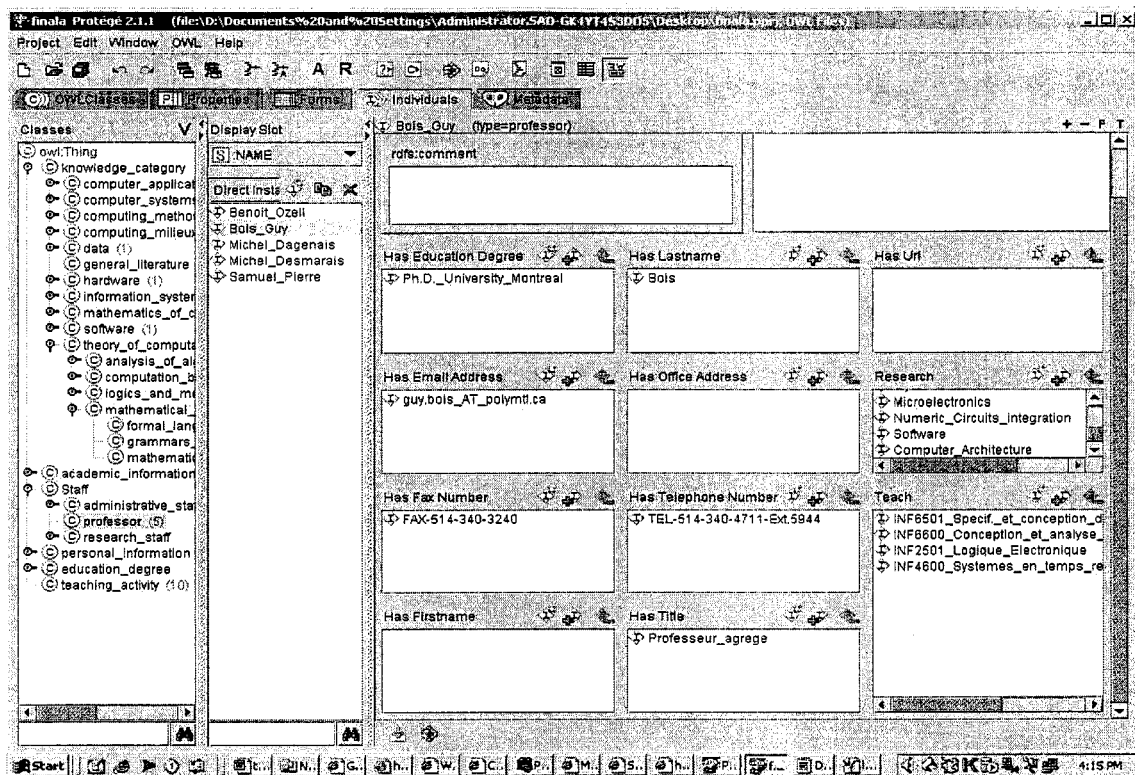


Figure 4.2 The Definition of An Instance of Professor Class

In the research property above, Microelectronics, Numeric Circuits integration, Software, Computer Architecture are instances and each one has individual property values. In the teach property, INF6501, INF6600, INF2501, INF4600 are also individual instances with different properties values.

4.2.3 Using ACM Computing Taxonomy

An Ontology is created for the management of the teaching and research work of Computer Engineering Department. It is based on the information of professors, courses, researches and other related information. In this Ontology we use ACM Computing Classification System for the classification and definition of the topics covered by

research and teaching courses in Computer Engineering Department. The ACM Computing Classification System (CCS), which has served as the primary and most generally used system for the classification and indexing of the published literature of computing, has been substantially revised to reflect the greatly changed nature of computing. In my project, I use the version of CCS 1998. ACM Computing Classification System is a full classification scheme and involves the following three concepts: the four-level tree (containing three coded levels and a fourth uncoded level), General Terms, and implicit subject descriptors.

By using ACM Computing Classification System, user could query a certain computing knowledge concept then get the result information about that computing concept, for example, what upper topics it belongs to and what other lower topics it includes. It is useful for user to learn the computing concepts. Furthermore, the knowledge node of ACM system could be used as course content and research content. But because different university has different definition of courses and researches, sometimes the topics of ACM are not corresponding to the university department course or research subjects, so in this case some concepts of the ACM computing classification system are difficult to be used directly to define the specific course or research in computer department, we have to create course or research content in the ontology ourselves. In my project, there are 90 percent concepts of ACM classification concepts that could be used directly by us in defining the content of course or research..

4.2.4 Metrics of the ontology

In the Ontology for SWSAS, there are 1515 classes 62 properties, and 1440 instances. Classes includes the academic topics, which mostly comes from ACM system. In the knowledge category there are 1473 topics which are used in the computer domain. This ontology includes computing knowledge topics, professor, course and research, properties include teach, research, personal information, course information, etc. This ontology is based on the teaching and research work of Computer Engineering

Department and almost all the information of professors, courses, research and other related information are involved in this ontology.

4.2.5 Why database is not used to store the Ontology file

In this architecture, a database is not used to hold the ontology file and the file that stores the individuals, although Jena 2.1 provides method for persistent storage. The reason for not storing the ontology file into the database as split triples is because it does not benefit performance. Regardless of whether they are kept in the database or as files, the entire ontology content has to be read into a model that is kept in-memory before it can be used for querying or manipulation.

In a database system, entailments are not possible. Furthermore, in our ontology we managed to describe the fields in Computer Science at different levels. If we keep value in tables which are flat, there will be no way to describe Computer Science fields in the multi-level hierarchy that we wanted. Scalability of our application is not an issue now as in the real environment the department does not offer thousands or millions of projects. So we had the individuals of the ontology kept in an OWL file. However, if the data is to grow very large, then storing the OWL files in a database would be recommended since only a database can provide the security and scalability sought. Nevertheless, we do not foresee a large amount of data that makes the use of a database worthwhile.

4.3 Use Jena as API

I use Jena2.1 as the reasoner of the ontology. There are two forms of Jena querying: triple match and RDQL querying. A triple match returns all statements that match a template of the form (subject, predicate, object). An RDQL query [47] can be expressed

as a conjunction of triple match templates. In these two queries the query terms can be constant or variable. Variables enable joins across the triple matches. A query returns all valid bindings of its variables over statements in the graph.

In Jena 2.1, the ontology and individuals stored in files can be processed, manipulated and queried only if they are represented as models. There are two types of models provided by Jena 2.1 : the inference models and the non-inference models. Reasoners can only perform inferencing on inference models. Jena 2.1 API provides an RDFS inference model with the `InfModel` class and an OWL inference model with the `OntModel` class. Inference models, when queried, return not only those statements that are present in the original data but also additional statements that can be derived from the data using rules implemented by the reasoner. On the other hand, non-inference models return only the asserted facts when queried. Inference models are built from non-inference models that hold the ontology and the individuals (data). Inference can derive additional information which is entailed from assertions made in the knowledge base together with rules associated with the reasoner. Three predefined reasoners available in Jena 2.1 that is sufficient to elicit the type of answers we want from the knowledge base. They are transitive reasoner, RDFS or OWL reasoners. If we sought a reasoner that can provide transitive closure inference on the property `rdfs:subClassOf` or a separately defined property in our ontology, all three are capable of providing that.

For example, my project requires means to infer all the descendents of a given class so that when the reasoner is asked “What are the computing topic that are related to Artificial Intelligence(AI)?” all topics that relate to Artificial Intelligence or any of its descendents will be returned. A query that asks “Find all fields that are a subclass of the field AI” is sent to the inference model which Using the transitivity of `rdfs:subClassOf`. We can use the transitive property of `rdfs:subClassOf` to infer all descendent subjects under a root or parent field for example the AI field. The experiment was carried out with a subset of the Computer Science taxonomy from our ontology. Figure 4.3 shows a subset of our Computer Science field taxonomy:

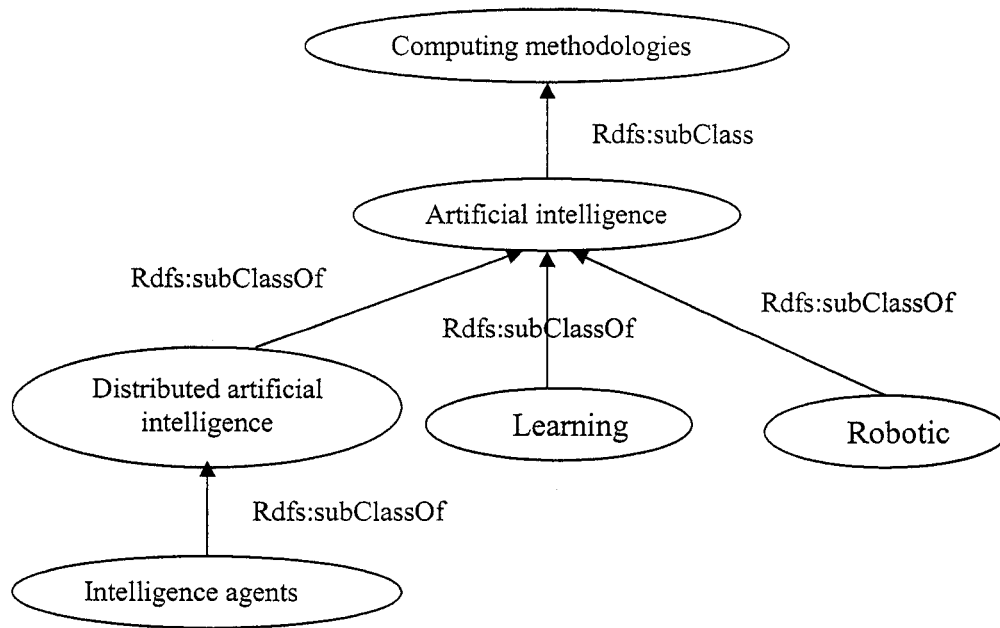


Figure 4.3 A fraction of Computer Science taxonomy

The query “What are the computing topics that are related to Artificial Intelligence(AI)?”

is sent to the inference model, we get the following result:

<http://www.polymtl.com/ontology#DistributedArtificialIntelligence>

<http://www.polymtl.com/ontology#Learning>

<http://www.polymtl.com/ontology#Robotics>

<http://www.polymtl.com/ontology#IntelligentAgent>

4.4 Characteristic of SWSAS

SWSAS is a search engine system which adopt the most advanced technology. It is a prototype for the future search engine technology. The whole application system is

based on the knowledge and technology of Semantic Web and Ontology, Knowledge Presentation and Intelligent Search Engine.

In the following sections I will introduce some special attributes of SWSAS which make it an efficient application search system: it supports almost all the ontology languages, for example OWL, DAML, RDF; It has triple clause search as well as the multiple clauses search; Offer two search patterns, Simple semantic search and Triple semantic search; it has a reasonable and proper ontology.

4.4.1 Support both OWL ontology and DAML ontology

SWSAS supports not only OWL ontologies but also DAML ontologies. OWL and DAML are the highest layer of all the ontology languages and OWL is replacing DAML in the field of SW, but DAML has been used for a long time and on internet there are plenty of resources in DAML, for example, on internet there are many DAML ontologies, some DAML search engines and DAML Web Service systems, etc. On another side, DAML is still a standard ontology language and adopted by many developers and users, it is in the same level with OWL, so I think there exist necessity to make SWSAS to be compatible with DAML. Briefly, SWSAS gives users a possibility to make full use of the ontologies in OWL and DAML.

Though OWL and DAML are in the same layer of the ontology languages, OWL has some properties which DAML does not have. OWL will definitely become the most popular and most powerful ontology language, so in SWSAS OWL is adopted as the highest priority language. OWL enables software agents to dynamically identify and understand the data sources. In the OWL ontology which I created for the Department Computer Engineering of Ecole Polytechnique de Montreal, there are many classes, sub-classes and properties which give information about teachers, research domains and courses, also some instance in it. By the SWSAS some potential information can be

extracted through the inference of ontology language and constraints between the classes or properties from the ontology data.

Before creating this ontology, we searched in the internet but we did not find any ontology suitable or available for reusing in our project. So we have to create it ourself. “The ACM Computing Classification” is adopted as the computing knowledge category and courses definition of the Computer Engineering Department of University when we create the otnolgoy. As a famous standard “The ACM Computing Classification” is accepted widely all over the world. Therefore, it is a good choice for stating an ontology. Users are not required to know about 1400 ACM items, those item be found in “help” or appear in the dropdown menu automatically.

The Ontology document could be OWL, DAML or RDF. Before the search, it needs load the ontology file into the system, software can get the ontology file through the address.

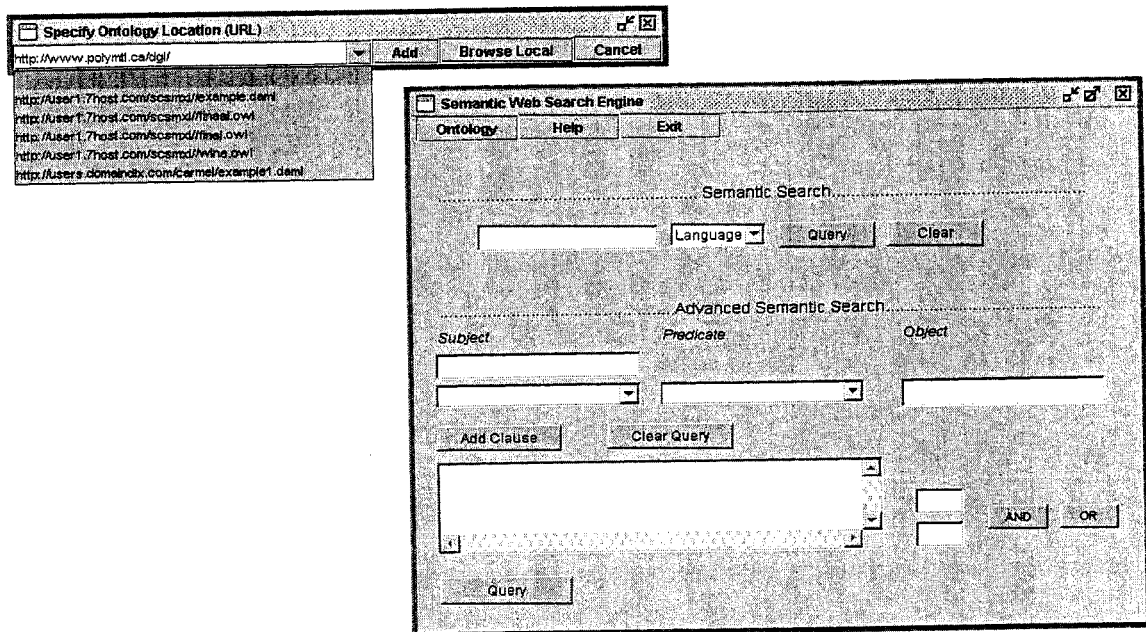


Figure 4.4 Upload Ontology

4.4.2 Two Search Models

In SWSAS, two search patterns are available to users. One is the Semantic Search engine and the other is the Advanced Semantic Search engine. Both of them can work independently from each other (see Figure 4.5). From the interface, Semantic Search engine looks like the traditional search engine. The user just needs to input one word which he wants to query. Then he can get the semantic search result, SWSAS can return many general semantic information related with this word. The advantage of Simple Search engine is that it is easy to operate. We just input one word or several words for search, then click the 'search' button and the search results is presented to the user. But the search process and result of this search come from concept search, certainly quite different from traditional web search engine. For example, if a user want to know some information about "real time system" in the department of computer engineering, he just inputs "real time system" into the search engine, then the search engine returns him all the semantic information about "real time system" including the courses and research which are related with "real time system".

Example of Semantic Search:

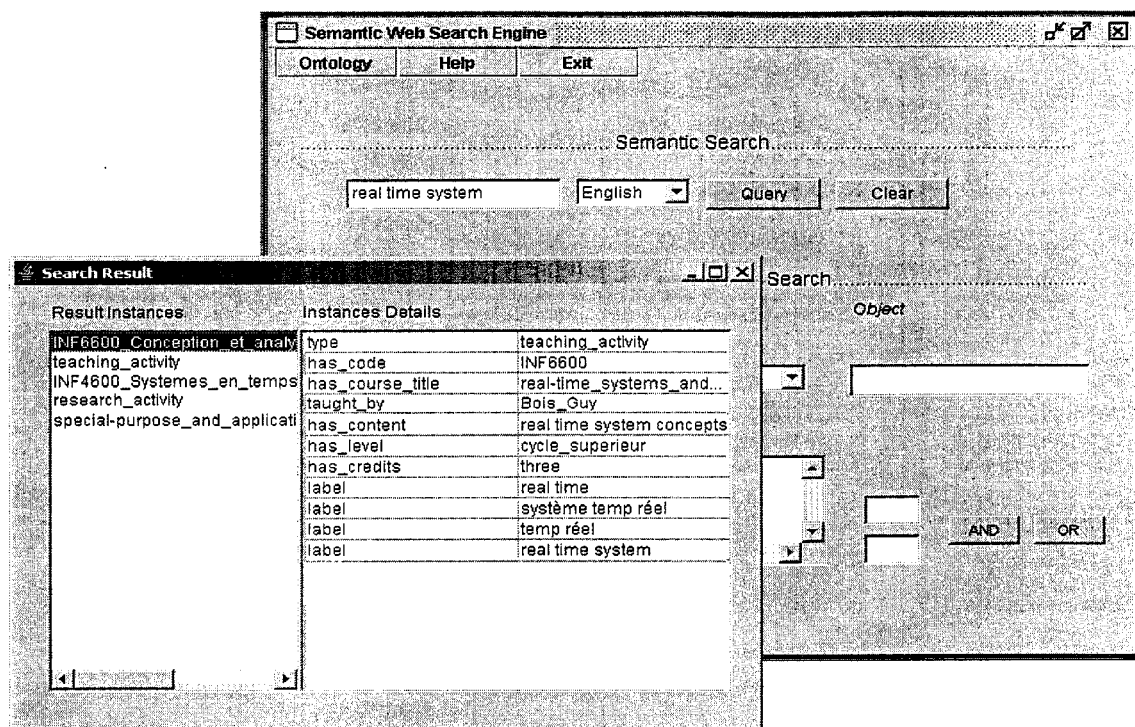


Figure 4.5 Semantic Search

In Triple Search engine of SWSAS the user needs to construct a clause which consists of subject, predicate and object. There is no limitation about the number of clauses. The quantity of the clauses depends on the user's needs and how many restrictions users want to add. The Triple Search engine of SWSAS is very efficient, user can query a very concrete question or more than one question in one query, then get the concrete query result according to his concrete question. For example, if user want to know "who teach real-time system?", in this phrase, the subject is " who ", the predicate is " teach " , the object is "real-time system ", then user can construct this question which consist of subject, predicate and object in the search engine, process the semantic search and get the result.

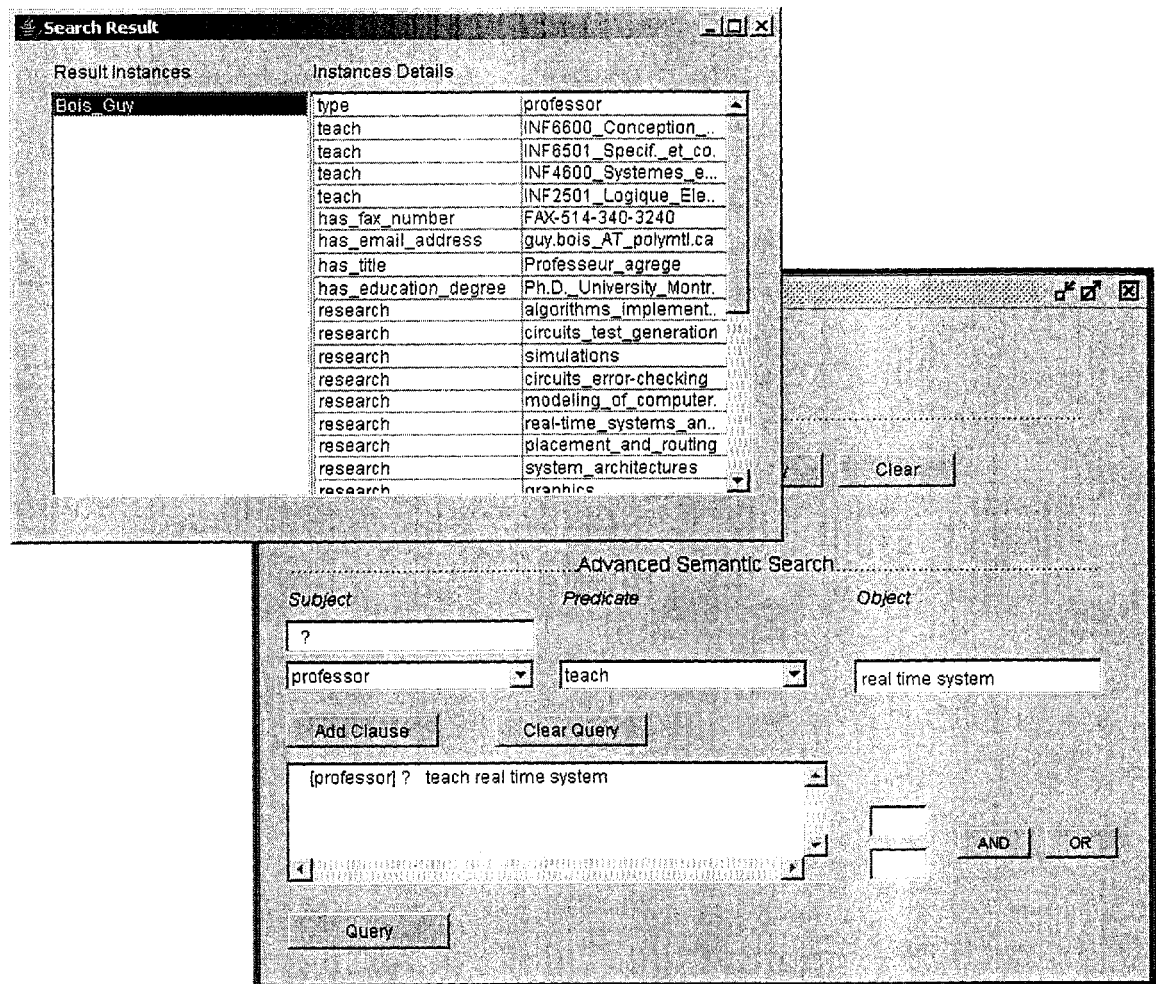


Figure 4.6 Advanced Semantic Search

4.4.3 Support Multiple Clauses Search

Currently almost all the SW search engines support only one clause at one search, but SWSAS is able to support multiple clauses at one query. It also support Boolean combination of clauses. This give SWSAS more powerful functions and applications. Users can add as many search constraints as they want through the multiple clauses, in order to filter the search results and obtain the exact search results.

Example of multiple clauses query:

For example, these are two clauses which users probably want to query:

1. Which professor has the title of “Associate Professor”?
2. Which professor research in “human-machine interaction” ?

If the user wants to do the query “which professor has the title of Associate Professor and does research in Artificial Intelligence?” the user just adds two query clauses as the query question, and add relation “and” to these two clauses. Then the result information comes out through one click on the button “query”. The user can also put the relation “or” to the two clauses in order to search “ the professor who has the title of Associate Professor or the professor who research in Artificial Intelligence?”.

The screenshot displays the SWSAS search interface. The top window, titled "Search Result", shows a table of instances for Michel Desmarais. The bottom window shows a query builder interface with two clauses: "[professor] ? has_title associate professor" and "research human-machine interaction" connected by an "AND" relation. The query builder also includes buttons for "Add Clause", "Clear Query", and "Query", as well as "AND" and "OR" relation buttons.

Result Instances	Instances Details																																		
Michel Desmarais	<table border="1"> <tr> <td>type</td> <td>associate professor</td> </tr> <tr> <td>research</td> <td>human-machine interac</td> </tr> <tr> <td>research</td> <td>software engineering</td> </tr> <tr> <td>research</td> <td>user modeling</td> </tr> <tr> <td>research</td> <td>networks bayesiens</td> </tr> <tr> <td>research</td> <td>intelligent agents</td> </tr> <tr> <td>has_education_degree</td> <td>Ph.D_in_computing_d...</td> </tr> <tr> <td>has_office_address</td> <td>D-6351</td> </tr> <tr> <td>has_url</td> <td>profsgl.php</td> </tr> <tr> <td>research</td> <td>artificial intelligence</td> </tr> <tr> <td>has_email_address</td> <td>michel.desmarais@poly</td> </tr> <tr> <td>has_lastname</td> <td>Desmarais</td> </tr> <tr> <td>teach</td> <td>INF3300_Gestion_de...</td> </tr> <tr> <td>teach</td> <td>INF2700_interface_per...</td> </tr> <tr> <td>has_telephone_number</td> <td>TEL-514-340-4711-Ext...</td> </tr> <tr> <td>has_fax_number</td> <td>FAX-514-340-3240</td> </tr> <tr> <td>seeAlso</td> <td>www.cours.polymtl.ca/i...</td> </tr> </table>	type	associate professor	research	human-machine interac	research	software engineering	research	user modeling	research	networks bayesiens	research	intelligent agents	has_education_degree	Ph.D_in_computing_d...	has_office_address	D-6351	has_url	profsgl.php	research	artificial intelligence	has_email_address	michel.desmarais@poly	has_lastname	Desmarais	teach	INF3300_Gestion_de...	teach	INF2700_interface_per...	has_telephone_number	TEL-514-340-4711-Ext...	has_fax_number	FAX-514-340-3240	seeAlso	www.cours.polymtl.ca/i...
type	associate professor																																		
research	human-machine interac																																		
research	software engineering																																		
research	user modeling																																		
research	networks bayesiens																																		
research	intelligent agents																																		
has_education_degree	Ph.D_in_computing_d...																																		
has_office_address	D-6351																																		
has_url	profsgl.php																																		
research	artificial intelligence																																		
has_email_address	michel.desmarais@poly																																		
has_lastname	Desmarais																																		
teach	INF3300_Gestion_de...																																		
teach	INF2700_interface_per...																																		
has_telephone_number	TEL-514-340-4711-Ext...																																		
has_fax_number	FAX-514-340-3240																																		
seeAlso	www.cours.polymtl.ca/i...																																		

Query Builder Interface:

professor research

Add Clause Clear Query

[professor] ? has_title associate professor
AND
research human-machine interaction

Query

Object

Clear

AND OR

Figure 4.7 Multiple Clause Search of SWSAS

4.4.4 Support multiple languages

This search engine has an obvious advantage, it can support different languages, such as English and French. This is very important because it can satisfy different users with different languages, users can query in their favorite language.

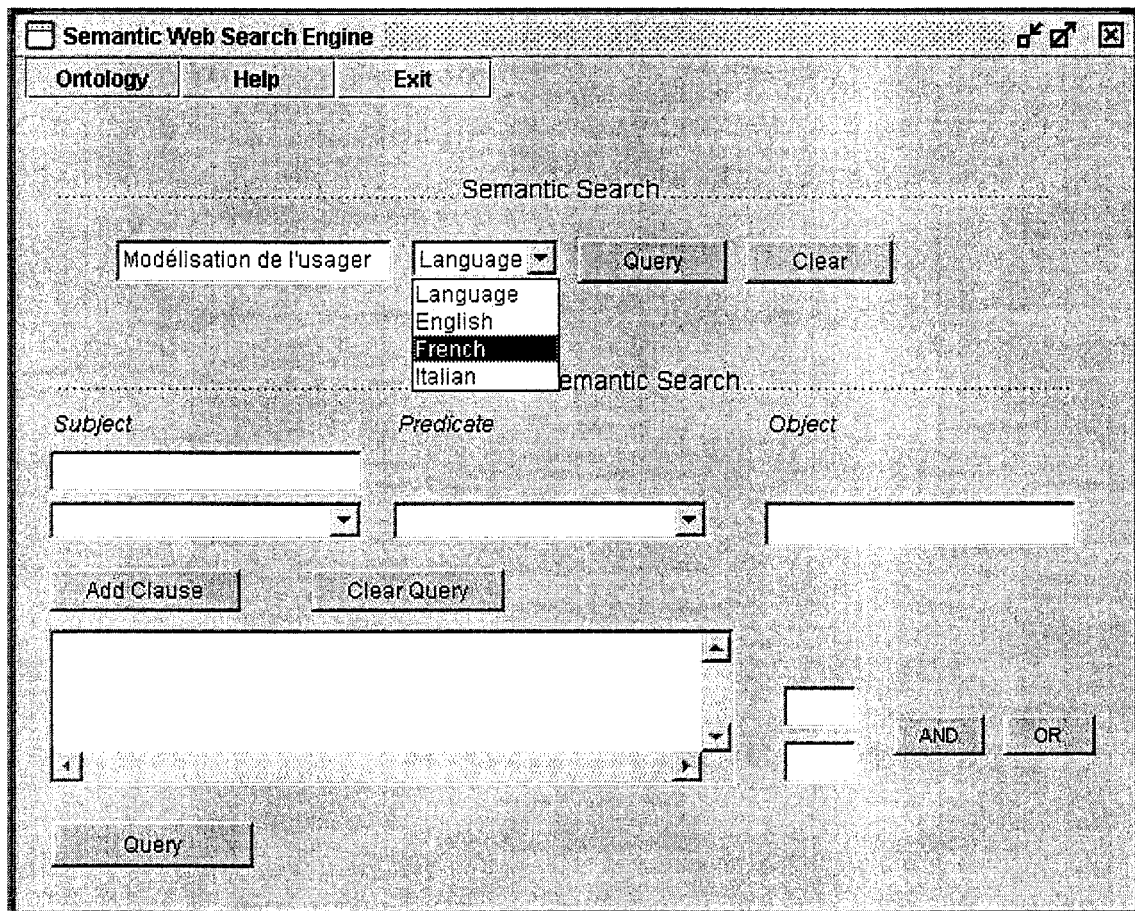


Figure 4.8 Multiple Languages

This project is based on the background of the computer engineering department of Ecole Polytechnique de Montreal, a French university, aim to solve the problems of the traditional website with the technology of SW. SWSAS support multiple languages and satisfy the requirement of the French university department. In French university all the

information on the website including the course name are in French, people know them exactly in French, but do not know the exact term for them when they are translated into English.

4.5 Algorithm of Search in SWSAS

As introduced before, there are two modes of query in SWSAS which available for user to choose, Simple Semantic Search and Advanced Semantic Search. In order to implement Simple Semantic Search query, Algorithm SimpleSearch(Description D) in Figure 4.9 is needed. In order to implement the Advanced Semantic Search, the algorithm InstanceSearch(class, property, D) and algorithm AdvancedSemanticSearch(class, property, D) are needed. The main idea of such algorithms is to get the exact result of the semantic search process. All the algorithm is introduced in pseudocode:

Algorithm SimpleSemanticSearch(Description D)

```

1: if reasoner.isInconsistent(D) then
2:   raise Exception
3: end if
4: result  $\leftarrow \emptyset$ 
5: bclass  $\leftarrow$  Reasoner.listClasses(D);
6: if bclass  $\neq \emptyset$  then
7:   equivalents  $\leftarrow$  reasoner.getEquivalents(D)
8:   if equivalents  $\neq \emptyset$  then
9:     for each  $c \in$  equivalents do
10:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
11:    end for
12:   End if
13:   instances  $\leftarrow$  Reasoner.getInstances(D);
14:   if instances  $\neq \emptyset$  then
15:     for each  $c \in$  instances do
16:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
17:    end for
18:   end if
19:   subclasses  $\leftarrow$  Reasoner.getSubClasses(D);
20:   If subClasses  $\neq \emptyset$  then
21:     for each  $c \in$  subClasses do
22:      result  $\leftarrow$  result  $\cup$  allIndividuals(c)
23:    end for
24:   end if
25: else
26:   bproperty  $\leftarrow$  Reasoner.getProperties(D)
27:   if bproperty  $\neq \emptyset$  then
28:     for each  $c \in$  Instance do

```

```

29:         result ← result U allIndividuals(c)
30:     end for
31: end if
32: end if
33: return result.

```

Figure 4.9 Simple Semantic Search Algorithm

This algorithm is to search whether the description concept D exists in the classes, properties or instances of this ontology. If it exists, continue to check whether that class have equivalent classes, the subclasses and the class instances, at last return all the related concepts to the result. If concept D is a property, then check the instances related with that property, then return all the instances to the result.

Figure 4.10 illustrate the algorithm InstanceSearch and figure 4.11 illustrates algorithm AdvancedSemanticSearch, they are for triple search (subject, predicate, object) used in the “Advanced Semantic Search” of SWSAS. InstanceSearch is for searching the instance which match with concept user input. Algorithm AdvancedSemanticSearch search instance with a pair of class and property, then continue to check the equivalent classes, the subclasses and the subproperties, at last return them to result. Jena reasoner can split the ontology data into a set of triples, then the two algorithms can compute the concept and instance by finding the triples matched with the user input.

Algorithm InstanceSearch (class, property, D)

```

1: Result ←  $\emptyset$ ;
2: inst ← class.listInstance(class)
3: if inst  $\neq \emptyset$  then
4:   for each  $c \in$  inst do
5:     iValue ← c.listPropertyValues(property);
6:     if iValue  $\neq \emptyset$  then
7:       for each  $cc \in$  iValue do
8:         if  $cc = D$  then
9:           result ← result U ( $cc$ )
10:        end if
11:      end for
12:    end if
13:  end for
14: end if

```

Figure 4.10 InstanceSearch Algorithm

Algorithm AdvancedSemanticSearch (class, property, D)

```

1: if reasoner.isInconsistent(D) then
2:   raise Exception
3: end if
4: result  $\leftarrow \emptyset$ 
5: result = InstanceSearch(class,property,D)
6: iEqClass  $\leftarrow$  Reasoner.listEquivalentClasses(D);
7: if iEqclass  $\neq \emptyset$  then
8:   for each c  $\in$  iEqclass do
9:     result  $\leftarrow$  result U InstanceSearch(c, property, D)
10:   end for
11: end if
12: iSubClass  $\leftarrow$  Reasoner.listSubClasses(D);
13: if iSubClass  $\neq \emptyset$  then
14:   for each c  $\in$  iSubClass do
15:     result  $\leftarrow$  result U InstanceSearch(c, property, D)
16:   end for
17: end if
18: iSubProperty = dp.listSubProperties(D);
19: if iSubProperty  $\neq \emptyset$  then
20:   for each c  $\in$  iSubProperty do
21:     result  $\leftarrow$  result U InstanceSearch(class, c, D)
22:   end for
23: end if

```

Figure 4.11 Advanced Semantic Search Algorithm

For example, if user want to search “artificial intelligence”, SWSAS uses the SimpleSemanticSearch algorithm, firstly find the concept match between “artificial intelligence” and concept labels, then find the concept match with ontology classes, at last identify its subsumed concepts and instance of “artificial intelligence”. Then we can get the search result which consists of all the concepts and instance. If the user wants to search “which course has the topic of Artificial Intelligence?” SWSAS uses the AdvancedSemanticSearch to find triple match (?course, has_topic, artificial intelligence) in the ontology, the result also includes equivalent class, subclass and subproperty.

CHAPTER 5

Evaluation of SWSAS

In this chapter, I will give a general evaluation of the search engine system SWSAS. Then, I will also compare SWSAS with some other semantic web search engines, in order to show its advantages and draw some conclusions. At last I will introduce the challenges in developing SWSAS.

5.1 General evaluation

General evaluation includes three parts: performance evaluation, ontology evaluation and search engine evaluation.

1. Performance evaluation

The evaluation of performance of this semantic web application system aims to ensure that all features of this system expected by the users are met as well as to know the effectiveness of this system.

This system is highly dependent on the correct usage of the system interface to produce the correct search process. The usability evaluation is done by observing how the users interact with the application. The result has proved that a user can use it easily and get familiar with the usage of this system quickly.

This system has a wide compatibility in languages, operation systems and ontology languages. It works under different operating systems, just like Windows, Linux or Unix. It can also work with different ontologies in different languages OWL, DAML or RDF. Furthermore, it also supports multiple languages, then people from different countries can use it in their own languages. All of these elements are very useful to the users, these expected features of users are met in this system.

Jena2.1 provides the deduction or reasoning power in this system, it is written in Java, and the programming language of SWSAS is also Java; they have shown a perfect compatibility to each other and this contributes to the high efficiency of the system.

2. Ontology evaluation

Ontology of SWSAS is created in language OWL. It has all the advantages of this new ontology language. It is very important to use OWL in this project, because OWL is becoming the most popular ontology language in Web. This not only makes it more compatible with other OWL ontologies but also makes it easier to update the SWSAS in the future. The ontology that we created for the computer engineering department can also be easily shared or reused by others based on its standard and sound design.

We use ACM computing classification system in building Topic ontology. ACM computing classification system is a known and standard classification, and accepted widely. Because in ACM all the concepts are in a reasonable hierarchy constructs the computing knowledge system. Through using it in the ontology, we can achieve a clearer understanding of the role of a certain concept in the whole computer science system knowledge nodes, also including the subsumed concept or upper concept of that topic. Briefly, using the ACM system cannot only save us a lot of time for designing the ontology, it also makes the ontology standardized and very clear in the computer knowledge topics.

3. Evaluation of the SWSAS search engine

We tested the utility and effectiveness of the SWSAS. We examined the results for a set of questions to verify that the SWSAS is accurate. We test the Simple Search and Advanced Search of SWSAS was able to assign exact result for the query. This

evaluation aims to ensure that all features expected by the users are met as well as to gauge the effectiveness of the search features. The utility evaluation was designed as a series of practical scenarios (or tasks) that a student will perform. Potential users in this case are the current students in university. In this test their task was to try and accomplish all the following 10 scenarios:

Table 5.1 Evaluation Questionnaire

1	I am interested in projects on AI (artificial intelligence), what is available? which professors knows the knowledge(teach or research in) of AI? There are too many projects on AI, can I see what are the subtopics of AI? (by Advanced Semantic Search)
2	What are the projects that be researched by a laboratory, e.g, the laboratory NanoRobotics? if projects are found, Can I get more information about this labotory, e.g. what is the web site of this laboratory? (by Advanced Semantic Search)
3	Continue with 2, are there other projects similar to this project or they have the same subject? Who is supervising this project? If I am interested in this project, what is the contact information of this supervisor? (by Advanced Semantic Search)
4	Which master degree classes are taught by professor Samuel Pierre? Does he teach class to the first cycle students? (by Advanced Semantic Search)
5	Who research in Multi-Agent System and he does not teach master student class? (by Advanced Semantic Search)
6	Professor Desmarais Michel research in which field besides the Artificial intelligence? (by Advanced Semantic Search)
7	How many professors in the department are associate professor and what are their names? (by Simple Semantic Search)
8	I do not know what I am interested in. Show me all the research domain offered in the department. (by Simple Semantic Search)

9	Find me professors that relate (teach or research in) to the following subjects: Linux/Unix. (by Simple Semantic Search)
10	I am interested in Semantic Web, I want to know the class or research information about it, I also want to know whether professor Michel Gagnon research and teach in Semantic Web (by Simple Semantic Search)

Five students participated in the test and evaluation of this search engine system SWSAS, everybody was given a questionnaire with the ten questions above, and everybody was required to finish the scenarios from the questionnaire independently. The results of the utility evaluation test is shown in the following table.

Table 5.2 Evaluation Result

Question	Experiment Result
1	All participants could find the project and subtopics of AI also find the professors of computer science department who has the knowledge of AI.
2	Participants could find the 3 projects in the laboratory NanoRobotics.
3	All participants found that Prof. Sylvain Martel is the supervisor for NanoRobotics. Also found the website address of the laboratory is http://www.nano.polymtl.ca
4	All participants found the 4 master classes which taught by professor Samuel Pierre, and found he teaches 2 first cycle classes.
5	4 participants found professor Quintero Alejandro, 1 participant did not found.
6	All participants found 5 other research domain of professor Desmarais Michel.
7	All participants found the list of the 12 assistant professors with their names.
8	All participants found the 223 research domain in the department computer

	engineering.
9	All participants found professor Dagenais Michel R.
10	All participants found there is not course in Semantic Web, but professor Michel Gagnon research in Semantic Web.

After their experiment, the test result including their feedbacks and suggestions are collected. All participants have the conclusion that SWSAS is very useful and efficient. The participants also commented that the two search style of “Simple Semantic Search” and “Advanced Semantic Search” are very convenient to use and meet their expected results. All the functions crucial for searching a class or research project are created and work correctly. Unfortunately there is a students did not get the answer for question 5, that is because the queries he input with serious misspellings, after he corrected his input he got the exact result. Overall, the application has fulfilled the utility expectations.

4. Comparison with the Current Website

On the website of École Polytechnique de Montréal, the search engine of Computer Engineering Department and the search engine of the university are different from each other; this is because they have two separated database. Search engine of SWSAS use databases which can be a global linked database and can make full use of the data resources by adopting ontology technology. We can imagine, if everybody uses the style of RDF or other ontology language to set up their website and share their knowledge base, and link all the resources together, people will be able to weave a global database which can be used everywhere and shared to everybody. This is also the exact aim of SW. Search engine of SWSAS can do the “concept” search instead of “keyword” search in its database and return intelligent and accurate answers to the users’ query. Here are

some example questions of users which searched both in the current computer engineering department website and in SWSAS:

Table 5.3 Comparison with Current Website

Question	Current Website of Polytechnique	SWSAS
<i>What is the telephone number of professor Martel Sylvain?</i>	At least 4 steps <method 1: by department search engine> Input 'Martel Sylvain' in the search engine of the department, we can get 4 links, click them one by one to check which web page has the information of telephone number. <method 2: by department web page> Click link 'professors' → check the name 'Martel Sylvain' → click on the 'Martel Sylvain' → check the telephone number → get the result	1 step. Input the query question in Simple Search or Advanced search and click the button "search", it returns the result.
<i>Which research field does professor Dagenais Michel R. research in ?</i>	At least 4 steps <method 1: by department search engine> Input 'Dagenais Michel R.' in the search engine of the department → we can get one link and click on it → enter the web page of 'professors and researchers' and check the name of 'Dagenais Michel R.' → click on the link 'Dagenais Michel R.' → enter the web page of 'Dagenais Michel R.' and check his research domain. <method 2: by department web page> Click link 'professors' → check the link of professor 'Dagenais Michel R.' → click on the link → check his research domain → get the result	1 step. Input the query question in Simple Search or Advanced Search and click the button "search", it returns the result.
<i>Which master degree classes are taught by professor Samuel Pierre?</i>	At least 8 steps <method 1: by department search engine> Input 'Samuel Pierre' in the search engine of the department, we can get several links, but none of them are related with the teaching and courses. <method 2: by department web page> If we click on the link 'professors', we can not find the	1 step. Input the query question in the Advanced Search and click the button "search", it returns the result.

	class information.....→ then we click on the link 'teaching'→ click on the 'graduated students' → click on the 'computer engineering master students' → to check the class taught by 'Samuel Piere'→ go back to last web page, and click on the 'model master students'→ to check the class taught by 'Samuel Piere'→ get the result.	
<i>Who teach the class LOG2000 and research in Software Analysis?</i>	At least 6 steps or 9 steps <method 1: by department search engine> Input 'LOG2000' in the search engine of the department website → click search and we can get one link→ click the link and check 'LOG2000' in the page→ click the 'detail' link of 'LOG2000' → check the professor who teach LOG2000→ check whether the same professor research in Software Analysis→get the result. <method 2: by department web page> Click the link 'teaching'→ select 'first cycle students'→ select 'computer engineering program'→ check 'LOG2000' → select 'software engineering program' → check 'LOG2000' → check which professor teach it (two professors) →check the links of the two professor to see which professor research in 'software analysis' → get the result.	1 step. Input the query question in the Advanced Search and click the button "search", it returns the result.
<i>which professors knows the knowledge(teach or research in) "artificial intelligence"?</i>	More than 15 steps <method 1: by department search engine> Input 'artificial intelligence' in the search engine of the department website, we can get four links. But none of them are related with teaching class. All of them are about the research work. So we can not get the result successfully by this method. <method 2: by department web page> Click 'teaching'→ select 'first cycle classes' → select 'software engineering program' → check 'artificial intelligence'→ go back to last page and select 'computer engineering program' → check 'artificial intelligence' → go back to the last page and select 'graduate students'→ check 'artificial intelligence' in 'diploma student class' → check "artificial intelligence"	1 step. Input the query questions in the Advanced Search and click the button "search", it returns the result.

	in 'computer engineering master classes' → check 'artificial intelligence' in 'model master classes' → get the class and professor teach 'artificial intelligence' → click on 'professor' → click on 'computer/system/robot /software' → check 'artificial intelligence' and click on the link → check the professor who research in this domain. → get the result of professor who teach and research in it.	
<i>How many professors in the department are associate professor and what are their names?</i>	many steps <method 1: by department search engine> Input the 'associate professor' (in French it is 'professeur adjoint') in the search engine of the department website, we can get one link return, then click on this link and open the web page, but on this web page we can not get the information who is associate professor. <method 2: by department web page> Click on the 'professors', one the web page of 'professors' click every link of the name list of the professors. We should click 28 links of all the professors to check if they are 'associate professor'.	1 step. Input the query question in the Simple Search and click the button "search", it returns the result.
<i>How many professors research in the domain Linux/Unix?</i>	many steps <solution 1: by department search engine> Input the keyword 'linux/unix' in the search engine of the department website (separated from the search engine of the university website), we can not get any result from it. <solution 2: by department web page> Click on the 'research' → select 'research domain' → get the links of every professor (it will be 28 steps if we click them one by one) → go back to the last page and click 'interest domain' → get the links of every professor (it will be 28 steps if we click them one by one to check Linux/Unix)	1 step. Input the query question in Simple Search or Advanced Search and click the button "search", it returns the result.

5.2 Comparison with other SW Search Engines

There are important differences between SW search engines and CW search engines. The SW search engines are based on keyword, the result depends on the keyword occurrence in the database. With CW search engines sometimes we are not able to get the exact information or answer to our question, and the whole search process needs to be supported by the user. SWSAS make the search job very convenient and efficient. Just input a word or define one or more clauses in the query and the search engine processes the query and gives us the result.

Currently, there are many other SW search engines, so what are the advantages of SWSAS over those search engines? In this section I compare SWSAS with other semantic web search engines and table 5.2 shows comparison of SWSAS with them:

1. OWL Semantic Search

This SW search engine support DAML, but it can not support OWL successfully yet. This search engine is in style of Triple Search and defines the clause with subject, predicate and object. But it has not the Simple Search style as SWSAS. It does not support multiple clauses search, neither could define the relations between the clauses. Its interface is very similiar to the Triple Search style search engine of SWSAS, they are both triple constructed clause search engine, but the interface of SWSAS looks more concise.

2. Swoogle

Swoogle is a crawler-based indexing and retrieval system for Semantic Web Documents (SWDs) which are written in RDF or OWL. Swoogle discovers, digests, analyzes and indexes online SWDs. Swoogle Search service which supports SWDs' URLs and the

classes/properties constraints. An Ontology Dictionary that indexes the classes and properties defined by the discovered SWDs.

Compared with SWSAS, we find that it can not do the concrete search in Triple Clause, it can not do the multiple clauses constraints to the search either. Without these two functions it could hardly search information in details and get the exact result which users need, so it is not as intelligent and accurate as SWSAS.

3. Search Engine for the Semantic Web (BETA)

This SW search engine supports RDF language whereas SWSAS supports not only RDF but also DAML and OWL. This search engine only has the function of Simple Search engine of SWSAS, and it has no function like Triple Search part of SWSAS. So what it can do is just simple semantic web search with RDF ontologies, it can not offer a service of concrete search just like triple clause query either.

4. The SHOE Search Engine

OWL is recommended by W3C as a standard rule for all the developers in the field of ontology or semantic web, this standard from W3C is widely accepted all over the world. OWL has many advanced properties that SHOE does not have. SHOE is another ontology language, but compared with OWL it is less popular. The main problem of SHOE Search Engine is it depends on SHOE ontology languages.

Another shortcoming of SHOE Search Engine is that there are not so many SHOE API for us to choose, so it is not convenient for the developer to use. On the contrary, there are many free API supporting OWL languages that people can select their preferred API. And this enhanced the efficiency of the development.

Table 5.4 Comparison of SWSAS with other Semantic Search Engine

Subject	OWL Semantic Search	Swoogle	Search Engine for the Semantic Web (BETA)	The SHOE Search Engine	SWSAS
Support OWL	No	Yes	No	No	Yes
Support DAML	Yes	Yes	No	No	Yes
Support RDF	Yes	Yes	Yes	No	Yes
Triple clause support	Yes	No	No	No	Yes
Multitple clauses support	Yes	No	No	No	Yes
Simple words Semantic Search	No	Yes	Yes	Yes	Yes
Multitple languages support	No	No	No	No	Yes
Semantic Web Statistic	No	Yes	No	No	No

5.3 Conclusion about SWSAS

From the comparison in table 5.2 we find that SWSAS is different from any other semantic web search engines, it has some advantages and powerful functions that other SW search engines do not have.

1. more convinient and efficient

In SWSAS two kinds of searches are available, Simple Semantic Search and Advanced Semantic Search engine. Simple Search engine offers general semantic answers to the query; Triple Search engine offer the clause search, offer users concrete results to the

query; which query to use depends on which kind of query the user wants to do. The Triple Search engine of SWSAS has the functionality of dealing with multiple clause queries. That means it can deal with several clauses of query at one time, users can use different clauses to limit the query in order to satisfy different kinds of search requirements.

2. more accurate and intelligent

SWSAS could return exactly the answer for the user's question without any complicated operation. OWL is used as the ontology language in this system, OWL could offer different kinds of DL relations to the classes or properties, SWSAS can make inference from the ontology data and get potential information from the data, then presents the result information to user. Another intelligence of SWSAS is that users can constraint their query by putting multi-clauses query in order to get their exact query.

3. more compatible and adaptive

It is compatible for OWL, DAML and RDF, almost all the ontology languages could be used in this search engine system, so SWSAS can make full use of the plenty of ontology data resources on the internet. Besides English, it also is compatible with French language, users can input French words to query the information.

4. more standardized and regular

The Ontology is built on the standard of "The ACM Computing Classification", so the ontology content is absolutely standard. OWL is recommended by W3C, and SWSAS adopt OWL as the ontology data storage. Jena 2.1 is developed by HP company and used in SWSAS as the ontology API in the system development. Program language is Java, so this system can be run in different operating systems.

In a word, SWSAS has many advantages, it also has disadvantage, there is one aspect where SWSAS cannot (yet) compete with general-purpose engines like Google: Scope. This is because in Google or other general purpose search engine one can search for any keywords in any domain. SWSAS can only search the information of the academic department. But if SWSAS really need a big scale database, then the general database technology can be used in ontology design and application. For example, we can use Oracle or Microsoft SQL Server database system in storing and developing ontology, no doubt this will enhance the efficiency and scalability of the ontology.

5.4 Challenges in developing SWSAS

In this research project I met several serious challenges. In order to make the system portable, we decided to use Java as the programming language in this project. In about four thousands lines of code, I uses a lot of Java classes, functions and programming skills and met many programming challenges.

I use Jena as ontology API. Jena has hundreds of classes and functions which the developers are supposed to know and use freely. Unfortunately there is not too many references or instructions materials about Jena on the Internet about how to program with it. There is a special group in www.Yahoo.com which is for discussing how to use Jena correctly. This group is supported by the creator of Jena and they can answer the questions of users on the Internet. Right now, thousands of questions have been raised there, from this we can conclude that Jena is not an easy tool to use, especially if we want to use it efficiently and properly. To use Jena as the API does not mean that Jena can solve all the problems. It can only function as the parser and reasoner of the ontology data. A lot of analysis process and data flow control needs to be added in the program. Jena is like bricks, how to use well the bricks to build a high building which has different beautiful structures depends on the skills of the programmer. Jena is a professional tool for the professional developer in developing SW. It is powerful and efficient but it is a complicated tool.

A reasonable and logical ontology is very important in my project. In this ontology we have to setup everything because we cannot find some other ontology available for reusing. In the development of the ontology I decided to make use of ACM system. But how to put ACM system into my ontology is another difficulty, because different universities have different computer engineering knowledge system, class content, research content; getting the ACM and the ontology corresponding to each other very well is not an easy job. The new ontology language, OWL, is becoming more and more popular, but right now we can not find many search engines which can support OWL very well on the Internet, that is because of the technological difficulties. In SWSAS using OWL as the ontology language was a challenge.

If the search engine can support multiple languages, the user can choose their favorite languages to query exactly what they want to query, it will be very convenient and efficient. On the other side, because this project takes a French university as the research background, it is quite necessary to make the search engine to be compatible with French language. So multiple language support is another challenge for this system.

CHAPTER 6

Conclusion and Future Work

The purpose of SW is to make the metadata serialize for processing or searching by the meaning of its content in website. It can overcome the shortcoming of the CW and make full use of the resource on the Internet. By SW data on the Web can be used on a global large scale.

Ontologies act as a central tool enabling flexibility, reusability and expressiveness of communication capabilities required by SW domain. Ontology language OWL is the newest technology for building Ontologies which is recommended by W3C and it is a widely accepted standard and common criteria of Ontology. OWL is not only an ontology language but also a description logic language; it makes the instances and instance properties more relevant in SW. OWL is adopted in our research project as the ontology language.

In this work, I proposed SWSAS, a semantic search engine system which can offer instant and accurate query. It is not only superior to the classical website, but also has some advantages which other SW search engines do not have. I compared SWSAS with other popular SW search engine on the web in order to show its advantages. Jena 2.1 is used and plays an important role in the development of this system. Jena2.1 is also compared with other ontology APIs. This system has two powerful query models that help the user to define exactly his query and get the query result; one is the Simple Semantic Search mode, the other is Advanced Semantic Search mode. This system SWSAS, is designed to extract data not only from specific pages, but the whole web on the Internet. It can also support multiple languages, such as French and English.

We created a procedure used for developing an ontology efficiently. With this

procedure, we developed an ontology for a computer engineering department, using the ACM computer science taxonomy.

As future work I intend to enhance SWSAS in the following aspects: deepening the Knowledge Management of SW to make it capable to summarize and automatically classify a document by concept; deepen the function of e-learning for the students and professors. SW has provided all means for e-Learning including ontology development and ontology-based annotations for learning material, their composition in learning courses and active delivery of the learning materials according to user input, experiences and heuristics. Definitely these are powerful e-learning functions in SWSAS. Another work which needs to be done is the evaluation of the ontologies. For example, if there are two ontologies which are using the same concepts but in different terms, how to distinguish or evaluate them? All of these would be considered in the next edition of SWSAS.

Reference

ACM(Association for Computing Machinery), The ACM Computing Classification System [1998 Version], <http://www.acm.org/class/1998/> , Last visit on 2005/1/10. [18]

Agent Semantic Communication Service (ASCS), OWL Semantic Search, <http://plucky.teknowledge.com/daml/damlquery.jsp>, last visit on 2005/01/18. [42]

BAADER. F, HORROCKS. I, and SATTLER. U, Description Logics as Ontology Languages for the Semantic Web, November 01, 2002, University of Manchester. [17]

BAADER. F, CALVANESE. D, MCGUINNESS. D, NARDI. D, and PATEL-SCHNEIDER. P, editors. The Description Logic Handbook. Cambridge University Press, 2002. [12]

BECHHOFFER. S, The DIG Description Logic Interface: DIG/1.1, Proceedings of DL2003 Workshop, Rome, June 2003. [34]

BERNERS-LEE. T, HENDLER. J and LASSILA. O, The Semantic Web, In *Scientific American* May 17 2001, 34-43. [1]

BERNERS-LEE. T, Semantic Web Road Mmap, September 1998. v 1998/10/14. [3]

BERNERS-LEE. T, SemanticWeb - XML2000, <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>. Last visit on 2005/01/08. [22]

Computing Technology Industry Association, XML Schema-A Business Tutorial, http://eidx.comptia.org/education/xml_schema.pdf, Last visit on 2005/02/08. [14]

COSTELLO. R, owl-genie, <http://www.xfront.com/owl-genie/>, last visit on 2005/01/08. [26]

DACONTA. M and OBRST. L, SMITH. K “ The Semantic Web - A Guide to the Future of XML, Web Services, and Knowledge Management”, June, 2003, ISBN: 0-471-

43257-1. [20]

DAML - The DARPA Agent Markup Language Homepage: <http://www.daml.org>. DAML+OIL (March 2001) language released by the Joint Committee on 27 March 2001. <http://www.daml.org/2001/03/daml+oil-index.html>. [7]

Department of Computer Science of University of Maryland. The SHOE Search Engine. <http://www.cs.umd.edu/projects/plus/SHOE/search/>, last visit on 2005/01/08. [45]

FENSEL. D, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin Heidelberg 2004, pp. 90-96. [35]

German Research Center for Artificial Intelligence, DFKI Competence Center Semantic Web, <http://ccsw.dfki.de/>. Last visit on 2005/01/18. [40]

GOLBECK. J, PARSIA. B, and HENDLER. J, Trust networks on the semantic web. In Proceedings of Cooperative Intelligent Agents 2003, Helsinki, Finland, August 2003. [24]

GOMEZ-PEREZ. A, FERNANDEZ-LOPEZ. M, CORCHO. O Ontology Engineering: with examples from the areas of Knowledge Management, e-commerce and the Semantic Web. Springer-Verlag London Limited 2004. [8]

HAARSLEV. V, MOLLER. R, WESSEL. M, Racer: semantic middleware for industrial projects based on RDF/OWL, a W3C standard. <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>. Last visit on 2005/01/08. [32]

HEIN. J, HENDLER. J, LUKE. S, SHOE: A Prototype Language for the Semantic Web, Department of Computer Science University of Maryland, December 12, 2000. [39]

HORROCKS. I, Description Logic Reasoning, <http://www.cs.man.ac.uk/~horrocks/Teaching/cs646/Slides/pt3-dlreasoning.pdf>, last visit on 2005/01/08. [30]

HORROCKS. I, SATTTLER. U and TOBIES. S, Practical Reasoning for Description

Logics with Functional Restrictions, Inverse and Transitive Roles, and Role Hierarchies EPSRC, Grant GR/L54516, the Esprit Project 22469 – DWQ and by the DFG, Project No. GR 1324/3–1. [31]

HP labs, Jena 2.1, <http://www.hpl.hp.com/news/2004/jan-mar/jena2.1.html>. last visit on 2005/01/08. [25]

HP labs, DAML API, <http://www.hpl.hp.com/semweb/daml.htm>, last visit on 2005/01/10. [27]

HP Lab Semantic Web Program, Jena – A Semantic Web Framework for Java , <http://jena.sourceforge.net/RDQL/>, last visit on 2005/01/08. [47]

Intellidimension Semantic Web Search. <http://www.semanticwebsearch.com/>, last visit on 2005/01/18. [44]

InfoEther and BBN Technologies, SemWebCentral Home Page, <http://www.semwebcentral.org>. Last visit on 2005/01/18. [33]

JENNIFER. G, JAMES. H, Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Networks - Proceedings of EKAW 04. [37]

KARAGIANNIDIS. C, SAMPSON. D, and CARDINALI.F, An architecture for Web-based e-Learning promoting re-usable adaptive educational e-content. Educational Technology & Society, 5, 4 (2002). [36]

Knowledge Representation, http://catalog.jbpub.com/pdf/Coppin_Chapter03.pdf, Last visit on 2005/01/29. [11]

MELNIK. S, RDF API Draft, <http://www-db.stanford.edu/~melnik/rdf/api.html>, last visit on 2005/01/18. [28]

MIND LAB at University of Maryland Institute for Advanced Computer Studies, MindSwap, <http://www.mindswap.org/>, Last visit on 2005/01/08. [38]

MINDSWAP, Pellet, <http://www.mindswap.org/2003/pellet/index.shtml>, last visit on 2005/01/08. [29]

MILLER. E, SWICK. R, BRICKLEY. D, Resource Description Framework (RDF), <http://www.w3.org/RDF/> v 1.168 2004/08/17. [2]

NATALYA-F. N and DEBORAH-L. M, “Ontology Development 101: A Guide to Creating Your First Ontology”. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. [46]

“OWL Pizzas” 7th international protégé conference , Bethesde, Maryland, 2004. [16]

RAPHAEL.V, KAON: the ontology and semantic web tool, <http://www.KAON.SemanticWeb.org>. Last visit on 2005/01/18. [41]

ROSSBERG. M, First-Order Logic, Second-Order Logic, and Completeness. First-Order Logic Revisited, Logos Verlag Berlin (2004), 303–321, <http://www.st-andrews.ac.uk/~mr30/papers/RossbergCompleteness.pdf>. [9]

Stanford University Encyclopedia of Philosophy, Model Theory, <http://plato.stanford.edu/entries/model-theory/>, Last visit on 2005/01/30. [10]

Stanford Medical Informatics. Protégé 2000. <http://protege.stanford.edu/index.html> Last visit on 2005/1/10. [19]

STOJANOVIC. L, STAAB. S, STUDER. R, eLearning based on the Semantic Web, <http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/WebNet2001eLearningintheSemanticWeb.pdf>, last visit on 2005/02/08. [21]

SHAH. U, FININ. T, JOSHI. A, SCOTT-COST. R, MAYFIELD. J, Information Retrieval On The Semantic Web . Copyright 2002 ACM 1-58113-492-4/02/0011. [23]

SIEGFRIED. H, Andreas.H, DANIEL. O, STEFFEN. S, RUDI S, YORK.S,

eBiquity Research Group, UMBC Swoogle, <http://pear.cs.umbc.edu/swoogle/index.php>, last visit on 2005/01/18. [43]

The World Wide Web Consortium, W3C (MIT, ERCIM, Keio), <http://www.w3.org/>, Last visit on 2005/02/08. [15]

University Stanford Knowledge Systems Lab, What is an Ontology , <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html> , Last visit on 2005/1/10. [4]

W3C , RDF Vocabulary Description Language 1.0: RDF Schema.
<http://www.w3.org/TR/2004/REC-rdf-schema-20040210/> v 1.0 2004/02/10. [5]

W3C Web Ontology Working Group, OWL Web Ontology Language,
<http://www.w3.org/TR/owl-features/> , Last visit on 2005/1/10 . [6]

W3Schools, XML Attributes, http://www.w3schools.com/xml/xml_attributes.asp, Last visit on 2005/02/08. [13]