UNIVERSITY OF CALGARY

Concept-Learning Supported Semantic Search using Multi-Agent System

by

Cheng Zhong

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

April, 2009

ISBN: 978-0-494-51171-8

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Concept-Learning Supported Semantic Search using Multi-Agent System" submitted by Cheng Zhong in partial fulfilment of the requirements of the degree of Master of Science.

Supervisor, Dr. B. H. Far
Department of Electrical and Computer Engineering

Dr. M. Moussavi
Department of Electrical and Computer Engineering

Dr. D. Krishnamurthy
Department of Electrical and Computer Engineering

Dr. Y. Hu
Department of Electrical and Computer Engineering

Dr. M. Ghaderi
Department of Computer Science

Date

Abstract

Currently, the mainstream of semantic search is based on both *centralized networking* that could be barrier to access trillions of dynamically generated bytes on individual websites, and *group commitment* to a common ontology that is often too strong or unrealistic. In real world, it is preferred to enable stakeholders of knowledge to exchange information freely while they keep their own individual ontology. While this assumption makes stakeholders represent their knowledge more independently and gives them more flexibility, it brings complexity to the communication among them. To solve this communication complexity, in this thesis, we present (1) a method for semantic search supported by ontological concept learning, and (2) a prototype multi-agent system that can handle semantic search and encapsulate the complexity of such process from the users. The method introduces a layered structure of semantic interoperability to guide the agents' communication. Agents, which conduct semantic search on behalf of users, deploy ontologies to organize structured and unstructured documents in their corresponding repositories. The ontology for each repository is individualized and commitment to a common ontology is not required. Using this method, the agents can improve their search capability by learning new concepts from each other, and in return, agents can also identify new concepts under the help of semantic search. This method thus allows agents to dynamically establish common grounds to retrieve documents related to a given concept.

Acknowledgements

There are many people to whom I wish to express my gratitude and without them I could not have succeeded in this endeavour.

First, I would like to express my sincere gratitude to Dr. Behrouz Homayoun Far, my direct supervisor and mentor, who provided me with the opportunity to study under his instruction in his lab. He is the person who led me to this wonderful front-edge research area. He taught me how to think logically in order to solve problems. His support, patience, kindness, understanding and encouragement throughout my program were fundamental to my success. Without his help, I would not have gotten this far. He has taught me many things, not only in science, but also in personal situations, which is most important in my life. I am very pound to have been your student. I am unable to define all that you have done for me and I am sure what I have learned in the past four years will continually influence me throughout my life.

I would like express my utmost appreciation and gratitude to the examiners of this thesis,

Dr. M. Moussavi, Department of Electrical and Computer Engineering

Dr. D. Krishnamurthy, Department of Electrical and Computer Engineering

Dr. Y. Hu, Department of Electrical and Computer Engineering

Dr. Majid Ghaderi, Department of Computer Science

for their generous assistance and patience in reviewing my work and providing advice.

Thanks to my fellow members of research team, Mohsen Afsharchi who has already obtained his Ph.D degree and M.Sc. candidate Zilan Yang whose constructive suggestions always inspired me. I also would like express my appreciation to four other

members, Benjamin Hsieh, Jessie Love, Albert Luong, and David Piepgrass, for their supports and helps on exploration of JXTA with their unforgettable works, in their 4th year of BSc. Study.

Finally, and most importantly, I would like to thank my all family (my parents, and my elder sister) for their unconditional love and encouragement during this challenging time in my life. This project is a testament to the strength of their exceptional support and kindness. They are the most wonderful parents in the world! I am grateful for my sister Ye Zhong for being there companying my parents when I was absent.

At last, I want to specially thank one of most important person in my life, my wife, Yichun Sun, for her love and support in everything I do during my four years of study. She always gave me good suggestions and shared her experience gained from her five-year study pursuing Ph.D. degree with me. She encourages and supports me all the time on all aspects. Thank you very much, Yichun!

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
$\mathcal{Ag}$	Represents the agent
$\mathcal{Act}$	Actions taken by agents
$\mathcal{Dat}$	Internal states of agents
$f_{\mathcal{Ag}}$	Agent mapping function
$\mathcal{Sit}$	Situations in which the agent can exist

Abbreviations and Acronyms:

AE	Analysis Engine
AI	Artificial Intelligence
APQC	American Productivity & Quality Center
CAS	Common Analysis Structure
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DF	Document Frequency
GUID	Globally Unique Identifier
JXTA	Juxtapose
KAON	KArlsruhe Ontology
KIF	Knowledge Interchange Format
KM	Knowledge Management
KMS	Knowledge Management System
MAS	Multiagent System
MVC	Module-View-Control
OASIS	Organization for the Advancement of Structured Information Standards
OCML	Operational Conceptual Modeling Language Layer
OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity protocol
OLAP	On-Line Analytical Processing
OO	Object-Orientation
OWL	Web Ontology Language
P2P	Peer to Peer
RDF	Resource Description Framework
ROD	Ratio of Disambiguation
UIMA	Unstructured Information Management Architecture
UML	Universal Modeling Language
URN	Uniform Resource Name
UUID	Universally Unique Identifier
W3C	World Wide Web Consortium
WebODE	Web Ontology Design Environment

Chapter One: INTRODUCTION

In this chapter we present motivations and goals of the research team, together with the research problems that our team aims to solve. Also the detail contribution to the current state of the art of semantic search is presented. And, finally, we present overview of the prototype system to fulfill the research goal.

1.1 The Goal and Research Problems

The general goal of our research team is divided into three sub-goals, (1) algorithm(s) of concept learning, (2) methods of concept learning verification, (3) a cooperative search engine and supporting MAS as described in Figure 1-1. This thesis focuses on the third one, creation of MAS supported search by taking advantage of achievement of concept learning and concept learning verification.

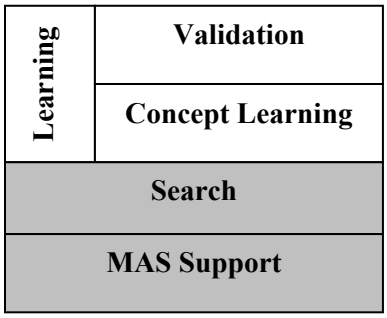


Figure 1-1: Outline of General Goal of Research Team

The work dedicated to the method of *verification of concept learning* was presented in [Far, 2007]; and the work dedicated to the algorithm for agents to *learn concepts* from several peer agents was presented in [Afsharchi, 2006], [Afsharchi, 2006A] and [Yang, 2008]. Comparing with existing approaches, the algorithm of concept

learning devised within our research group is enhanced on two aspects: group learning and feature diversity [Afsharchi, 2007].

The goal of this thesis is to advance the current state-of-the-art of semantic search by providing a model addressing implementation of multi-agent system (MAS) serving semantic search under the support of concept-learning, and to implement a prototype system conforming to this model. In order to achieve the goal, the following objectives must be reached:

1) Devising individual autonomous agents that are capable of learning ontological concepts from several teacher agents through interaction with other agents and validating these concepts to better communicate and share information in the future.
2) Developing a semantic search engine capable of dynamically annotating the data repositories that they are handling within MAS.
3) Integrating method or mechanism needed to support and facilitate the implementation of complex interactions among agents.

To achieve the first objective, ontological heterogeneity within multi-agent systems must be solved. This is directly caused by the fact that any ontology of certain domain can evolve independently. Therefore the only way for agents with diverse views of the world to understand each other is being able to understand each other's conceptualization of the domain, and then to share knowledge. Previous works about agents' communications mostly assumed a complete common understanding of the concepts used to depict domain knowledge and/or common language(s) among agents used to depict the concepts. However, in the real world, in order to initiate communication, agents may not be able to settle down common conceptualization of a certain domain first. For those agents who do

have common conceptualization, the first time contact still requires being aware that they have a common conceptualization in some way. This fact is well summarized with the point that any conceptualization of the world is accommodated, and is invented based on its utilization [Genesereth, 1987]. Nowadays, solutions of having agents to learn concepts from each other are getting more interest to the problem of ontology heterogeneity [Williams, 2004] [Steels, 1998].

To achieve the first and second objectives, a *semantic search engine* that continuously interacts with the *concept learning* is required. The interaction between them has the form of a spiral-like workflow as depicted in Figure 1-2. On one hand, search engine should be capable of responding to the requests according to agreements with concept learning module. On the other hand, annotation procedures of semantic search engine can be done on the fly based on the newly learnt concept instead of fixed predefined ontological concepts.

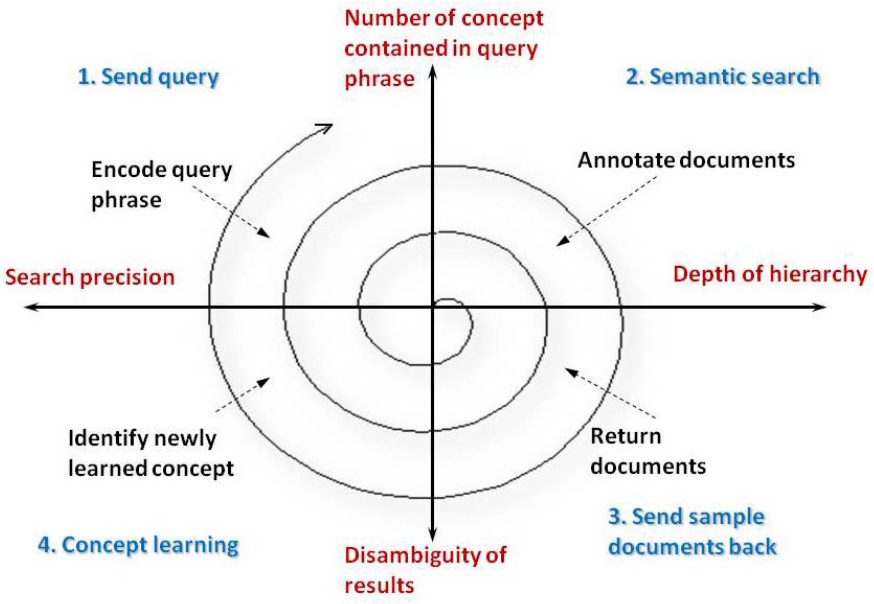


Figure 1-2: The Spiral-like Procedure of Concept Learning and Semantic Search

To achieve the third objective, the problem of integration and communication among agents raised by multiplicity of ontologies/conceptualization need to be fixed. As the essence of semantic search is semantic interoperability among agents towards denotation part contained in the search expression, semantic search is expected to be able to take advantages of concept learning to establish an integrated mechanism to help find common understandings of concepts, and based on it, higher-level modalities of ontology may accomplish interoperations with respect to those denotations.

1.2 Contributions

This thesis is aimed at giving solution to achieve the third objective listed in the previous section while integrating with the results of the first and second objectives.

With regard to the objective, the followings are original contributions of this thesis:

- **A novel description exposing intrinsic relationship between concept learning and semantic search in an ontological heterogeneous environment.** In such environment, concept-learning and semantic search are treated equally as basic roles, involved in spiral-like evolving procedure (shown in Figure 1-2), which support each other to achieve their own goals by enriching the set of ontological concepts and increasing disambiguity of the search, respectively. Following the spiral-like route, concept-learning module and semantic search engine take actions alternately to approach their goals.
- **A layered architecture for semantic interoperation architecture** [Zhong, 2008] that facilitates implementation of semantic interoperability through "divide-and-

conquer" strategy. The results of this work show that implementation of the multi-agent system following the architecture is achievable.

- **A MAS prototype system** that fulfils the basic workflow is developed under support of UIMA [Zhong, 2008].

1.3 Motivation

1.3.1 The Significance of Communication

The main motivators that prompted this thesis are:

The effort expended on solving ontological heterogeneity deviate from the inherent cognitive principle of knowledge that understanding comes from learning-oriented communication in which learning subjects acquire knowledge. Generally speaking, learning consists of two fundamental parts, learning ability (or algorithm) and learning procedure. The learning ability as internal factor addresses functions of perceiving new knowledge. The learning procedure as external factor addresses learning course such as interactive mode, and protocols. Currently, some popular projects [Berners, 2001] take the strategy, *understanding comes from standardization*, standardizing knowledge representation by developing complex ontology languages [Gomez, 2002], and some projects (e.g. [Corcho, 2005]) then standardize translation between those ontology languages. These approaches impose a strong or unrealistic assumption on the working environment that is common ontology (these approaches are reviewed in Chapter 2). When analyzing the process of learning language, either person to person or computer, we believe that being able to communicate, instead of using common ontology, is a prerequisite to understand each other. Current considerations regarding people speaking very different languages and still being able to understand each other with the help of

sign language as translators is the evidence. One might explore where and when the first languages developed and how they bridged communications. My opinion is that some methods other than languages are also utilized to understand one another. From the perspective of cognitive sciences, natural language normally carries ontology towards the world; therefore, we can say that human being already achieved the learning between peoples who adopted different ontologies long time ago. For computer-based learning, when examining learning methods such as supervised, unsupervised, reinforcement, and our concept learning method, I find that effective communication still plays a critical role as they are all relying on specified inputs fed by outside information providers who may use completely different ontology. I believe that more attention on the part of external communication should be paid to fix the problem of ontological heterogeneity.

The search responsibility centralized on small number of search engines (e.g. Google, Yahoo, etc.) should be distributed on several nodes which are willing to share their resources with other nodes. Compared with the centralization of knowledge and resources distributed search service has the following potential advantages:

- Direct access to the distributed resources. The search engine on each node can "see" the resources, it would be able to provide accurate and up-to-date search results;
- The task of indexing or annotating local data repositories with local ontologies would be better managed than the centralized ones.

1.3.2 The General Course of Learning Communication

Knowing the importance of interactions for searching, learning and knowledge sharing in an environment with ontological heterogeneity, investigation of the interactions will facilitate implementing effective communications among agents of a MAS prototype that

are dedicated to learning/searching concepts. With respect to the scenario of our MAS prototype, interactions is to perceive semantics of a concept, a phrase, or even relationships of concepts (in the future), so that the core of interactions is to achieve semantic interoperability.

New challenges are dealing with a lack of standards for interoperations. We need to ask "what can be taken as standard elements to facilitate semantic interoperation and guide agent communication?" We believe that a layered structure inherently existing in semantic interoperability and ontology could be the answer to this problem.

Euzenat presented possible levels of semantic interoperability needed to be considered when trying to understand an expression from other systems [Euzenat, 2001]. They are in ascending order of semantics: *Encoding, Lexical, Syntactic, Semantic, and Semiotic* (see Chapter 2 for details). In addition, layered architecture has been defined in the semantic web [Decker, 2000]. It is argued that, ontology, as a means for achievement of semantic interoperations, also has inherent layers ranging in terms of semantics intensity. In [Daconta, 2003] a spectrum of ontology semantics had been proposed which ranges from the simple notion of a taxonomy, to a thesaurus, to a conceptual model, and to a logical theory.

In this thesis we define a layered architecture for agent interaction (see Chapter 3). The proposal of applying layered architecture to interaction of semantic search and concept learning is an original contribution of this thesis. It is similar to the natural communication of humans that each semantic level would be achieved only if the lower ones have been implemented. For example, people can exchange useful information only

if they speak a common language, and clarify the meanings of concepts which are critical to the topic of conversation.

In our proposed MAS prototype, one of the modules is a semantic search engine. It would operate with another module, concept learning, at every layer. The approach in this thesis focuses on the lexical layer (or concepts) with the focus on developing and evolving a taxonomy of concepts. Accordingly, the search engine focuses on filtering special combinations that can identify attributes of concepts. This type of combination resembles water molecule as a combination of two hydrogen atoms and one oxygen atom. A complex search is composed of words representing concepts and/or interesting features of the concepts under pre-defined rules. This combination can go beyond a simple set of keywords.

1.4 System Overview

As stated above, the goal of this thesis is to devise a conceptual model for semantic interoperation between concept learning and semantic search, and under guidance of this model to implement a MAS prototype system. Given the fact that agents have not common knowledge, problems will rise when agents are working together. To solve this problem, agents need to acquire the concepts outside of the base concepts that other agents may have, at least those concepts that are needed to establish the necessary communication to work together through interactions. Each MAS is expected to have two main agents: concept learner and semantic search engine. P2P networking is selected to implement communication among agents. IBM (later Apache) UIMA is taken as development platform to support dynamic document annotation within semantic search engine.

1.5 Thesis Overview

In Chapter 1, we outline the basic problem that this research addresses, delineate our research goals, and make contributions clear.

In Chapter 2, we define the general notions that we use throughout this thesis. We give semi-formal definitions of agent, ontology, semantic search, and concepts. We also explore the literature related to ontology languages and tools of current stage and semantic search approaches.

Chapter 3 talks about conceptual layered semantic interoperability and analysis and design of prototype system using *GAIA methodology for multi-agent system analysis and design* [Wooldridge, 2000].

Chapter 4 contains the important aspects of the implementation of the prototype system, which accomplishes one round of spiral workflow of interoperation at lexical level. A simple application is developed to demonstrate a typical search procedure.

In Chapter 5, experiments are performed to demonstrate the correlation between concept learning and semantic search, and how query results are influenced by operations of concept learning and semantic search.

Finally, Chapter 6 contains conclusions of the research presented in this thesis, and suggestions for future work.

Chapter Two: Background

In this chapter we present a general background as well as definitions of concepts critical to the research, including agent, ontology, semantic search, concepts, etc. We will also review the literature to explore current state of the art of ontology languages and tools, and semantic search. In addition, we will outline the open source UIMA, which is utilized to support the implementation of the MAS prototype.

2.1 General Background

The general background of the issue that this thesis addresses is the problem rising from "knowledge sharing" in the application area of knowledge management (KM). Peter Drucker as one of the world's KM masters said "Knowledge is the most valuable property in the enterprise". Since the 1990's KM has received a lot of attention in scholarly, professional services firms and business organizations of all industrial sectors. A number of management consulting firms had begun their internal knowledge management programs including several worldwide famous U.S., European, and Japanese firms, such as American Productivity & Quality Center (APQC) and Ernst & Young. The concept of knowledge management was also introduced to the public in 1991 [Stewart, 1991]. From mid-1990 to early 2000, with the development of Internet, knowledge management initiatives were flourishing. Some pioneering companies developed initial knowledge management system(s) (KMS), for example, in 2002, Nortel Networks released supportive KMS "virtual mentor"; in 2001, IBM released Intellectual Capital Management tools to support the exchange of knowledge in a global environment [Maier, 2004]. Some vendors are also offering products to help an enterprise to manage inventory and access knowledge resources. IBM's Lotus Discovery Server and Oval, for

example, are products advertised as providing the ability to organize and locate relevant content and expertise required to address specific business tasks and projects [Daconta, 2003]. Estimates at leading research organizations show that managing knowledge represents the primary opportunity for achieving substantial savings, significant improvements in human performance, and competitive advantages.

2.1.1 Knowledge and Knowledge Management System

There are several definitions of "knowledge" with varying perspectives. Philosophical debates in general start with Plato's formulation of knowledge as "justified true belief" [Roderick, 1982]. Presently, there is however, no single agreed definition of knowledge, or any prospect of one. There is a variety of definitions of knowledge; some critical characteristics of knowledge obtain wide recognition. Kant says "knowledge is 'Human-embodied'. Philosophy tells us that there is no knowledge outside the possibility of experience" [Kant, 1965]. Experience is the human being's experience. Another critical characteristic is *Sociability*. Knowledge always occurs within a context of social activity by communication, inference, etc. Thus, by considering the goal of this research, the thesis made the conclusion that knowledge is structured, semi-structured, and unstructured data globally distributed; it represents facts, information, and skills acquired by a person through experience or education in a domain-specific organization, and has been recorded in some formats, recognizable to computer systems and potentially sharable among intelligent agents.

2.2 Important Concept Definitions

2.2.1 Agent and Multi-Agent System

The field of Agent Oriented Software Engineering (AOSE) and Multi-Agent Systems (MAS) are derived from many disciplines such as artificial intelligence (AI), Object-Orientation (OO), parallel processing, distributed computing, human-computer interface, and mobile code. Due to the diverse origins, there is no definition for agent considered universally accepted. Generally, an agent is a system that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [Russell, 1995]. A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent is equipped with cameras and infrared range finders for the sensors and with various motors for the effectors. A software agent has encoded bit strings as its percepts and actions [Russell, 1995].

We prefer to define agent with a mapping function inspired by [Denzinger, 2002] which takes arguments from perceived sequences to actions. The function is

$$\mathcal{A}g = (\mathcal{S}it, \mathcal{A}ct, \mathcal{D}at, f_{\mathcal{A}g})$$

$\mathcal{S}it$ is a set of situations in which the agent can exist. That means the agent's world can be in any one of a set $\mathcal{S}it$ of situations (or states). The representation of a situation naturally depends on the agent's sensory capabilities. Different agents can have different effective capabilities. To characterize these effective capabilities, we assume the existence of a set $\mathcal{A}ct$ of actions, all of which can be performed by the agent we are describing. We define a function $f_{\mathcal{A}g}$ as a mapping that maps each situation into an action. This function is called effective function **[**Michael ,1987**]** and is defined as

$$f_{Ag}: Sit \times Dat \rightarrow Act$$

We further assume that the agent can be in any one of a set Dat of internal states. While there is no need for internal states in a simple case, the ability to retain information internally is extremely useful in general. Therefore, there should be a memory update action in Act that maps an internal state to another internal state. This function defines an agent by exploring its internal structure.

As agents may possess abilities of negotiation, cooperation, learning, etc., agents and multi-agent system are expected in this thesis to have the ability of knowledge level communication. The actions that an agent will take obviously depend on the area that the agent is assigned to serve. Therefore, within the scope of thesis, the agents are required to be able to communicate with other agents with different ontologies through agents' actions including learning action. An element of *Sit* usually contains elements representing observations of agents towards the environment in which the agent is located. In this thesis, the establishment of the solving scheme is relying on the existing methodologies and theories.

2.2.2 Ontology

Ontology, as a popular term, is widely used in the fields of Artificial Intelligence, Knowledge Engineering, e-commerce, bio-informatics, education, and Semantic Web. In KM applications, ontologies are typically used for purposes:

- Semantic search
- Agent mediated knowledge management
- Data and knowledge translation
- Business process modeling

- Internet information processing

The concept ontology, originating from philosophy in early Greece, deals with the nature of being or existence. In the 1990's, ontology was adopted by AI community. Since then, like the term *knowledge*, multiple definitions of "ontology" had been proposed in the literature. The preferred definition consistent with the scope of this thesis is that "ontology is a formal, explicit specification of a shared conceptualization" [Studer, 1998]. "Formal" here means that the ontology should be recordable, computer-system readable. "Conceptualization" is a modeling procedure to create abstract model of concepts in a certain domain and relationships among those concepts. "Explicit specification" means properties, types of concepts to be modeled and the constraints applied to them should be explicitly defined. "Shared" addresses the fact that an ontology extracted from consensual knowledge need to be acceptable in communities and exchangeable among them. Therefore, the ontology in KM should bear the following characteristics, *formability, definability, publicity, and interchangeability*. Then ontology can help exchange and share knowledge with the help of information and communication technology (ICT) systems.

As described in Figure 2-1 [Daconta, 2003], there is an inherent semantics spectrum of ontology ranges from taxonomy to logical theory reflecting relationships between taxonomy, thesaurus, and conceptual model. In this thesis we focus on taxonomy as a form of ontology with focus on concepts with simple relations.

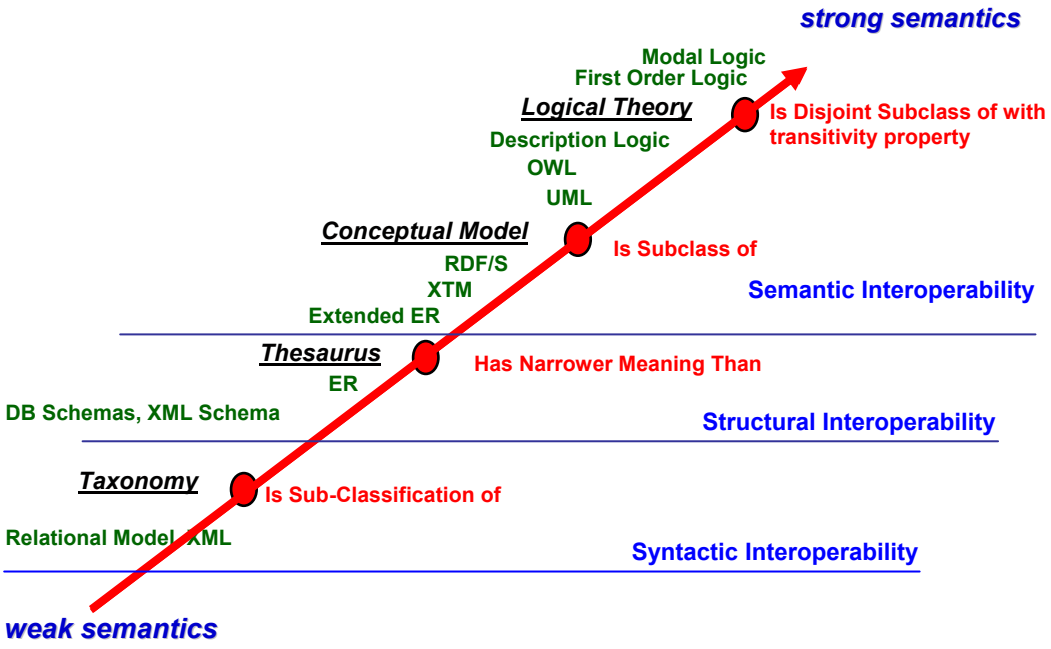


Figure 2-1: The Ontology Spectrum [Daconta, 2003]

2.2.2.1 Taxonomy

The term taxonomy denotes the classification of information entities in the form of a hierarchy, as illustrated in Figure 2-2, according to the presumed relationships of the real-world entities that they represent [Daconta, 2003].

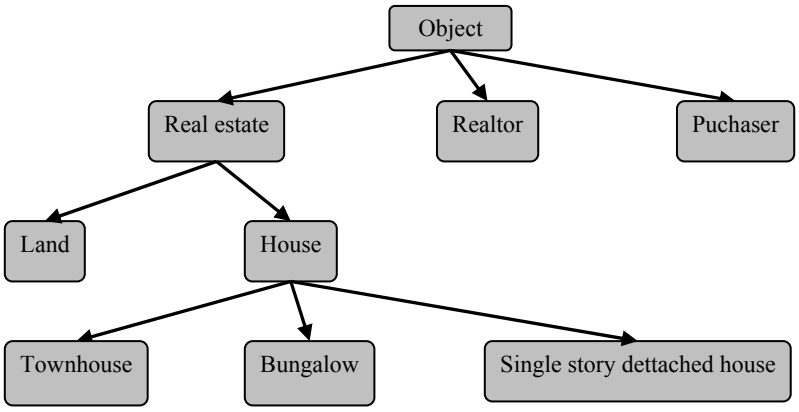


Figure 2-2: An Example of Taxonomy

Taxonomy can be modeled as a hierarchy of terms, and functions as the semantic basis for searching and visualizing a domain, e.g., a collection of documents. Taxonomy can contain definitions of terms, and, like a thesaurus, contain explanations, synonyms, homonyms and antonyms.

2.2.2.2 Thesaurus

From the point of view of Library and Information Science, a thesaurus is a "controlled vocabulary arranged in a known order and structured so that equivalence, homographic, hierarchical, and associative relationships among terms are displayed clearly and identified by standardized relationship indicators" (ANSI/ISO Z39.19-1993[R1998], p. l). No doubt, it is adopted to facilitate the document retrieval. In KM thesaurus denotes a list of important terms in a given domain of knowledge and a set of related terms for each term in the list. A simple example of thesaurus is depicted in Figure 2-3.

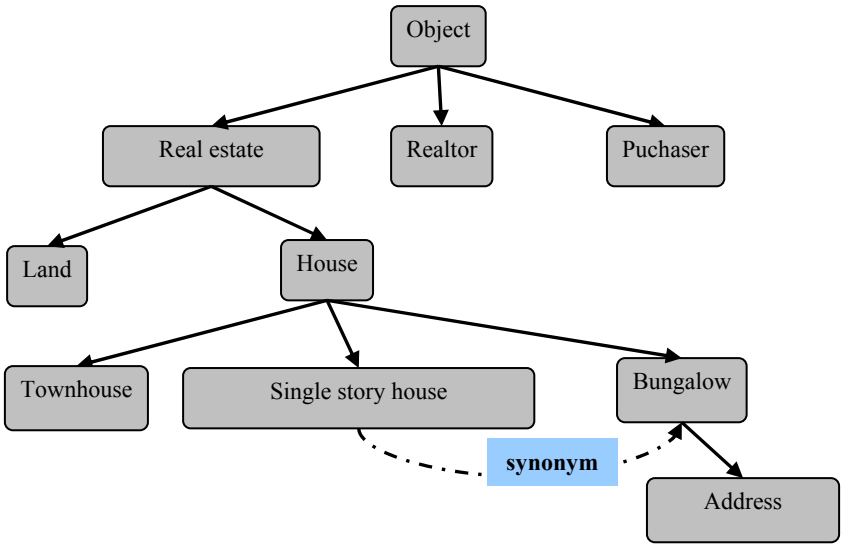


Figure 2-3: An Example of Thesaurus

2.2.3 Comparison among Taxonomy, Thesaurus and Ontology

Ontology can range from the simple notion of taxonomy, to thesaurus, to conceptual model, and to a logical theory as shown in Figure 2-1. Taxonomy contains structured concepts and their relationships as simple as "sub-class" or "is part of". Thus, taxonomy plays a crucial role for creation of ontologies by providing a *semantic basis* which acts as a skeleton of ontology. After the addition of complex relations, consequently, the body of ontology will emerge.

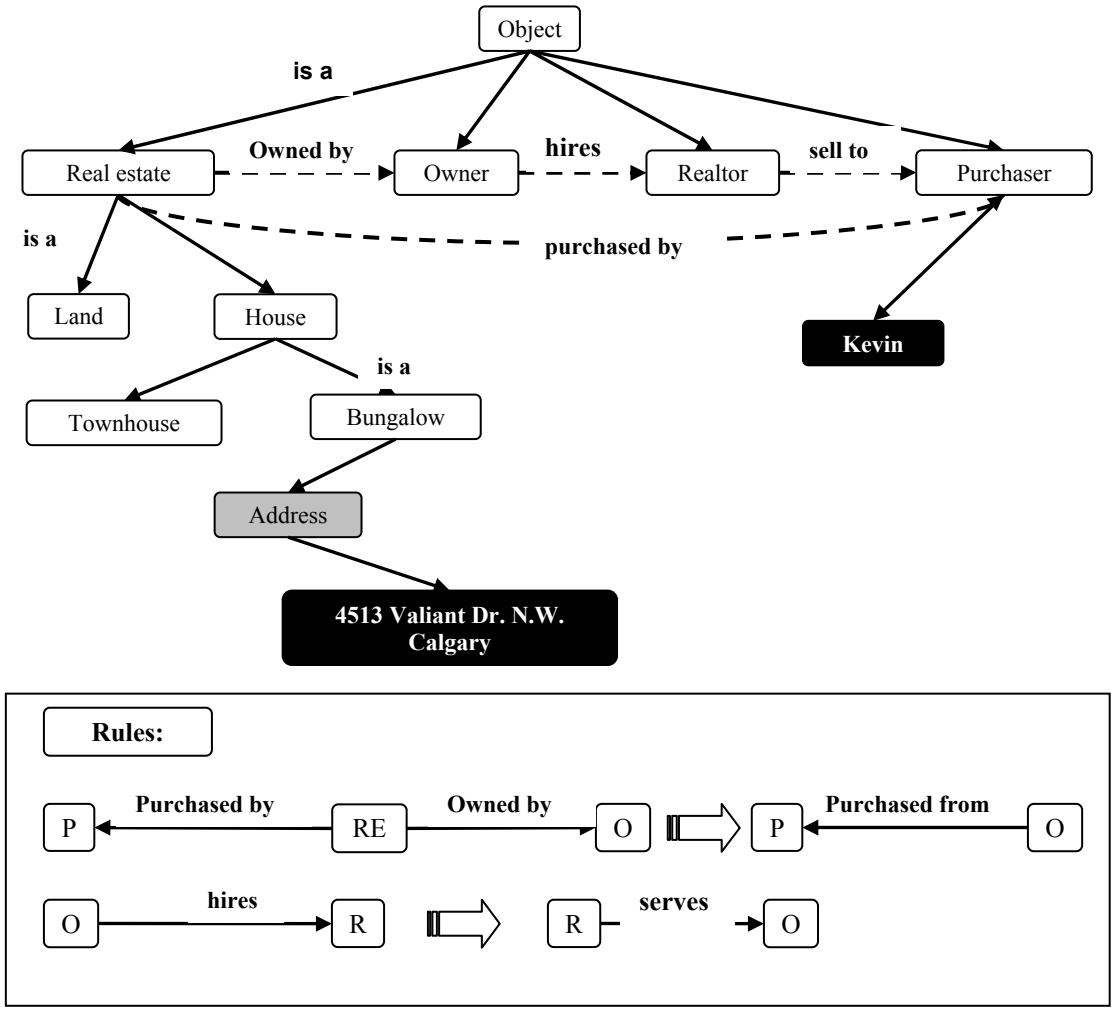


Figure 2-4: An Example of Ontology

Thesaurus is semantically stronger than taxonomy, but weaker than a conceptual model in which terms are connected through semantic links that are relationships between these basic concepts, including synonym and hierarchical relationships.

The synonym generally denotes the relationship like "what is what", meanwhile the hierarchical relationship represents generalization (broader) or specification (narrower) of a term in scope. Conceptual model has higher expressivity than both taxonomy and thesaurus. A conceptual model is modeling a certain domain instead of local parts by representing primary entities of domain, their relationships, properties of the entities, functions involving those entities and constraints on and rules involving those entities. Figure 2-4 shows this model. Logical theories of local domain represent a complete form of ontology which can be directly interpreted semantically by the software.

2.3 Ontology Languages and Tools

During the last decades, several ontology languages have been created which can be divided into two groups, *traditional ontology languages* and *web-based ontology languages*.

2.3.1 Traditional Ontology Languages

Traditional ontology languages have emerged since the beginning of 1990s. These languages are based on first order logic including KIF, CycL, Ontolingua, OCML, Flogic, OKBC, and LOOM. Some of them, e.g. KIF, are extinct, i.e. not used actively and not maintained any longer; some of them are used in a few systems, and protected by research groups such as F-logic; some of them including OKBC and LOOM, even though still are actively supported, have not been upgraded since the Web boom [Gomez, 2002].

- KIF: Knowledge Interchange Format

- OKBC: Open Knowledge Base Connectivity protocol
- OCML: Operational Conceptual Modeling Language

2.3.2 Web-based Ontology Languages

Quick growth of the Web triggered creation of a new generation of ontology languages aimed at exploiting resources from the Web, at sharing resources globally via the Web. The linguistic basis of these languages is based on markup languages such as HTML and XML. Figure 2-5 illustrates some important Web-based ontology languages. Moving up and left to right the figure shows the evolution of these languages.

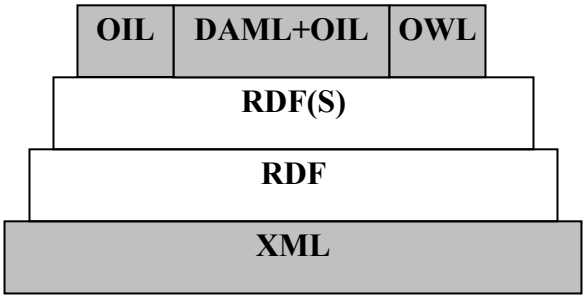


Figure 2-5: Web-based Ontology Languages

RDF [Decker, 2000] was developed and proposed as a W3C (the World Wide Consortium) Recommendation in 1999. It is a semantic-network-based language to describe Web resources. From 2000 to 2004, W3C proposed, revised, and finally published final Recommendation of RDF Schema [Fleming, 2005] as an extension to RDF language. The combination of both RDF and RDF Schema is subsequently known as RDF(S). These languages have formed the foundation of the Semantic Web [Daconta, 2003]. Since then, as extensions to RDF(S), three more languages have been developed

including OIL, DAML+OIL, and OWL. OIL [Lassila, 1999] adds frame-based KR primitives to RDF(S) and its formal semantics are based on description logic.

DAML+OIL [Brickley, 2001] was created in 2001 by joining DAML-ONT language and OIL language in the context of the DARPA project DAML. DAML+OIL adds DL-based KR primitives to RDF(S). In 2004, the W3C established a work group called Web-Ontology (WebOnt) Working Group aimed at making a new ontology markup language for the Semantic Web. The achievement of their work is the OWL language [Dean, 2004]. The OWL covers most of the features of DAML+OIL and renames most of the primitives that appeared in that language.

2.3.3 Ontology Tools

Following the ontology languages, in the mid-1990s, ontology tools were developed to ease the task of building ontologies. In practice, building ontologies is complex and time consuming without any type of tool support. The Table 2-1 is adapted from "*OntoWeb. D1.3. A survey on ontology tools, funded by the IST Program of the Commission of the European Communities*" [Gomez, 2002], in which the major ontology tools, since 2000, and their interoperability are listed. The survey concluded that "a lot of similar ontology development tools exist for the building of ontologies, but they do not interoperate and do not cover all the activities of the ontology life cycle (just design and implementation)" [Horrocks, 2000]. Horrocks mentioned that "The lack of interoperability is an obstacle to integrate ontologies built with different tools. Consequently, much knowledge can be lost during the translation process or can be translated in such a way that it is difficult to process by the target tool" [Horrocks, 2001].

Table 2-1: Major Ontology Development Tools (adapted from "OntoWeb. D1.3. A survey on ontology tools) [Gomez, 2002]

Features	Protégé	OilEd	OntoEdit	WebODE	KAON
Year released	2000	2001	2002	2003	2003
Developers	Stanford University	University of Manchester	Institute AIFB, University of Karlsruhe	Technical School of Computer Science (FI) Madrid	Institute AIFB, University of Karlsruhe
With other tools	PROMPT, OKBC, ArgoUML	FaCT	OntoAnnotate, Ontobroker, OntoMat, Semantic, Miner	JESS, PICSEL, OILEd, ODEMerge, ODE-KM	KAON-Portal
Import(s)	XML, RDF(S), OWL, XMLS, XMI	RDF(S), OIL, DAML+OIL	XML, RDF(S), Flogic, DAML+OIL	XML, RDF(S), CARIN	KAON, RDF(S)
Export(s)	XML, RDF(S), OWL, FLogic, CLIPS, Java, XMI	OIL, RDF(S), DAML+OIL, SHIQ, Dotty, HTML	XML, RDF(S), Flogic, DAML+OIL	XML,RDF(S), OIL, DAML+OIL, CARIN, FLogic, Prolog, JESS, Java, HTML	KAON, RDF(S)

2.3.4 Summary

From the brief review of the ontology languages and tools, we can learn that the main problem in the current application of ontology is incompatibility (lack of interoperability) on the ontologies developed individually in different organizations, and it could be an obstacle to search and/or share knowledge across the organizational boundaries.

Even though a wide variety of ontology languages and ontology tools have been developed for building ontologies, and most of tools share one or more import/export languages, they are not able to offer the strong interoperability that KM community expects. Corcho has mentioned that intrinsic complexity of ontologies led to the situation that interoperation among different ontology tools and languages is not as simple as transforming syntactically different formats, but also has to deal with higher-level transformations to ensure semantic and pragmatic preservation of knowledge in the source format and its intended meaning [Corcho, 2005].

2.4 Semantic Search

Semantic Search has been known as the most important application of Semantic Web [Guha, 2003]. Initially, Tim Berners-Lee proposed semantic web as the next-generation web, aimed at machine-processible information [Berners, 1999]. It implies that all the resources on the web and all kinds of semantic relationships among them must be understandable for the machine. As an extension of the current web, the semantic web will contain not only simple relation (i.e. the hyperlink) between resources, but also contain many rich kinds of relations between the different types of resources such as people, places, organizations and events, and these types of new resources will be useful for more effective discovery, automation, integration, and reuse across various applications. Semantic search by definition integrates the technologies of semantic web and search engines, attempts to augment and improve traditional search results. In semantic search, concepts (i.e. keywords) contained in a query phrase typically are denoted, so that it would help the search engine to understand a concept in the context of (or with relations to) some other concepts. The denotation enables the search engine to understand and filter the context of search, the activity that is currently performed by the user manually. Compared with the traditional web search, semantic search has two advantages: *disambiguation of query* and *increase of relevance of search results*.

2.4.1 Roles of Ontology Languages in Semantic Search

The W3C consortium has recommended separation of developing ontology languages which deal with aspects of both syntax and semantics. As a result, Resource Description Framework (RDF) was developed for knowledge representation, and then, Ontology Web Language (OWL) was announced for ontology description, which was supposed to

describe classes of objects, their properties and relationships in some domain. Along with the appearance of RDF and OWL, some ontology tools have been published to support documents publishing (see Section 2.3.3).

However, no single repository holding domain-specific knowledge can share knowledge until consensus of ontology derived from knowledge is achieved. To solve this problem, a first solution is to create ontology translation systems that translate one ontology to another. Difficulties related to this approach were articulated in Chapter 1. We could conclude that translation of ontologies between any two languages and/or tools cannot guarantee preservation of semantics and pragmatics. Also we mentioned that only ontology languages and tools will not be capable to extend semantic search services from one website to another.

2.4.2 Categories of Search Engines

In this section, we will trace the development of search engines to identify the central problems for the next generation search engines. Current popular search engines are mainly divided into three categories: *horizontal*, *vertical* and *blended* search engines. In addition, there is an emerging forth category called *self-help search engine*.

2.4.2.1 Horizontal Search Engine

Horizontal Search Engine is a typical search engine such as Google and Yahoo. People generally need to wade through pages of superfluous material to narrow their search to obtain their desired results. Horizontal search is featured with keyword-based indexing and minimal natural language processing.

2.4.2.2 Vertical Search Engine

Vertical Search Engine, also called "vearch", indexes content specialized by location (local venues and activities), by topic, typically for consumers, or by industry. Instead of returning thousands of links from a query, as is common with horizontal search, vertical search engines deliver more focused results to the user. However, this focus point is pre-defined for the repository and cannot be set by the user.

2.4.2.3 The Blended Search Engine

Blended Search Engine, such as the Google's "Universal Search" system, combines web search results with listings that come from vertical sources, such as news, video, images, and local information. Essentially, both blended and vertical search engines recategorize, reorganize, and/or refine content of websites by grouping documents by some attributes. However, they still work with fixed and predefined ontologies.

2.4.2.4 Self-help Search Engine

In a *self-helped search engine*, users describe documents posted by some attributes and submit them to a repository, and the search provider in turn will host it and make it searchable online. Unlike the other three types of search engines, the documents are uploaded by their owner instead of managing them on their own websites. Currently, the self-helped search engines do not support mechanisms to help the content providers construct the ontology and do not offer mechanisms for blending multiple repositories.

To summarize, with the contents of websites changing from pure textual data to annotated data, search engines have evolved from traditional search engine to more focused search engines that combine both collective users preferences and user provided contents. However, the data is annotated with proprietary ontology of the search provider,

which is predefined and kept fixed during the search process. So far, we have not been able to find search engines that provide mechanisms to seamlessly operate on multiple repositories with multiple ontologies.

2.4.3 Next Generation Search Engine

Currently, there are many suggestions from multiple perspectives for the next generation search engines, and their related products such as development platform and search protocols. Devising approaches to strengthen the "connectivity" between data items instead of documents and pages is a first choice. We believe that this kind of "connectivity" would be considered web-wide semantic interoperability and data ontology related items.

TechCrunch reports on an experimental search mechanism that is distinguished from previous versions by letting users to vote for or against search results [TechCrunch]. That means that search results are subjective, and there is no any standard result under certain query to be achieved. For example, for a single word "Prometheus", the results could be related either to the Greek mythology or to agent base software development methodology. "Satisfactory results" are depending on who will interpret them. Through voting mechanism, more commonly accepted results by most users can be obtained. However, voted results generally are compromised results, where they still contain, at least, minor parts, which cannot be commonly acceptable for all searchers. In addition, manual votes against search results are ineffective and are unable to catch up the pace of generating data. Finally, differences between voters would cause the results to dramatically deviate.

Yet, according to Oram, the bigger hurdle for centralized search providers is that trillions of bytes of dynamically generated data is being created by individual websites around the world (what some researchers call the *deep web*) and with the current pace no search provider can provide an up-to-date service to its users [Oram, 2008]. He also suggested a radically different architecture from any of the current popular engines, which is a peer-to-peer (P2P) solution to the search problem, named *Metasearch* engine [Oram, 2008]. Metasearch engine sends queries to multiple search engines and other data sources, then collates the results in some way and formats them for display; and the data sources can be internal indexes, associated text search engines, database search engines, message archives, etc. In this way, each website will play the roles of both information issuer and search service provider. This is not a new solution, and as [Oram, 2008] described, it at least appeared as far back as early 2000. Why Metasearch engines cannot be more widespread? What is holding Metasearch back? The answer to these questions is that "the lack of standards for categorizing data and knowing what to search for." [Oram, 2008]

To summarize, we believe that the innovations mainly fall into two types: *content-oriented renovation*, and *architecture-oriented renovation*. The former will resurrect search engines through the creation of real semantic web, on which data items (not pages) can communicate with each other, and then an ideal seamless web consisting of websites will emerge. The latter emphasizes restructuring search process model by introducing distributed P2P networking. This solution is aimed at effectively searching dynamically generated data on websites that form a cluster and communicate via

predefined protocols. However, so far, practitioners of Metasearch found that only primitive protocols are not enough to help interpret queries.

2.4.4 Our Suggestion

Based on the above argument, we consider that the next generation search engines should be semantic search engines that carry the following characteristics.

1. A search engine, as a part of semantic web, resides on a semantic web, and independently provides the service of semantic search. We would like to describe the working mode with the phrase "I search my way for others".
2. Semantic search engines can be dynamically organized in a cooperative group, in terms of common interest area, through a set of standard interoperations, and they will recognize and support each other by some predefined protocols.
3. There must be intelligent component(s) on every such website to help understand each other's ontology that are used to annotate the contents of websites, consequently automatically amend their own ontologies. Generally, these ontologies evolving in a distributed environment are not identical, and cause deviations during communications among websites.
4. The semantic search engine should be capable of dynamically annotating contents. When individual ontology has been amended, the contents of the website, correspondingly, need to be reorganized or recategorized to ensure the disambiguated search results.

The intelligent components, mentioned above, are considered a kind of learning agent from AI perspective, and currently some efforts are presented in [Williams, 2004] and [Jim, 2000]. As parts of our team research goals (see Chapter 1), we have presented a

method for agents to learn concepts from several peer agents [Afshachi, 2006] [Afshachi, 2006A] and a method for verification of the learned concepts [Far, 2007]. Through our analysis, I believed that there is correlation between the learning module and semantic search module, and they compose a generative process on both sides during communication, so the study on both semantic search engine and learning agent should both be taken into account instead of studying them separately.

2.5 Introduction of Supporting Open Sources

To assist in the implementation of the prototype (see Chapters 1 and 4), after reviewing some popular open sources, we finally settled on UIMA. In this section we provide a brief overview of UIMA.

2.5.1 UIMA

The UIMA standing for Unstructured Information Management Architecture, is an open, industrial-strength, scalable and extensible platform for creating, discovering, composing and deploying a broad range of multi-modal analysis capabilities and integrating them with search technologies [Apache, 2006]. Although UIMA originated at IBM [IBM, 2007], it has now moved on to be an Open Source project which is currently incubating at the Apache Software Foundation [Apache, 2006]. UIMA, referred to as the *UIMA specification,* is undergoing a standardization effort at OASIS [1], via the OASIS Unstructured Information Management Architecture (UIMA) TC, working on standardizing semantic search and content analytics. Apache UIMA is consisting of three parts:

---

[1] *Organization for the Advancement of Structured Information Standards is a not-for-profit consortium that drives the development, convergence and adoption of open standards for the global information society.*

- All-Java implementation of the UIMA framework for the development, description, composition and deployment of UIMA components and applications.
- Eclipse-based (http://www.eclipse.org/) development environment that includes a set of tools and utilities for using UIMA.
- C++ version of the framework, and enablement for Annotators (see Figure 2-6 below) built in Perl, Python, and TCL.

After IBM transferred UIMA to Apache, the implementation part of the research project, accordingly, has moved on Apache UIMA.

2.5.1.1 Theoretical Foundation of UIMA

The development of UIMA is based on analytics, the science of analysis [Apache, 2006], which is extended into many other areas including data mining, machine learning, statistics, web analytics, OLAP, etc. Data mining is an essential part of analytics; however, analysis and extensive computation on the mined data make Analytics distinguished. Mature means of analytics and their combination for discovering and understanding interesting data objects, no doubt, can be utilized to dig semantics out of raw data of specific domain during semantic search process.

2.5.1.2 Basic Concepts of UIMA

With the background of analytics, some basic concepts of UIMA inevitably are analytics-oriented. In this section, the roles these important concepts are playing under UIMA and the ways they are involved in a workflow of annotation are explained. Figure 2-6 shows a general workflow in which almost all critical concepts of UIMA: Analysis Engine, Annotator, CAS, and Component Descriptor are involved. In the following paragraphs, interpretations of these concepts and the general workflow will be explained.

- **Analysis Engine (AE)**: Basic building blocks of UIMA, which are composed to analyze a document and infer and record descriptive attributes about the document as a whole, and/or about regions therein [UIMA, 2007]. AEs are treated by UIMA as pluggable, discoverable, and manageable objects. The core of AEs are the analysis algorithms that do all the work to analyze documents and record analysis results.

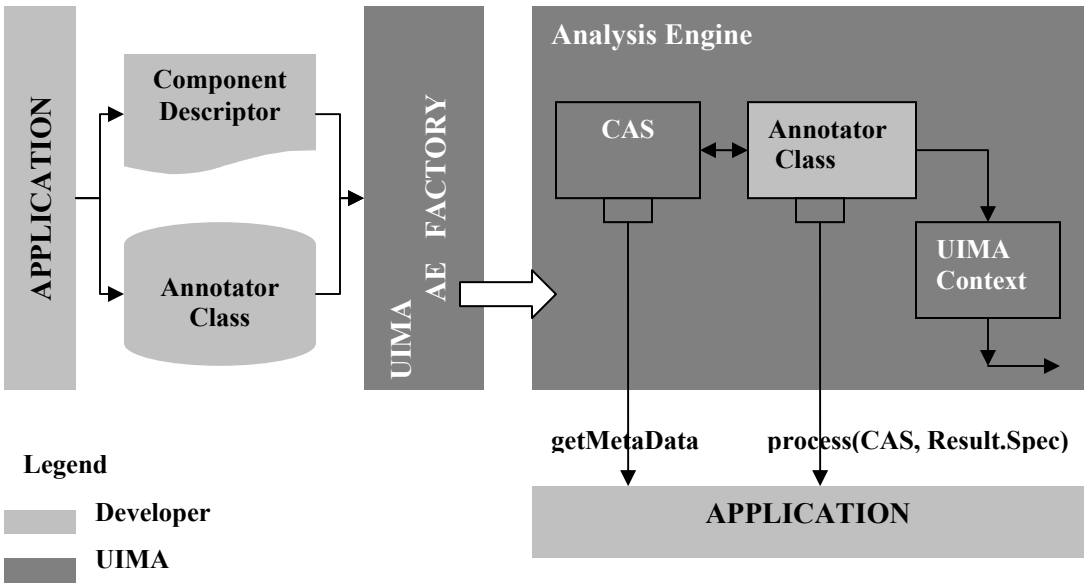


Figure 2-6: General Workflow of Annotation

- **Annotator:** Instance of basic component type intended to house the core analysis algorithms running inside AEs. UIMA framework provides the necessary methods for taking annotators and creating analysis engine at run time. This feature saves Annotator Developer's time expensed in creating complete software framework.
- **CAS**: UIMA defines a Common Analysis Structure (CAS) for Annotators to represent and share analysis results. The CAS contains
  - ✓ the artifact being analyzed (e.g. Document, audio file etc.),

- ✓ the analysis results, and
- ✓ the type system

- **Type and Type System:** Analysis results are descriptive information produced from AEs and include different statements about the content of a document. These statements, i.e. meta-data about the content of a document, are represented with special pre-defined terms characterizing the kinds of results that an AE may create. A type system defines several types that may be discovered in documents by an AE, can be seen as a schema for the analysis results.
- **Component Descriptor:** Components Descriptor is interface defined by UIMA for basic building blocks such as AE and Annotator for which users of UIMA framework should provide implementation. An complete Component Descriptor includes two parts: *declarative* part and *code* part.

  The declarative part is so-called Component Descriptor written in XML containing a series of standard metadata describing the components, its identity, structure and behavior. Actually, it is a specification to instruct UIMA how to compose analysis capabilities and ultimately applications. Along with it, developers should provide the code part to implement the analysis algorithm to make component become a real pluggable component for UIMA.

2.5.1.3 Work Flow of UIMA

Figure 2-6 depicts overview of an analysis process of UIMA. The application is responsible for interacting with the UIMA framework to instantiate Analysis Engine, create or acquire an input CAS, initialize the input CAS with a document and then pass it to the Analysis Engine through the process method. The UIMA AE Factory reads the

declarative information from the Component Descriptor and the class files implementing the annotator, and instantiates the AE instance, setting up the CAS and the UIMA Context (another type of interface for accessing external resources). The AE, possibly calling many other AEs internally, performs the overall analysis and its process method returns the CAS containing new analysis results. The application then decides what to do with the returned CAS.

2.5.1.4 Summary of UIMA

With practices of UIMA when implementing research prototype system, we noticed the following characteristics of UIMA that make it distinguished as a development platform:

- **Flexibility:** The framework itself is not specific to any IDE or platform. UIMA supports the development and integration of analysis algorithms developed in different programming languages. The Apache UIMA project is both a Java framework and a matching C++ enablement layer, which allows annotators to be written in C++ and have access to a C++ version of the CAS. The C++ enablement layer, in turn, enables annotators to be written in Perl, Python, and TCL, and to interoperate with those written in other languages.
- **Well-Supported:** UIMA provides APIs and tools for creating primitive analysis components including tokenizers, summarizers, categorizers, parsers, named-entity detectors, etc.
- **Theory-solid:** UIMA is developed based on Analytics.
- **Big-Community:** There is a big community contributing to the project. It is primary consideration to choose UIMA.
- **Future Benefits:** There are two features that possibly benefit future works:

1. UML model automated import.
2. Support of SOAP protocols.

Chapter Three: Concept-Learning Supported Semantic Search using Multi-agent System: Our Approach

This chapter presents the approach of MAS-supported semantic search [Zhong, 2008], based on reviewing the current progress of semantic search. This MAS system is modeled under guidelines of GAIA MAS design methodology, and is characterized by five-layer architecture, *encoding, lexical, syntactical, semantic, and semiotic,* under which procedures of a semantic query are regulated. Firstly, we begin with addressing some fundamentals and give a precise definition of the problem. Secondly, we propose the design solution under guidelines of GAIA [Wooldridge, 2000] methodology. Thirdly, we explain the work hypotheses, the assumptions considered as a starting point for this work.

3.1 Introduction

"In contrast with the traditional information retrieval technology, which purely depends on the occurrence of words in documents, semantic search denotes one or more concepts in the context of other concepts. Understanding the denotation of concepts can help retrieval part of search engine understand the context of search, the activity the users is trying to perform, thus drive expectations on the categories of documents." [Guha, 2008].

The essence of semantic search is semantic interoperation towards denotation part in the search phrase. Nowadays, general denotation procedures are realized depending on ontology-oriented means, and ontologies adopted are usually evolved and maintained in distributed ways. Thus, multiplicity of ontologies raises the issue of integration and thus directly renders the communication between peers involved in a semantic search ineffective.

The establishment of a common ontology of a certain domain is one of cornerstone among cooperative agents (peers) participating in semantic search. However, coming up with a common ontology may not be realistic in all cases. In multi-agent systems (MAS) research concerning agents' communication, having a common ontology is only possible when the design rationale, the concepts and meanings assigned to the concepts as well as the context of applying the concepts are shared. In other words, the agents should be designed by one group of developers and all of the concepts related to specific domain and their meaning (i.e. semantics) should be provided in advance. In heterogeneous MAS, many researchers assume that it is possible to have a common ontology for the agents and that the agent developers naturally will integrate this common ontology into their agents, thus allowing for easy communication (and understanding) among agents. However, the assumption of a common ontology is often too strong or unrealistic. In many cases, for a single domain, there is no agreement on the same ontology among developers, and for several domains, the potential ontologies are large, unwieldy and may encompass more than what is needed by one of particular MAS, and implementing complex ontologies can also easily lead to discrepancies between implementations.

Recently, the idea of having agents *learn* concepts (or languages) from each other has been suggested as a solution to improve agent communication. For example, the work in [Williams, 2004] suggests a method for learning a language and the work in [Steels, 1998] has focused on interactions between two agents to learn a single concept. In our previous work, we have presented a method for agents to learn concepts from several peer agents [Maier, 2004] and a method for verification of the learnt concepts [Kant,

1965]. In this chapter, we propose a layered architecture to guide analysis and realization of multi-agent semantic search based on the levels of semantic interoperability proposed in [Euzenat, 2001]. In addition, we present a method that makes use of the learnt ontologies in a multi-layer semantic search.

3.2 Semantic Interoperability and Semantic Levels

[Euzenat, 2001] defines semantic interoperability as the faculty of interpreting the annotations at the semantic level, i.e. to ascribe each imported piece of knowledge to the correct interpretation or set of models, and possible levels of interoperability need to be considered when trying to understand an expression from other systems are also presented. These levels in ascending order of semantic intensity are: *encoding*, *lexical, syntactic, semantic, and semiotic* (functionalities of each layer are explained below).

3.3 Conceptual Model of Layered Architecture of Semantic Search

The idea of layered semantic interoperability is directly applied in development of architecture of layered semantic search. Surrounding every level of the interoperability, the corresponding layer in the conceptual model is proposed and the model is illustrated in Figure 3-1. Definitions of functionalities of the layers are described below.

- The **encoding layer**, as base layer, should provide "ability to segment representation in characters" [Euzenat, 2001], i.e. defines encoding format of data exchange, and thus implicitly defines the character sets of a natural language for exchanging a search phrase. ASCII and Unicode are mainly used as encoding formats.

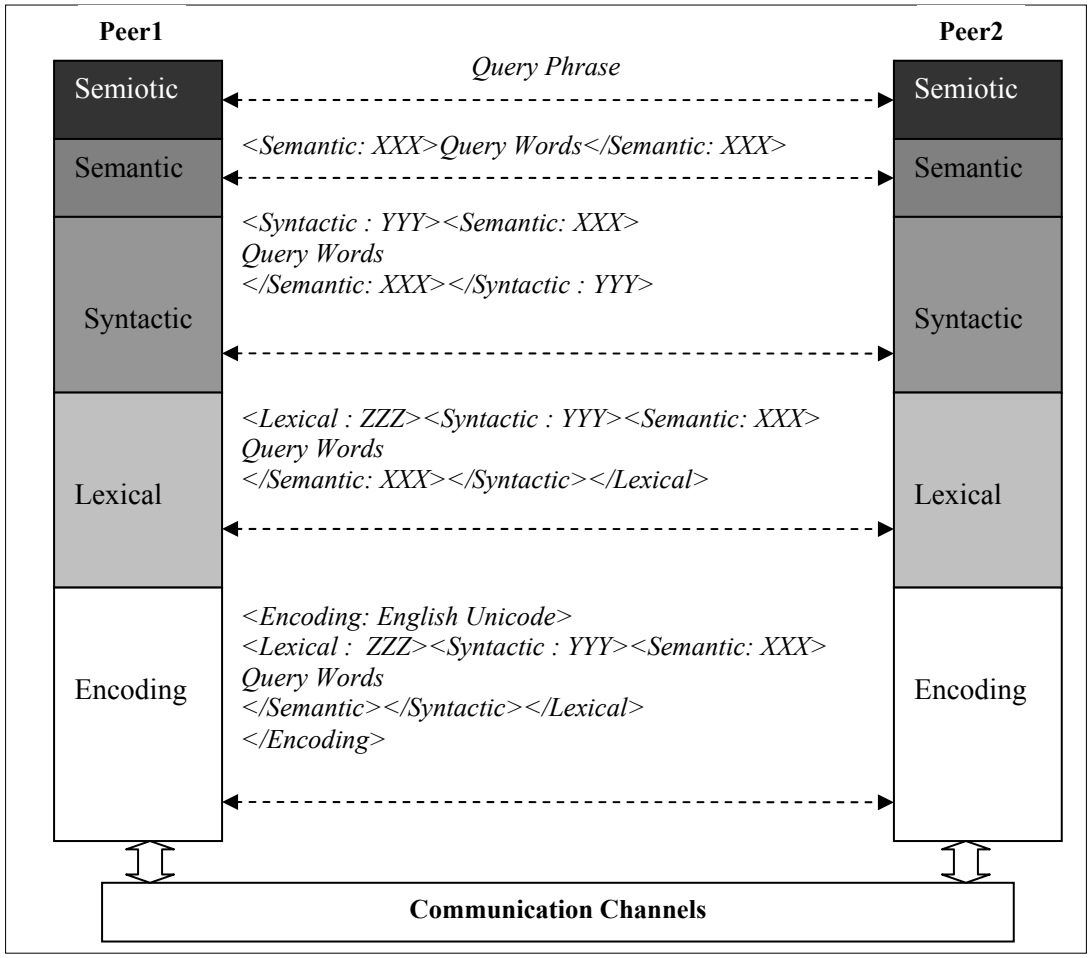


Figure 3-1: Layered Architecture of Semantic Search

- The **lexical layer** should provide "ability to segment a representation to words (or symbols)" [Euzenat, 2001], i.e. tokenizes the search phrase. At this layer, important identifiers of ontology components are identified. Functionality of this layer is not easily realizable for some natural languages such as Chinese because tokenization of sentences is a big issue due to the lack of explicit word delimiter, except for punctuation, to separate each single word (or symbol). This is basic functional layer.
- The **syntactical layer** is a more complex layer which is needed to provide "ability to structure the representation in structured sentences (or formulas or assertions)" [Euzenat, 2001], i.e. identifies concepts by structuring words following grammar at

query side, and it is capable of understanding the structured representation to extract concepts at responding side.

- The **semantic layer** provides ability to "understand propositional meaning of the representation of search phrase" [Euzenat, 2001].
- The **semiotic layer** provides ability to "understand meaning of the representation of search phrase in a context (specific domain)" [Euzenat, 2001].

There are some necessary constraints imposed on the model to regulate communications between peers, namely:

1. One layer only talks with its peer layer on remote side complying with some protocols. These protocols help both sides to settle natural languages, encoding standards for exchanging information, representation grammar of search phrase, etc. If two peers do not find any layer at which they can communicate, then the most primitive style of search, i.e. keyword-based search, will be conducted.
2. A search phrase can be optionally initiated at any layer, then will be passed down, layer by layer, to the bottom, encoding layer. Each layer will add corresponding annotation information to the search phrase. Packaged phrase, finally, will be sent out.
3. Each layer take charge of the operations only at the same level of the semantic spectrum [Daconta, 2003] of ontology (refer to the section 2.2.2).

3.4 Why Adoption of Layered Architecture?

First of all, I believed that the essence of semantic interoperation, within the environment targeted by this thesis, is aimed at calibrate the ontology at all semantic levels, which are depicted with the spectrum of ontology [Daconta, 2003]. This spectrum presents the

modalities of ontology according to the semantic intensity. Based on the calibrated, (so-called) common ontology, communication between peers will be achieved. This model resembles humans' natural communication style that each semantic level can be achieved only if the lower ones have been traversed. For example, people can exchange useful information only if they have chosen a language and clarified meanings of concepts which are relevant the topic.

Moreover, a layered architecture normally is able to reduce complexity by breaking complex semantic interoperability into smaller problems; standardized interfaces between adjacent layers facilitate modular engineering and development of search tools; and also accelerate evolution of technology.

Finally, a successful sample really is an inspiration to apply this type of layered architecture that is WWW as [Decker, 2000] pointed out: "The World Wide Web is possible because a set of widely established standards guarantees interoperability at various levels."

3.5 Lexical Layer - Target Layer of the Thesis

Before explaining why this thesis concentrates on the lexical layer rather than other layers of semantic interoperability, two main approaches to modeling semantics should be iterated: declarative and procedural semantics. The definitions of them [Decker, 2000] are listed below,

- Declarative approach gives an expression E the meaning by mapping it to another well-understood formalism. The expression can be understood without reference to any specific computational procedure.

- Procedural approach give expression E meaning by referring to the behaviour that some real or virtual procedure will exhibit on E. Often the only way to obtain the expression's meaning using procedural semantics is to simply execute the procedure and observe the outcome.

According to the definitions of functionalities of layers of semantic interoperability, in order to achieve the interoperations between peers, modeling semantics of concepts and use them on the layers of syntactic, semantic, and semiotic need external "schema" in procedural approaches, meanwhile, for the lexical layer, the declarative approach could solely accomplish modeling by referring concepts to some well-commonly-understood formalism.

Considering the fact explained above that the research team is facing, at initial phase of the research project, the concept learning module [Afsharchi, 2007] is built with a kind of declarative concept learning algorithm, i.e. concentrating on lexical layer, therefore, as one of two main interactive entities involved in the spiral-like route, semantic search engine accordingly will be aimed at the lexical layer, also.

Besides, from the perspective of cognitive principle, lexical layer would be a basic layer of communication which ultimately leads to understandings of complicated semantics, so that successfully achieving semantic interoperation would lay a solid foundation for other layers.

Being aware of the relationships of layers of semantic interoperability and modeling approaches, and the relationships between layers of semantic interoperability and semantics spectrum (refer to Section 2.2.2) of ontology, is critical to delimit the prototype system, as the semantic interoperation for certain layer strongly relies on the

support of ontology modeled at the corresponding level. Based on the explanation of the previous paragraphs of this section, the objective of the thesis is decided to design and implement a prototype of semantic search system focusing on the lexical layer as indicated in Figure 3-2. Implementation details are provided in Chapter 4.

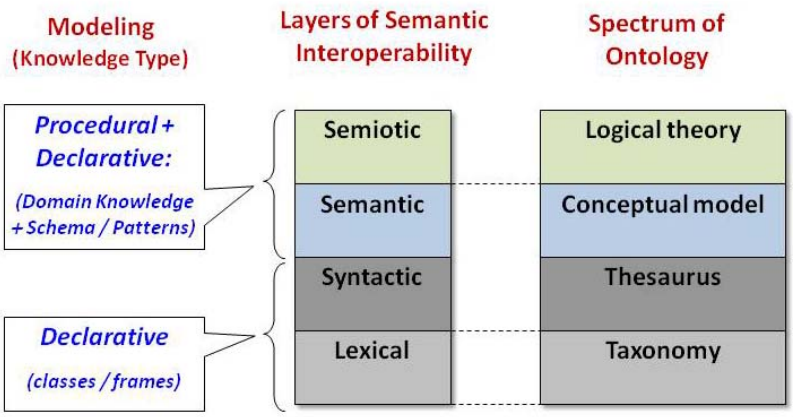


Figure 3-2: Scope of the Prototype System about Relationships between Semantics Spectrum and Semantic Interoperability of Ontology

3.6 Introduction of GAIA

GAIA methodology, as an increasingly detailed model elaboration process that emphasize on social-level abstractions [Wooldridge et al. 2000], is selected to guide our analysis activities including requirement analysis, system analysis and system design including acquisitions of agent model, service model, and acquaintance model. GAIA is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly, and it is appropriate for these real-world applications with following main characteristics which our research project possesses [Wooldridge, 2000]:

- Agents are heterogeneous, in that different agents may be implemented using different programming languages, architectures, and techniques. No assumption about delivery platform needed to make.
- The organization structure of the system is static, in that inter-agent relationships do not change at run-time.
- The abilities of agents and the services they provide are static, in that they do not change at run-time.

3.7 System Requirement Analysis

Our system focuses on realizing lexical layer of semantic interoperability (i.e. taxonomy range of semantic spectrum of ontology). We assume our system only deal with English words at lexical layer of semantic interoperability. The overview of prototype system for annotation-learning workflow within lexical layer is shown in Figure 3-3. In this system,

1. Software agents are able to automatically form a cooperative group through social activities and independently provide search services without being mediated by any centralized search engines such as Google or Yahoo. Concretely speaking, they should be able to register in the group, and access each other directly to obtain information necessary to initialize a conversation.
2. Agents are responsible for organizing documents in their own data repository using any ontology they deem appropriate, including

- annotating documents dynamically or statically with concepts and/or keywords.
- re-categorizing data repository in accordance with taxonomy level of private ontology.

3. Agents collaborate with each other to accomplish semantic search and concept learning, and the search complexity is hidden from human users. Concretely,

- agents should be able to learn new concept through their private algorithms to enrich their ontology, i.e. enrich the concepts on taxonomy level;
- agents should be able to propagate newly-learnt concept within the cooperative group using special agreements (or protocols).
- agents should be able to maintain concepts.

As first essential step of analysis stage, we should identify all the roles in a system, usually; we apply such guidelines to help finding the roles:

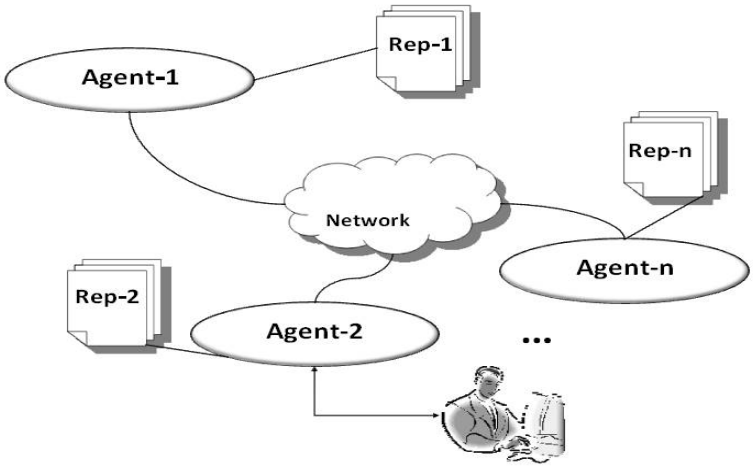


Figure 3-3: Overview of Prototype System

- Identify the stakeholders of the system, i.e., those who will interact with the system, and those inside the system that will take care of the interactions.
- Identify the roles need to perform the main functionalities or services of the system. These roles are usually acquired by analyzing the system requirements.
- Identify whether there are roles as facilitator or supporter for other roles.

This section, based on the requirements listed above and following the GAIA Model structure, we acquire an agent's role model for the roles: ***Query Handler***, ***Peer Finder***, ***Document Annotator***, ***Concept Learner***, ***Register Handler***, ***Concept Manager, and USER,*** and interaction model in the analysis stage. Then, describes the agent model, service model, and acquaintance model generated in the design stage later.

3.7.1 The Role Model of the Prototype System

As GAIA suggested, the objective of the analysis stage is to develop an understanding of the system and its structure (without reference to any implementation detail).

Role Schema: DocumentAnnotator (DA)

Descriptions:

Responsible for filter out satisfactory documents by dynamically annotating all documents with received query phrase.

Protocols and Activities:

DoAnnotation, ReturnDocuments.

Permissions:

reads	**supplied**	*queryPhrase*	*// query phrase.*
generates		*annotations of documents*	*// annotations for all documents*
		documents	*// satisfactory documents*

Responsibilities:

Liveness:

DocumentAnnotator = (DoAnnotation . ReturnDocuments) $^{\omega}$

Safety:

- queryPhraseIsLegal = false => documents = nil

Figure 3-4: Schema for Role Document Annotator

From GAIA perspective, an application system can be viewed as an organization, which is a collection of roles. GAIA analysis is an increasingly detailed model elaboration process that emphasize on social-level abstractions [Wooldridge, 2000]. In this section,

we will follow the GAIA Model structure to acquire the role model and represent them through the GAIA approach of Role Schema.

3.7.1.1 Role Schema

Due to the page limit, we only listed role schemas for two main roles, and the explanation of role schemas of rest role models can be found in Appendix A.

The role schema of Document Annotator is presented in Figure 3-4.

The role schema of Concept Learner is presented in Figure 3-5.

Role Schema: ConceptLearner (CL)

Descriptions:

Responsible for initiate a query which will result in a number of documents returned, and then based on these sample documents, special learning algorithm(s) will be applied to produce a new concept.

Protocols and Activities:

RetrieveConcept, QueryInit, ConceptLearn, RequestConceptIntegrate

Permissions:

sends *queryPhrase* // *formal query phrase written in certain syntax*
generates *newConcept* // *newly cognized concept*

Responsibilities:

Liveness:

ConceptLearner = [RetrieveConcept] • (QueryInit, ConceptLearn, RequestConceptIntegrate) +

Safety:

- queryPhraseIsLegal = false => documents = nil

Figure 3-5: Schema of Role Concept Learner

3.7.2 The Interaction Model of the Prototype System

As GAIA suggested, the links between roles need to be represented in the interaction model consisting of a set of protocol definitions, one for each type of inter-role interaction [Wooldridge, 2000].

3.7.3 The Definition of Protocols Associated with Roles

The protocols associated with QueryHandler Role are listed in Figure 3-6.

The protocols associated with ConceptLearner Role are listed in Figure 3-7.

The protocols associated with QueryHandler Role are listed in Figure 3-8.

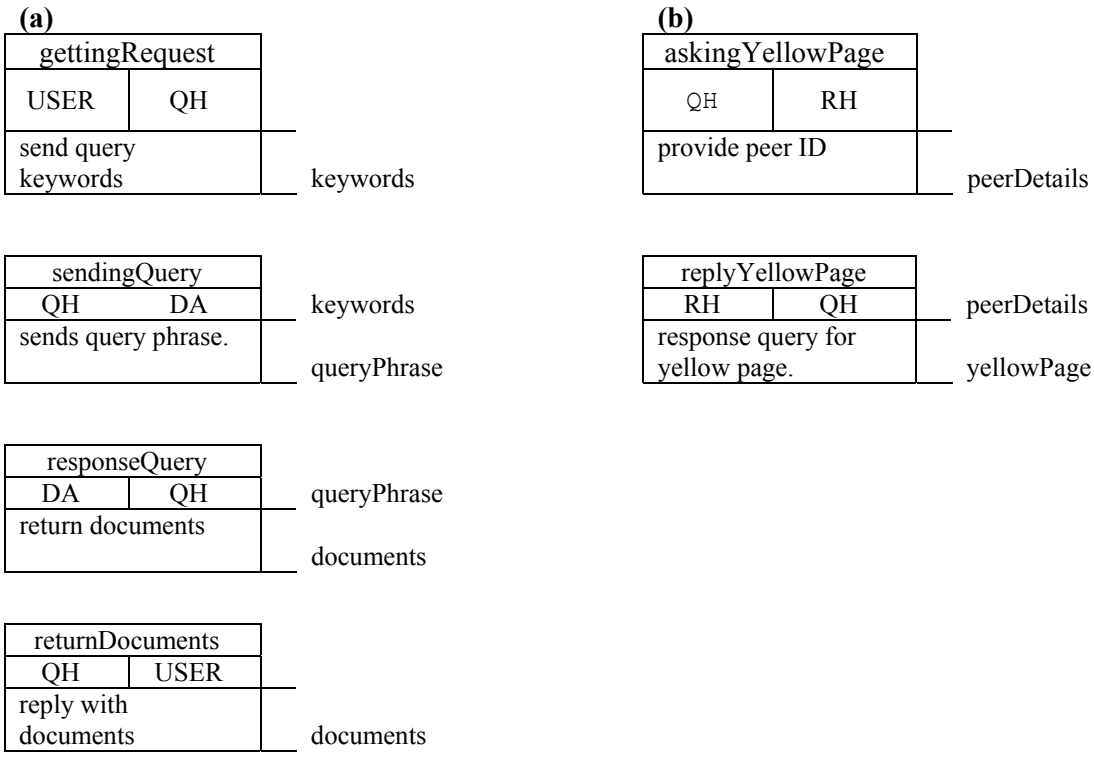


Figure 3-6: Definition of Protocols Associated with the QueryHandler Role: (a) QueryDocuments, and (b) RequireYellowPage

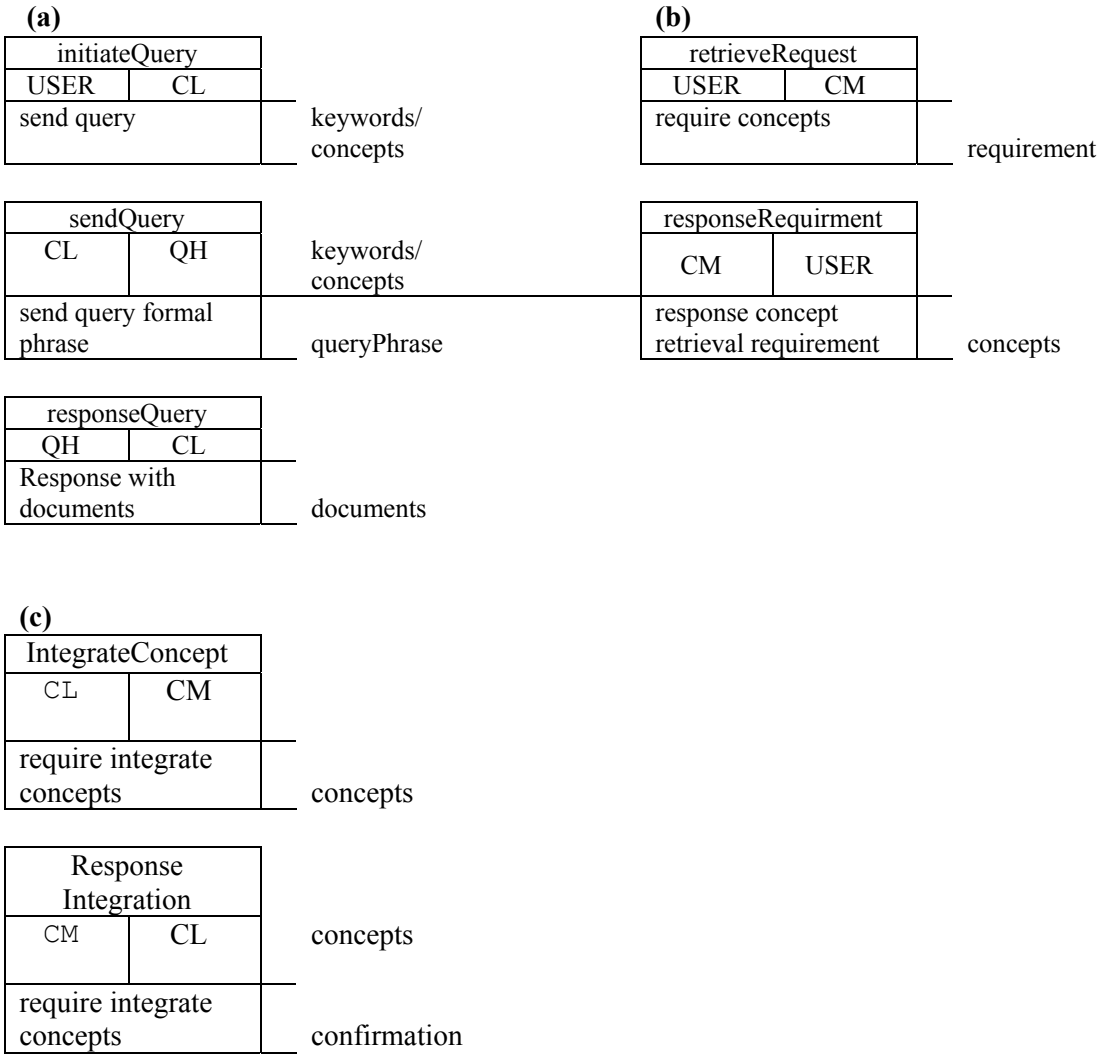


Figure 3-7: Definition of Protocols Associated with the ConceptLearner role: (a) InitiateQuery, (b) RetrieveConcept, and (c) IntegrateNewConcept

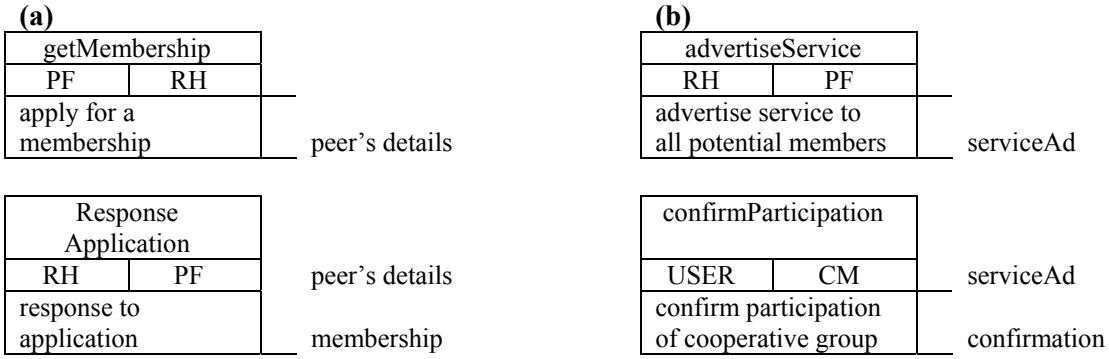


Figure 3-8: Definition of Protocols Associated with the RegisterHandler Role: (a) ApplyForMembership, (b) AdvertiseService

The GAIA design process involves generating three models: agent model, services model, and, finally, the acquaintance model.

3.7.4 The Agent Model of the Prototype System

The agent model is shown in Figure 3-9.

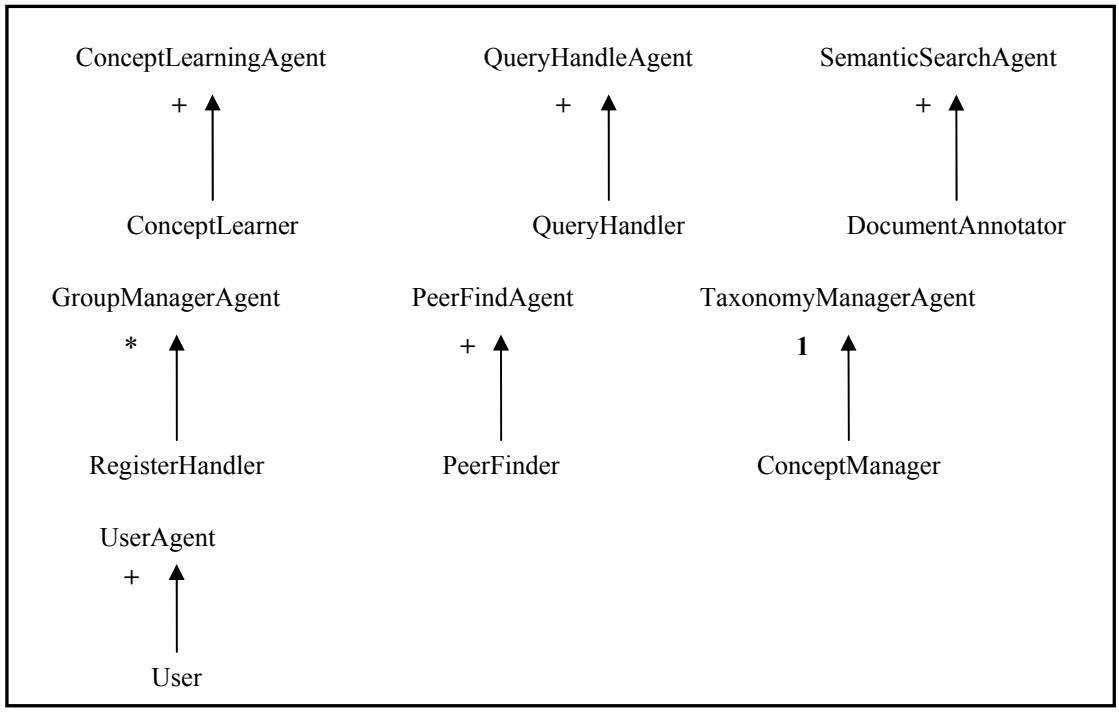


Figure 3-9: The Agent Model.

3.7.5 The Service Model of the Prototype System

The aim of the GAIA services model is to identify the services associated with each agent role, and to specify the main properties of these services. A service is simply a single, coherent block of activity in which an agent will engage. It would be clear there every activity identified at the analysis stage will correspond to a service, though not every service will correspond to an activity. The service model is represented in Figure 3-10.

Service	Inputs	Outputs	Pre-condition	Post-condition
encode query phrase	keywords + concepts	query phrase	keywords ≠ nil	query phrase ≠ nil
Annotate documents	row documents	annotated documents	documents satisfying search condition	**true**
concept learn	positive and negative sample documents	new concept	documents ≠ nil	new concept ≠ nil
create taxonomy	predefined terminology and simple relationships	taxonomy of specific domain	terminology ≠ nil	**true**

Figure 3-10: The Services Model

3.7.6 The Acquaintance Model of the Prototype System

The acquaintance model of prototype system is shown in Figure 3-11.

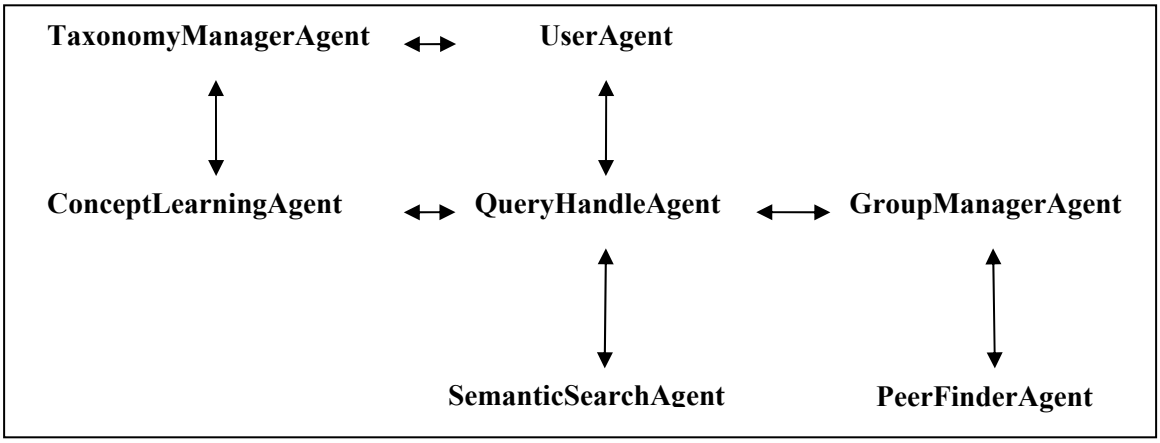


Figure 3-11: The Acquaintance Model of Prototype System

3.8 Implementation Design

The annotation process is depicted in Figure 3-12. The roles played in the annotation process are:

3.8.1 CreateConceptHierarchy([concept], keyword1, keyword2, …, CH1)

Annotator first will create a Concept Hierarchy (CH) using received series of keywords. This CH directly goes into the Annotation Engine (AE) to tell what is needed to be searched from the document repository.

3.8.2 CreateAnnotationEngine (Type1, AE1)

This method takes CH as a parameter to dynamically build Annotation Engine (AE) which is supposed to be the algorithm's container.

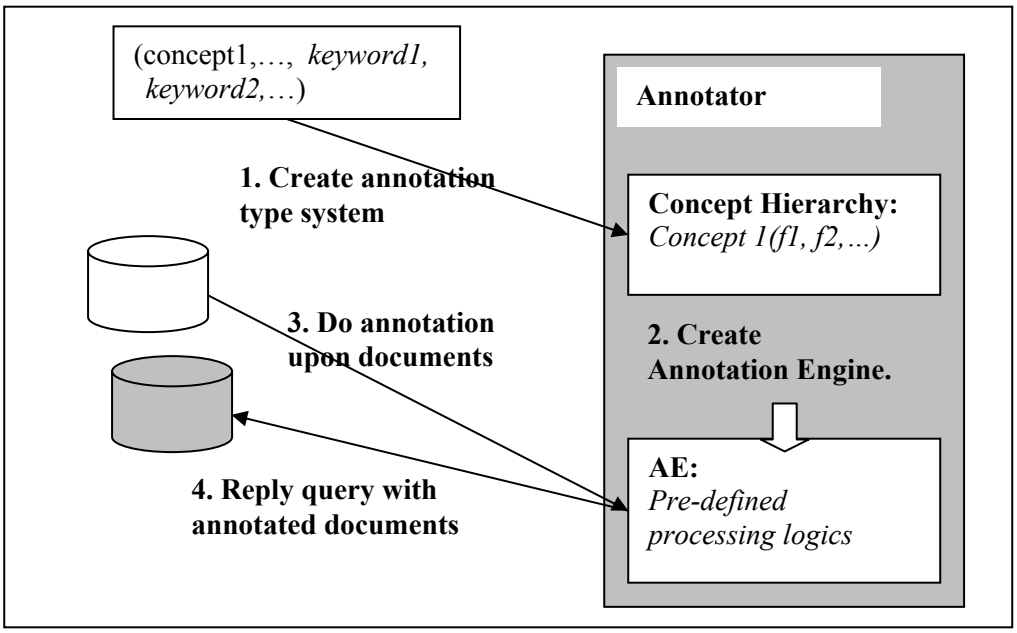


Figure 3-12: Process of Documents Annotation

3.8.3 DoAnnotation(Annotator1, Doc1, Doc2, …)

Once annotation engine is created, it will be run against repository to annotate and grab satisfying documents.

3.8.4 ReplyQuery(Annotated Documents, PositiveExamples, NegtiveExamples)

This method takes charge of replying to query. In this project, it also implements some specific filtering work as concept learning required.

Chapter Four: Implementation

The main goal of implementation of the prototype, Concept-Learning Supported Semantic Search MAS, is to investigate the feasibility of the spiral-like workflow (explained in chapter one) between semantic search and concept learning at lexical level of semantic interoperability. Examine how both dynamical annotation and new concept identifying procedure exert influences on the search results. Once this system have been mature and solid, as long term goal, a general KM experimental framework would be expected to be incubated from it, on which a variety of algorithms concerned with KM can be flexibly plugged in to be testified (This goal is not covered in this thesis).

In this chapter, the major part of implementation design described with UML language will be given. Following the design section, core implementation works are provided and a study case is taken to demonstrate the basic spiral-like circle.

4.1 Design of Implementation

Following the specification of GAIA methodology, we obtained the scheme of the MAS prototype. Base on it, especially on agents' social role module, we will carry on analysis and design for implementation. In this section, following UML syntax, use case diagram and class diagram of analysis and design will be given to depict the prototype system in high level. Besides, we will provide responsibilities for every main class.

4.1.1 Use Case Diagram

The prototype is consisting of the roles shown in Figure 4-1 drawn with IBM Rational Rose. They are Document Annotator, Peer Finder, Concept Learner, Register Handler, Query Handler, and Concept Manager.

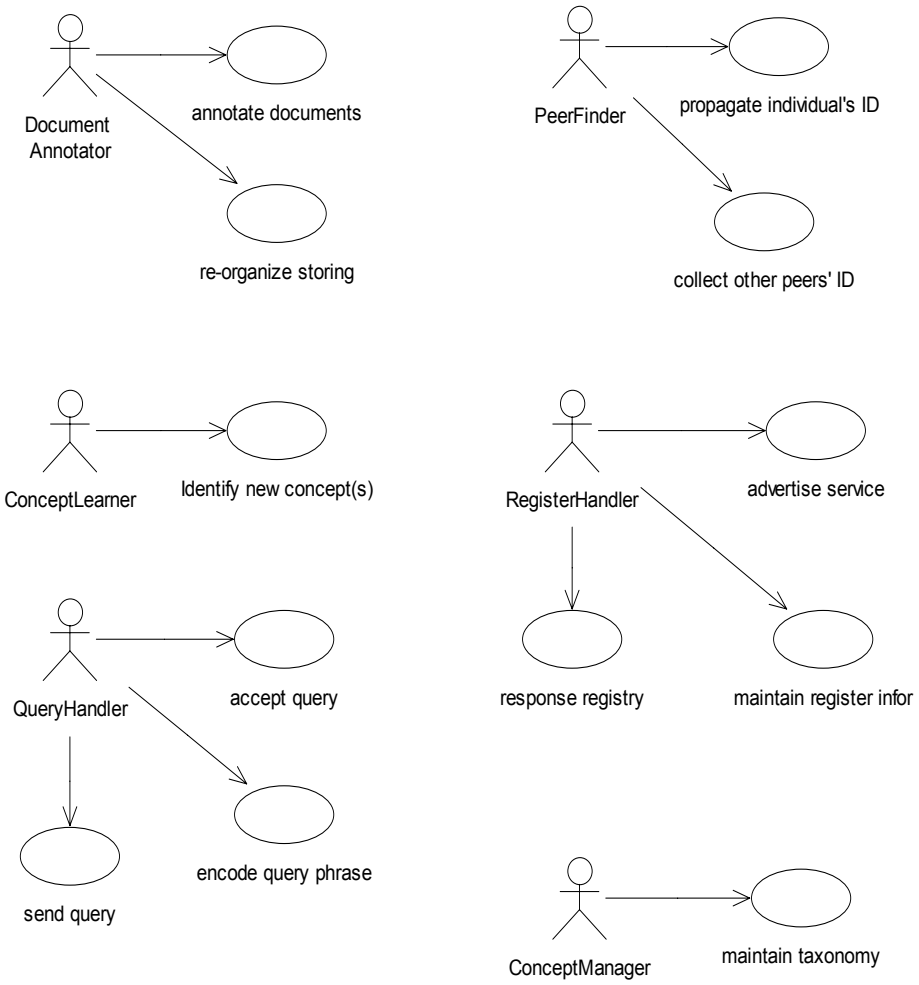


Figure 4-1: Use Case Diagram of the Prototype

4.1.2 Class Diagram of Analysing and Design

The analysis class diagram as shown in Figure 4-2 sketches the abstract class model. The definitions of them are explained in the following section, along with implementation of each role of the prototype.

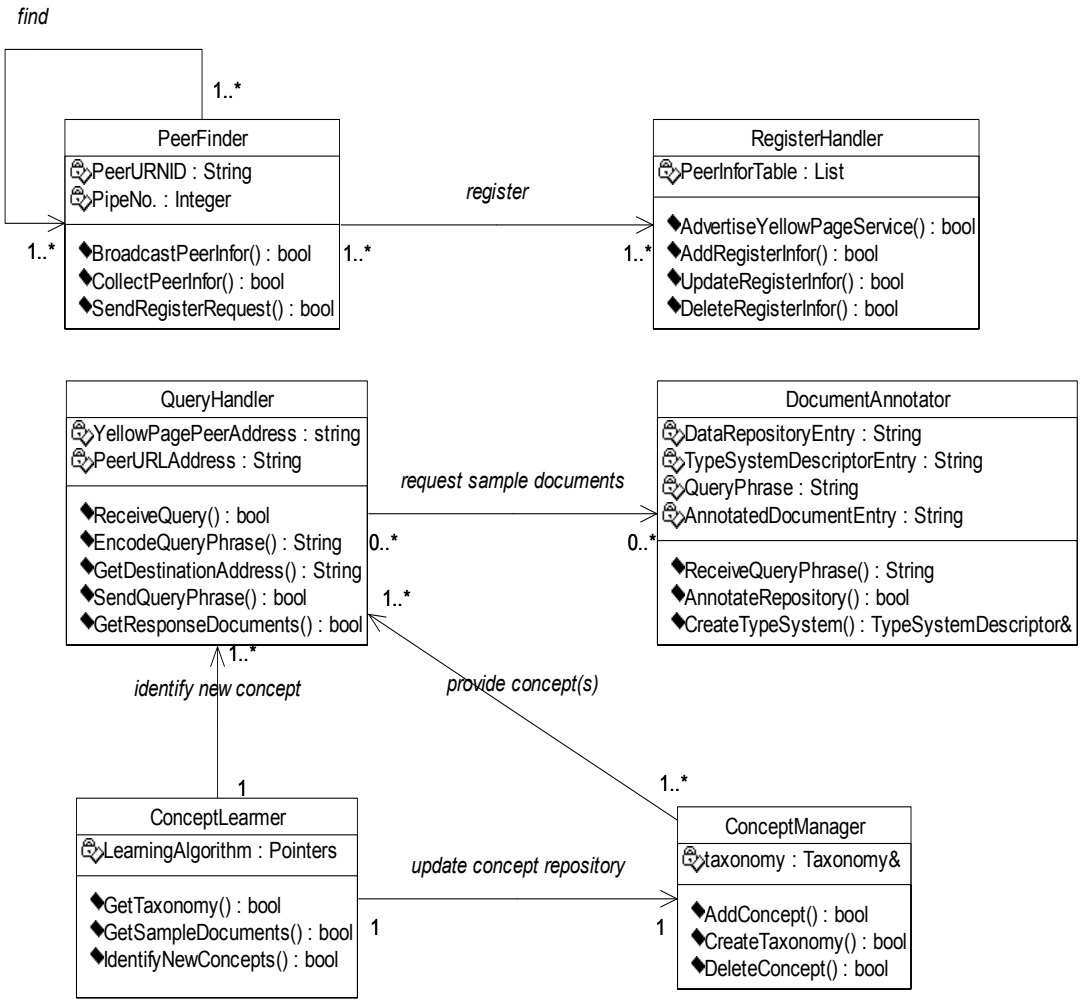


Figure 4-2: Class Diagram of Analysis and Design

4.1.3 Definitions of Classes

In this section, we would like to describe responsibilities by describing workflows involving classes by taking use of UML sequence diagrams.

4.1.3.1 Workflow of Peer Group Establishment

A peer group can be created in two ways as illustrated in Figure 4-3 : (1). Register Handler initiatively broadcast yellow page service, and then listens to the replies from

each peer. Information of each peer concerned with identification will then be registered. Peers need reference Register Handler before trying to connect with each other; (2). Without Register Handler, peers broadcast their own information and collect responses from others try to find some potential cooperative partners, and negotiate one to one to gradually form a group.

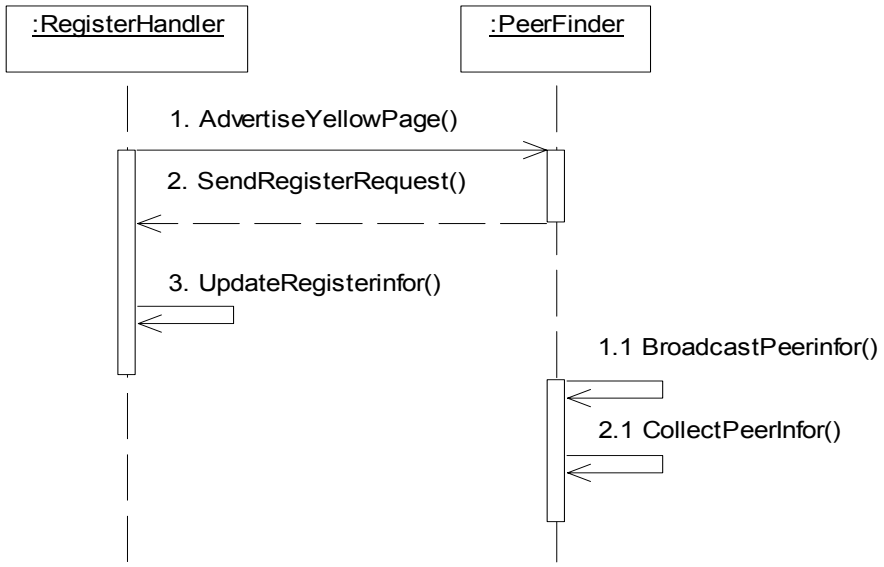


Figure 4-3: Sequence Diagram for Peer Group Establishment

4.1.3.2 Workflow of Query Operation

Surrounding the Document Annotator, the workflow reflecting query process is presented in Figure 4-4 that covers rest of classes described in Figure 4-2. Document Annotator mainly responds dynamically create concept hierarchy and annotation engine, execute dynamical annotation process, and reply the query. Concept Manager Role should be able to support agent to manage the set of local concepts and taxonomies composed of them. It directly serves Concept Learner. The sources that concepts come from include,

1. Selected by experts of specific domain, which should be unique to the domain,
2. Newly learnt through training course,
3. Newly learnt through semantic search.

Concept Manager should support two basic actions:

- *SaveConcept:* Save concepts and/or taxonomy composed of them in the repository. It is called by Concept Learner.
- *RetrieveConcept:* Retrieve concepts or taxonomy composed of them by corresponding ID, which are encoded into query phrase to complete training or semantic search processes.

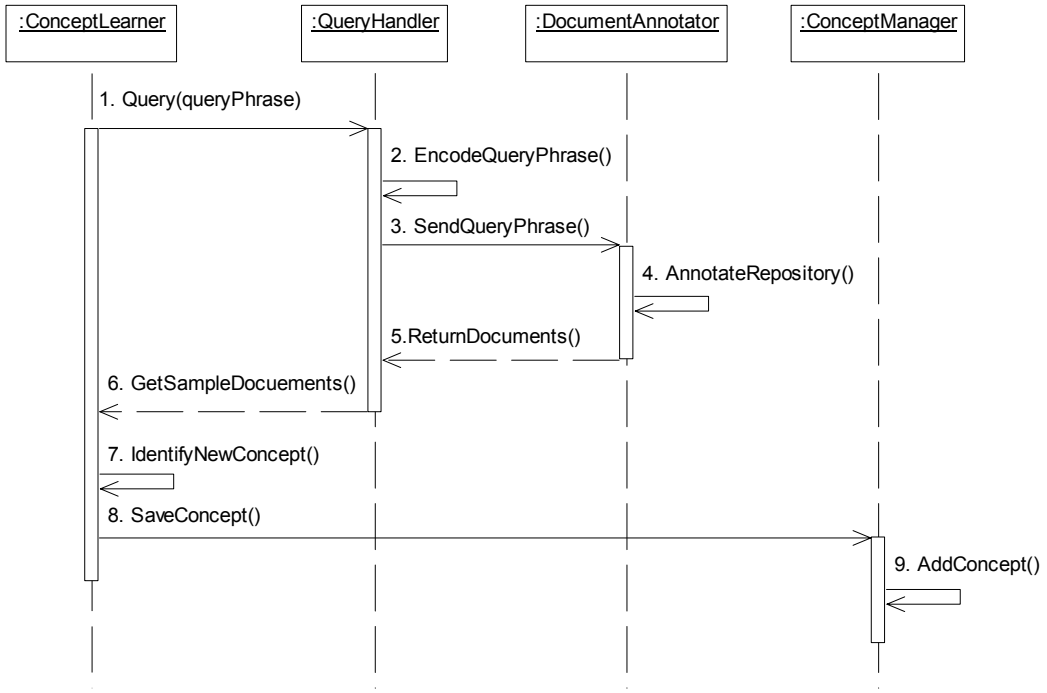


Figure 4-4: Sequence Diagram for Query Workflow

4.1.3.3 Concept Learner

The *Concept Learner* role is responsible for implementing action *Learn* which takes training documents as input, and produces concept classifier. Also it offers methods to do action *Integrate*. The implementation is not covered by this thesis, and the details of implementation are presented in [Yang, 2008].

4.2 Implementation of Prototype

In this section, we review two options of network architecture adopted for the prototype: hybrid P2P networking and pure P2P networking, and based on those network architectures, we present the implementation details about Document Annotator which is working in the hybrid P2P networking.

4.2.1 Network Architecture of the Prototype

Recently, Peer-to-Peer (P2P) computing is receiving ever increasing attention of both academia and industry, and especially, in multi-agent system (MAS) applications. In many cases, MAS may be thought of as a set of autonomous entities, and, therefore, structuring the MAS as nodes of a P2P network mostly is taken as a solving solution for establishing MAS group. Considering the roles acted by the MAS system and research goals of different phases, in the initial phase aimed at model testing, a hybrid network mode, webhosting has been taken as system framework to implement interactions among agents. In the following phase (not be addressed in this thesis), pure P2P networking under the support of JXTA will be integrated in the prototype to achieve the automatical establishment of cooperative group.

4.2.1.1 Webhosting

A web hosting service is a typical application of first P2P-generation network to implement web-based sharing. As Figure 4-5 illustrates, it allows individuals and organizations to provide their own website accessible via the Web. It makes it possible to exchange files privately. In small communities, popular files can be distributed very quickly and efficiently. This P2P networking mode is applied in the first phase, model testing phase.

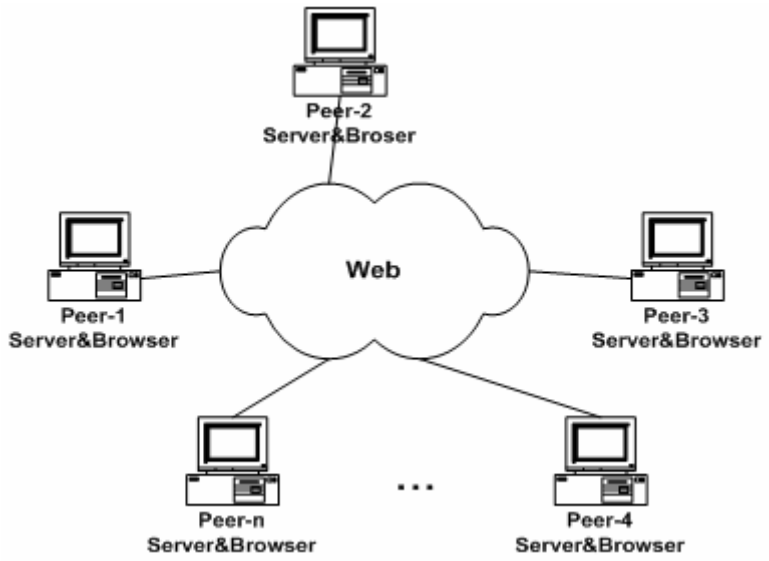


Figure 4-5: Webhosting P2P Model

4.2.1.2 Decentralized P2P

It is second P2P-generation network featured by decentralization, initially attempts to decentralize central index server. Through distribution of responsibilities of central index server to some higher-capacity nodes with lower capacity nodes branching off them, all nodes become more equal than first P2P-Generation network. It allows for large and efficient networks without central servers. As Figure 4-6 demonstrated, nodes, autonomous entities, are interacting directly via a variety of links with each other. This

type of P2P networking makes the role of "Peer Finder" able to establish Ad-hoc cooperative networks, and therefore to meet the requirements of second phase of implementing prototype.

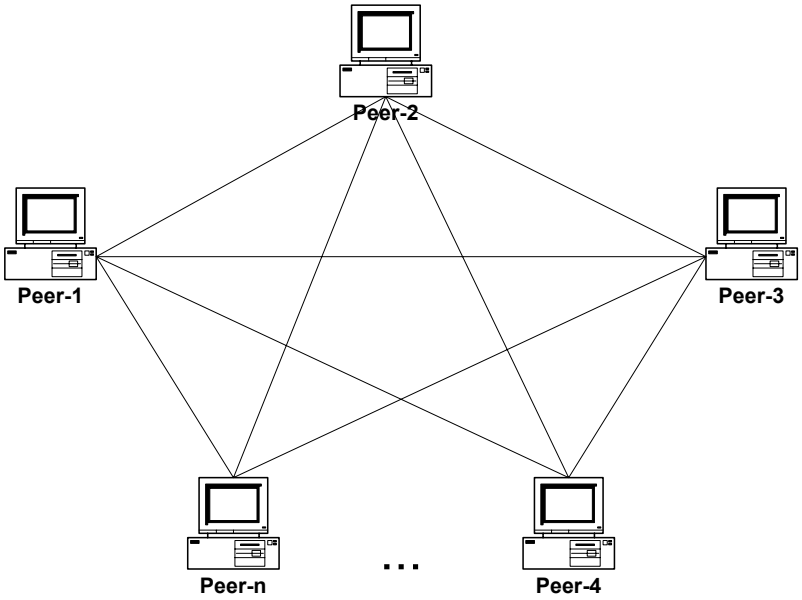


Figure 4-6: Decentralized P2P Model

4.2.2 Overview of Development Environment

Figure 4-7 shows topology of open sources utilized in different phases of development.

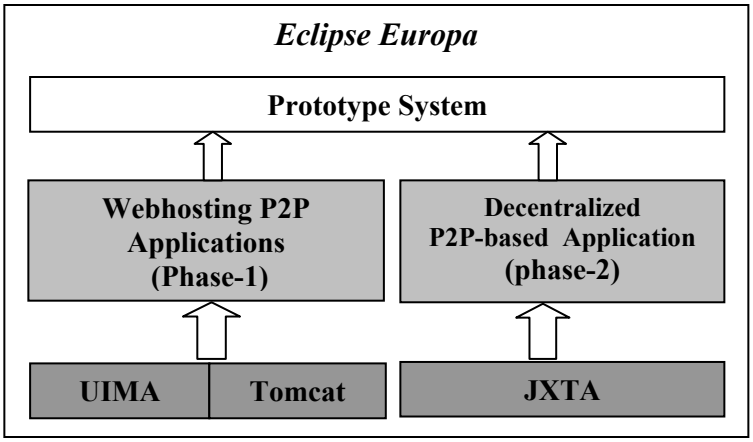


Figure 4-7: Overview of Development Environment

The IDE, IBM Eclipse Europa, is selected because it supports UIMA as well as several web servers such as Apache Tomcat, etc. (http://www.eclipse.org/europa/). In the initial phase we chose Apache UIMA and Tomcat to support the development. Apache Tomcat is applied to help deploy document annotation service on the websites.

4.2.3 Implementation of the Initial Phase

Figure 4-8 shows MVC model of prototype system. In this chapter, we mainly talk about implementation of Document Annotator, as representative of phase one.

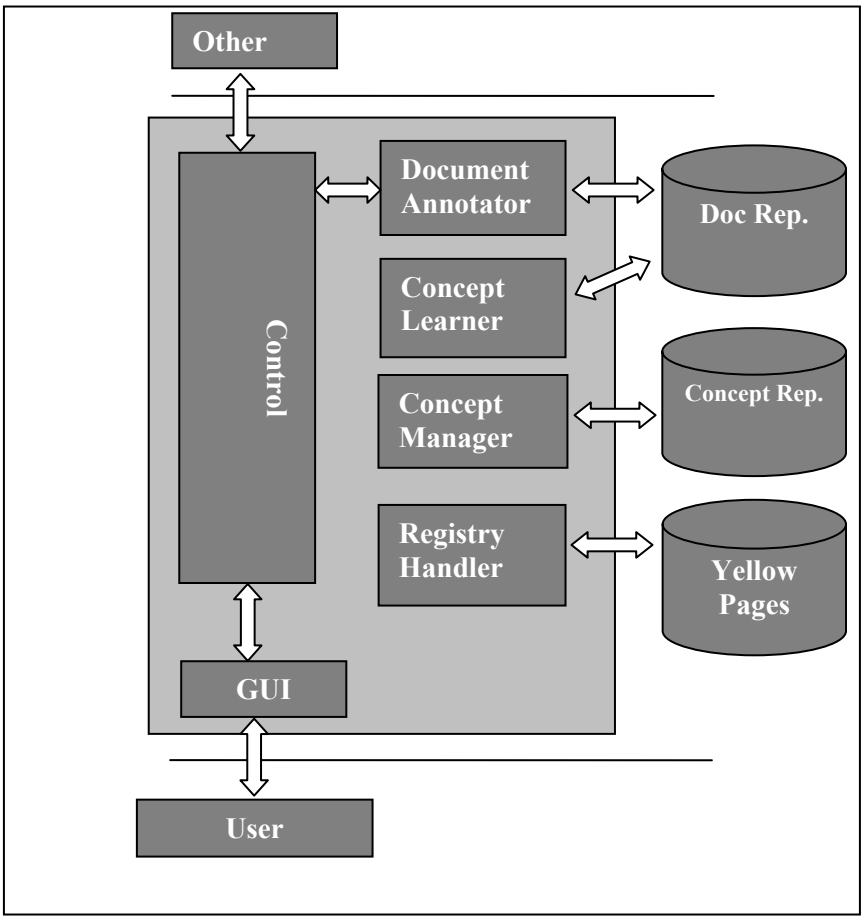


Figure 4-8: Architecture of Prototype System

4.2.3.1 Document Annotator

The central responsibilities of Document Annotator are creations of type system and annotation engine. In following paragraphs, we will introduce ways to implement them.

Creation of Type System

According to type system definition specification specified by UIMA, the first step is to build XML-based type system descriptor. Figure 4-9 shows an actual descriptor used in the prototype. This aggregate type system is composed of several basic primitive types.

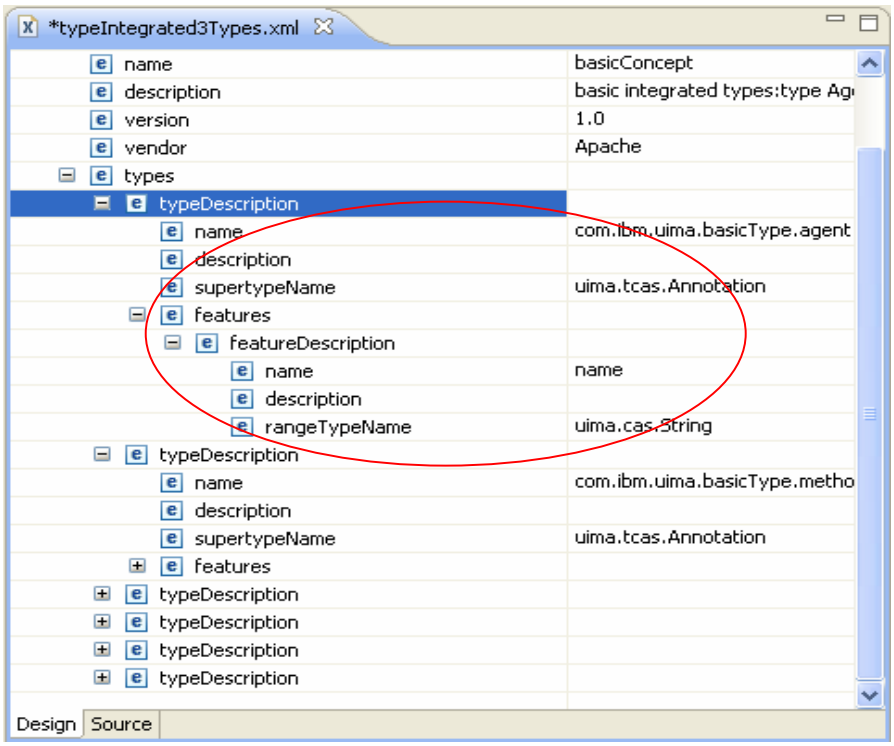


Figure 4-9: A Type System Descriptor

The lines enclosed by the red oval represent one type definition which includes *name*, *description*, *super-type* name, and its *features*. Here, super type is one built-in UIMA which can be optionally used depending on developers. UIMA provides corresponding

mechanism to build class components to dynamically adjust contents of descriptors and create a corresponding Java class.

Creation of Annotation Engine

Under UIMA frame, Creator of an annotator has to implement a standard interface which has several methods to embed processing logic into it, the most important of which are:

- initialize(),
- process(), and
- destroy()

There are two ways for creator to tell *process*() method that which types need to take and which types need to produce. One is manually creating a component descriptor which is XML-based document, like type system descriptor; another is using APIs that come along with UIMA to dynamically set this component descriptor. The latter is preferable and frequently used as creator can select input/output type system at run time. Immediately after Annotation Engine is created, the *process*() will take over process to scan documents and produce types as AE descriptor specified. Among these methods mentioned above, the *process*() is the only one required for implementation.

4.3 Discuss of the Network Architectures

Based on the practices with the prototype, I believed the hybrid P2P network architecture has some advantages from the perspective of file sharing over the centralized networking:

- Be able to leverage individual data repositories of all participants. Information can be accessible as soon as joined in the P2P group;

- As each peer handle its own data repository, They should be comfortable with the ontological heterogeneity existing in the peer group and provide accurate and up-to-date information to search query; and
- Should be able to tolerate to the single point of failure.

Coming with advantages, there are certainly some disadvantages inherent in the hybrid P2P networking:

- Extra overheads required by peer registering, peer routing, and peer group management than centralized networking;
- Security control is another issue need to be considered;

Further investigation on scalability and performance of the MAS group, costs on refinements of concepts, and accurate overheads on interactions for connection establishments are left for future works.

4.4 Study Case

In the local repository, there are several documents: some of them are about MAS methodology; some about other concepts. The initial status, as shown in Figure 8, tells that all documents, having not been annotated, are organized in flat structure.

Based on status, a regular query, as shown in Figure 4-10, is initiated. In this case, a user wants to know "what the token Prometheus means in software engineering". Processing this query with no semantic search (as depicted by empty "Concepts" box in Figure 4-11) will return two documents, with completely different contents.

The document 03 is talking about Prometheus MAS design methodology; meanwhile, the document 11 is about Greek mythology which is apparently not the one that user intended to receive.

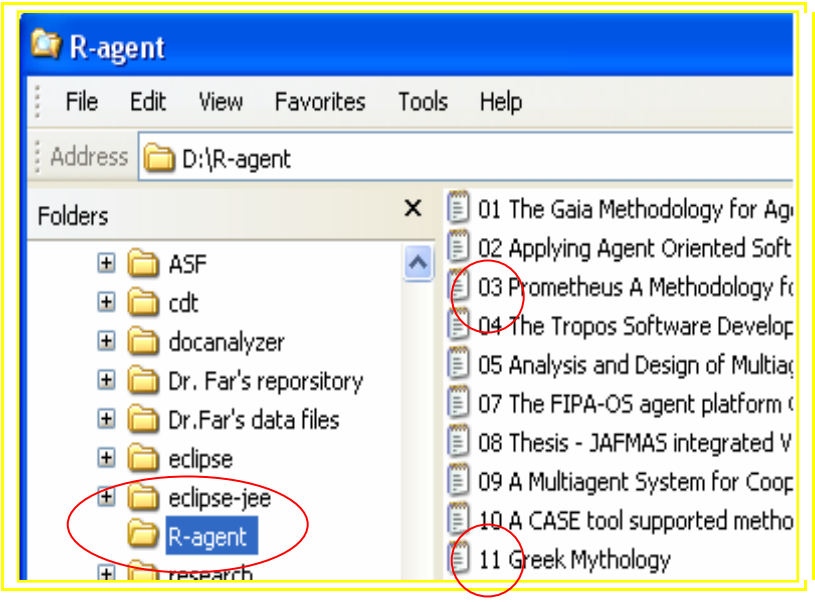


Figure 4-10: Snapshot of Initial Repository

To disambiguate search results, the agents will take the following steps to kick off a semantic search:

1. A concept learning routine is started to evaluate returned documents, through which new concepts, for example, "agent" (including its features) is identified; it then will be propagated within the cooperative group (refer to [Yang, 2008] for details).
2. Each agent, upon receiving this concept, will annotate its own repository. Annotation procedure re-categorized repository by conforming to concept hierarchy. Figure 4-11 shows changes happened to repository R-agent.
3. New concepts will be added into concept repository in order to support decoding query phrase, or searching operation later on.

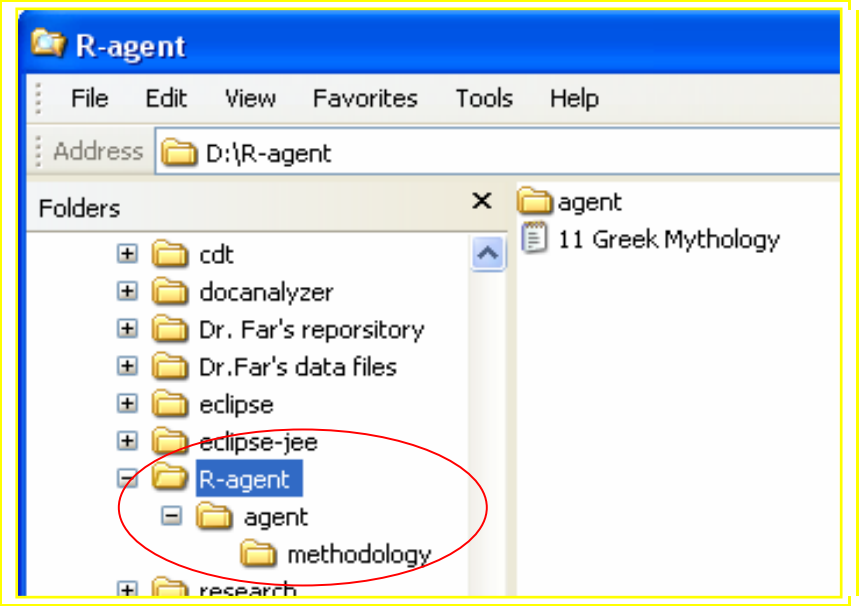


Figure 4-11: After Adjustment of Data Repository

Once these steps are completed, a next round of search will be kicked off by involving the newly learnt concepts, as shown in Figure 4-12.

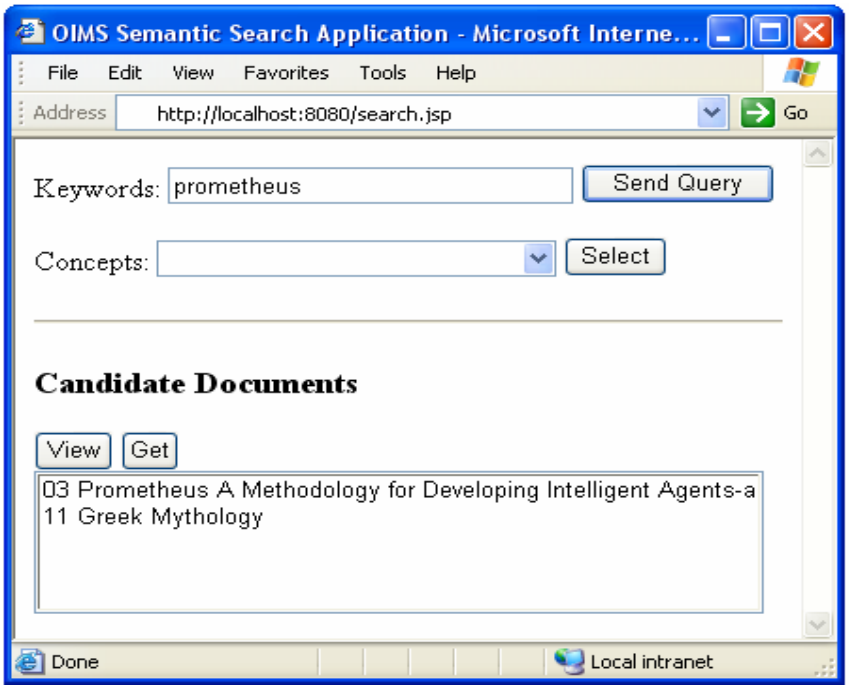


Figure 4-12: Illustration of Regular Search Procedure

Figure 4-13 shows a disambiguated result by sending query phrase consisting of both keywords and concepts.

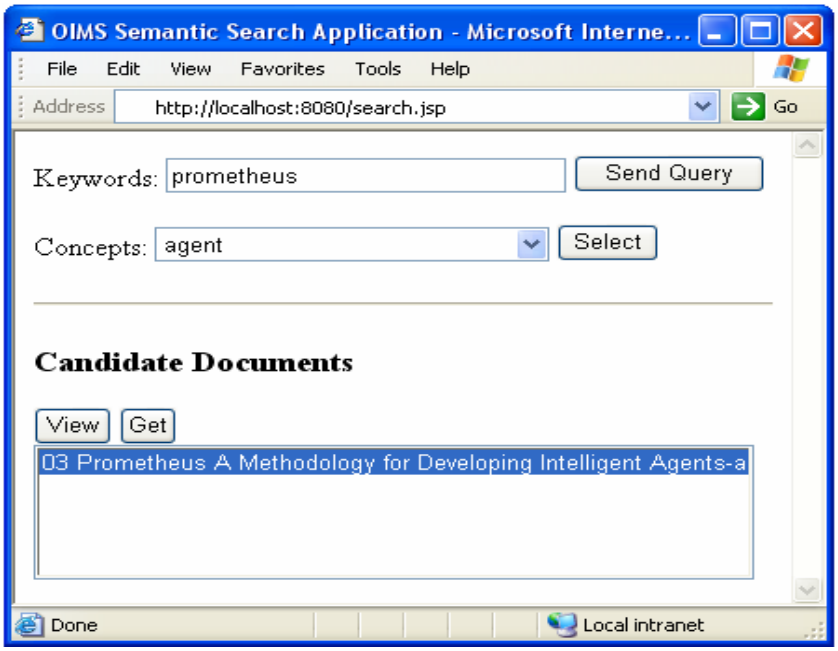


Figure 4-13: Illustration of Semantic Search Procedure

From the procedure explained above, we can conclude that dynamical annotation guided by concept hierarchy (here we just used flat structure of hierarchy) is capable of categorizing the repository, consequently, making retrieval of documents more efficient. On the other hand, efficient annotation process will help concept learning routine to identify concepts that are more accurate.

Chapter Five: Experimentation and Evaluation

The discussion in this chapter focuses on the experiments we have conducted with our prototype. We have designed three experiments using the developed prototype system to observe how the evolution of search results is influenced by concept learning and semantic search and compare them with traditional search.

5.1 Experiment Plan

5.1.1 Experiment Schema

In Experiment 1, a series of traditional queries will be processed by the agents residing on data repositories in order to observe behaviours of a traditional search and to set benchmarks for comparing with the results of other experiments.

Experiment 2 is designed to observe the concept learning stage of the spiral search process (refer to Figure 1-2). Before sending queries, a new concept is supposed to be identified through interactions between Concept Learner (CL) and Document Annotator (DA) agents, and under the guidance of the attributes of the new concept, the initial repositories are re-structured to be hierarchical repositories.

Experiment 3 is carried out using the hierarchical data repositories refined in Experiment 2. It represents the stage of semantic search of the spiral search process (refer to Figure 1-2). The queries will be processed after the annotation process in which annotators initiatively annotate data repositories they are handling with the same type system which is designed to filter documents.

The disambiguation of search results is measured by a metrics named ROD (Ratio of Disambiguation) which represents the precision of query results.

$$ROD = \frac{Pos}{Pos + Neg} \times 100\%$$

- *Pos*: The number of positive documents. The contents of a positive document meet the query conditions.
- *Neg*: The number of negative documents. A negative documents, actually, in the experimentation means *false positive document*.

The positive or negative is determined a human expert.

Each experiment will go through the following steps: (1) record of experiment data; (2) visualization of experiment results; (3) analysis of experiment results. The goal of experiment is to explore how both concept learning and annotated data repository influence the evolution of search results, comparing with expected results manually computed in advance.

5.1.2 Preparation of Test File and Domain Ontology

The test data set consists of files describing the course syllabi in Computer Science offered by three major universities [UIl]. A course syllabus file normally contains a course identifier, a course description and the prerequisites of a course. The University of Michigan organizes Computer Science (EECS) as an engineering discipline and as a joint program with electrical engineering; the University of Washington considers Computer Science (CSE) as an engineering discipline but independent from electrical engineering and as a joint program with computer engineering; in Cornell University Computer Science (CS) is a pure science program in the science faculty. The three universities together offer 279 courses in electrical engineering and/or computer science, not including some featureless courses such as seminar course. We set up three data

repositories for each university courses, with each repository having a MAS (Figure 5-1) to handle it. The Ag_C, Ag_W and Ag_M stand for Cornell University, University of Washington, and University of Michigan, respectively. The course files for Cornell University and the University of Washington were taken from [UIl], and those for the University of Michigan from their web site at [UMi]. The course catalogue used in this experiment has been created as a part of evaluation of concept learning by other team members [Afsharchi, 2007].

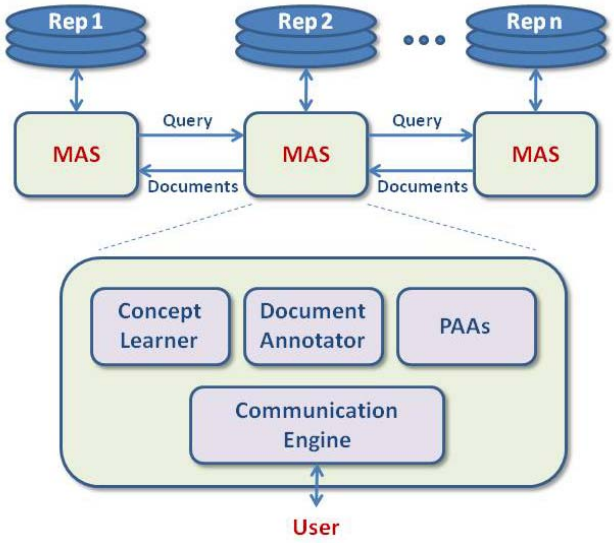


Figure 5-1: Prototype system and MAS components

5.1.3 Keyword Preparation

We defined a series of feature, a vector of keywords that are applied to scan file texts. The way of defining features is to group them to look for particular words or word combinations in the texts [Sahami, 1996], [Peng, 2003]. Using this method some keywords (i.e. terms) are combined to make a new feature.

For example, feature $f_{\text{picture, photo,figure}}$: text→Boolean is a feature which is made by combining terms, *picture, photo, and figure* and it is true for a text *t*, if either one of the

keywords occurs in *t*. This word combination allows us to come up with a fixed set of features that represents all objects of a particular concept ent. All keywords are derived from the achievement of concept learning which are selected through Document Frequency (DF) (refer to [Afsharchi, 2007]). Those keywords are used to create the feature sets for our agents regarding every concept which is being learned and are utilized to annotate data repository. Table 5-1 lists keywords for the concept Computer Science. The column priority shows the appropriate priority number for each keyword. Although there is a manual way in which a computer scientist could determine these keywords, it would not guarantee a unique and stable set of keywords.

Table 5-1: Keywords for Computer Science [Afsharchi, 2007]

Keyword	Priority
computer	**26.0**
system	**19.0**
design	**18.0**
science	14.0
performance	13.0
model	12.0
theory	12.0
parallel	12.0
algorithm	**9.0**
technology	9.0
language	**9.0**
logic	**9.0**
analysis	**8.0**
program	**8.0**
structure	6.0
synthesis	6.0
knowledge	6.0
development	6.0
process	5.0
formal	6.0
project	5.0
information	5.0
digital	5.0
software	5.0
control	5.0
complexity	5.0
circuit	5.0
data	4.0

5.2 Experimentation

5.2.1 Experiments Settings

The search goal is to find all courses in *programming languages* from the three data repositories. Features (or query phrases) utilized for all three experiments are constructed with five keywords picked up from the Table 5-1 which should be general and related to the search goal, *computer programming languages*. There are a total of five query phrases are listed in Table 5-2.

Table 5-2: Feature Set of Query

Feature ID	*Feature Content*
F1	Language
F2	Language, Program
F3	Language, Program, Computer
F4	Language, Program, Computer, Science
F5	Language, Program, Computer, Science, Software

5.2.2 Experiment 1: Traditional Search on Current Data Repository

Traditional search is conducted in Experiment 1. The results are recorded in Table 5-3, and its visualization of results is represented in Figure 5-2.

Table 5-3: Summary of Results Obtained in Experiment 1

	$\mathcal{A}g_C$			$\mathcal{A}g_M$			$\mathcal{A}g_W$		
	Pos.	**Neg.**	**%**	**Pos.**	**Neg.**	**%**	**Pos.**	**Neg.**	**%**
F1	4	1	80	4	16	20	4	15	21
F2	6	2	75	8	30	21	6	28	18
F3	6	5	55	8	55	13	6	50	11
F4	6	7	46	8	56	13	6	51	11
F5	6	7	46	8	62	13	6	55	11

Examining the record of Experiment 1, we can find that the ratio of disambiguation of $\mathcal{A}g_C$ is much higher than the $\mathcal{A}g_M$ and $\mathcal{A}g_W$. We think that this is caused by different

composition of data repositories. $\mathcal{A}g_C$ actually holds courses of pure computer science, whereas $\mathcal{A}g_M$ and $\mathcal{A}g_W$ manage courses with composition of both computer science and electrical engineering.

In addition, it is worth to mention that:

1) The more terms added to each query, the more documents were retrieved, regardless of whether the documents were positive or negative.
2) Ratios of disambiguation were not guaranteed to be improved with terms added to the query. In this case, it caused the ratios to get worse by adding more terms.
3) All three data repositories were isolated so the number of positive documents was definite. The queries with feature F2 obtained all positive documents in the repositories. After that, no other positive document could be found and search noise made results worse.

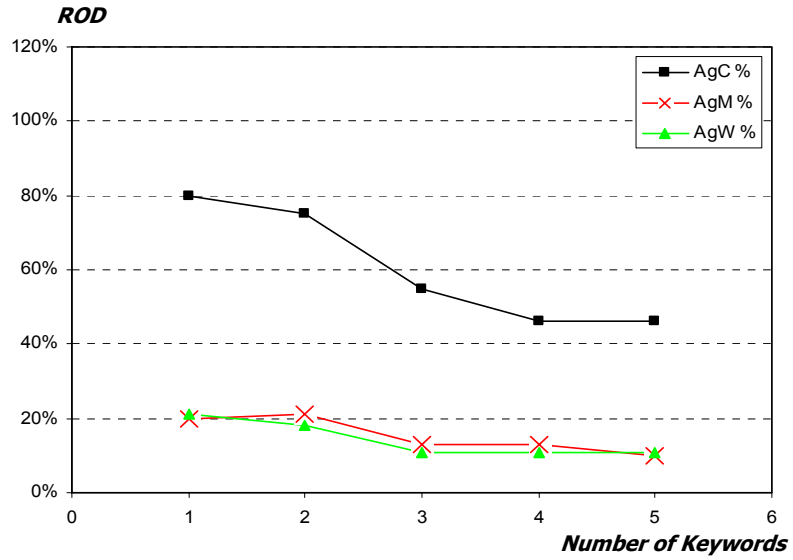


Figure 5-2: Visualization of the Results of Experiment 1

From Experiment 1 we can conclude that the **composition** of data repository influences the search results, confirming that the expected results are significantly correlated with the data repository.

5.2.3 Experiment-2: Search with concept learning

Experiment 2 focuses on examining the behavior of queries, when the Concept Learner has been introduced. The algorithm built into the Concept Learner takes the same data repositories as in Experiment 1 to identify a new concept, *Computer Science*, and then using it to identify all its subcategories. Then using the learnt concept, the data repositories are screened and reorganized for the subcategories of Computer Science listed in Table 5-4 [Afsharchi, 2007]. One subcategory, the *programming languages*, is directly adopted to annotate data repositories when annotating action is performed.

Table 5-4: Subcategories of Course of Computer Science after Applied Concept Learning [Afsharchi, 2007]

Computer Programming I Design and Analysis of Algorithms II Computer Science Research Seminar Introduction to Artificial Intelligence Computer Networks Introduction to Computer Organization Computer Architecture Foundations of Computer Science Interactive Computer Graphics Applied Logic Theory of Computing Computer System Performance Computer Game Design and Development Parallel Computing Computational Geometry Introduction to Formal Models in Computer Science	Computational Molecular Biology Computational Tools and Methods for Finance Computers and Society Intelligent Transportation Systems Introduction to Logic Design Reliable Computing Systems

Through the manipulation of concept learning, the initial flat data repositories were restructured to a two-level hierarchy as illustrated in Figure 5-3. We repeated the

queries as in Experiment 1 on these structured data repositories. These queries were no longer traditional because at this point any query would have been assumed by the search engine to be a query for all courses of computer science. In practice, new concept (in this case *Computer Science*) will be involved in each query feature to semantically describe it.

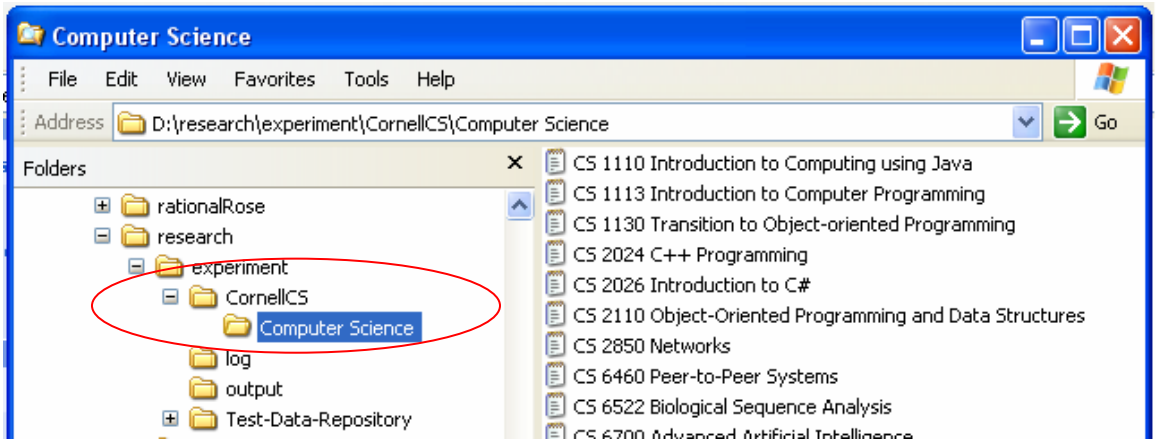


Figure 5-3: Hierarchy of the Data Repository $\mathcal{Ag}_C$ (Cornell University) for Experiment 2

The results of Experiment 2 are listed in Table 5-5 and Figure 5-4, and from them, we can figure out the following:

Table 5-5: Summary of the Results in Experiment 2

	$\mathcal{Ag}_C$			$\mathcal{Ag}_M$			$\mathcal{Ag}_W$		
	Pos.	**Neg.**	**%**	**Pos.**	**Neg.**	**%**	**Pos.**	**Neg.**	**%**
F1	4	0	100	4	4	50	4	6	40
F2	6	0	100	6	10	38	6	13	32
F3	6	3	67	6	13	32	6	19	24
F4	6	4	60	6	13	32	6	19	24
F5	6	4	60	6	20	23	6	19	24

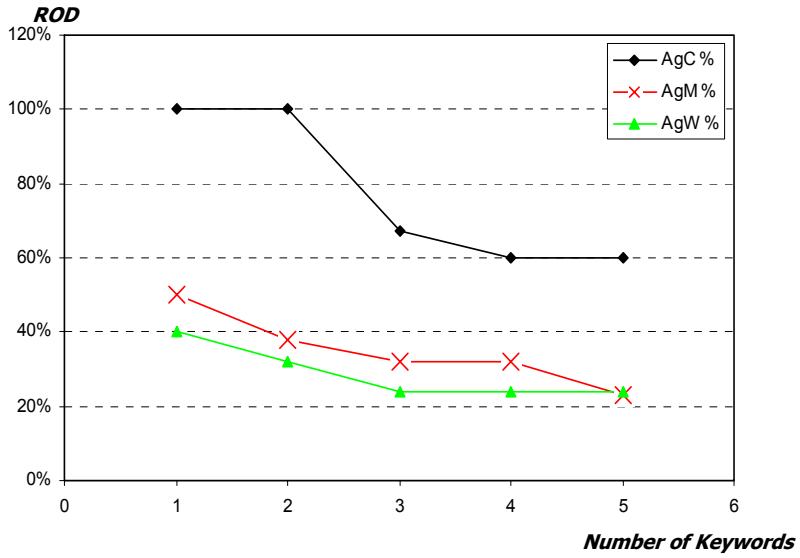


Figure 5-4: Visualization of the Results of Experiment 2

For Experiment 2 we can conclude that:

1. RODs have been improved for all the three repositories and for all the queries. Intuitively, as shown in Figure 5-4 all the lines representing trends of change of RODs have shifted up significantly.
2. Variations of ROD are still following the same trend as in Experiment 1 (i.e. with the terms added to query, the RODs are decreasing).

In order to quantitatively measure the improvement of ROD, we computed the increasing ratio of RODs of Experiment 2 to its counterpart of Experiment 1, with the results listed in Table 5-6.

Table 5-6: Increment of Ratios of Disambiguation (Experiment 2 vs. Experiment 1)

	Ag_C			Ag_M			Ag_W		
	R1	**R2**	**R**	**R1**	**R2**	**R**	**R1**	**R2**	**R**
F1	80	100	*25*	20	50	*150*	21	40	*90*
F2	75	100	*33*	21	38	*81*	18	32	*78*
F3	55	67	*22*	13	32	*146*	11	24	*118*
F4	46	60	*30*	13	32	*146*	11	24	*118*
F5	46	60	*30*	13	23	*77*	11	24	*118*

Based on the analysis of the results of Experiment 2, we believe that Concept Learner, at least in this case, plays a significant role for improving the composition of data repositories through reconciling the conflicts between the data repositories. Guided by attributes of the new concept, some irrelevant documents had been filtered out, therefore effectively suppressing the noises impacting on the query result. The results listed in Table 5-6 present the results we expected with the RODs of both Ag_M and Ag_W having increased significantly more (with maximum values of column R are 150% and 118% for Ag_M and Ag_W, respectively) than Ag_C (with the maximum being 33%).

We believe the cause that directly brought increasing ratios of RODs to be different is the differences of composition between the data repositories as well. In the data repositories mixing courses of both disciplines *Computer Science* and *Electrical Engineering* such as Ag_M and Ag_W, irrelevant courses (e.g. electrical courses) would be eliminated more effectively and obviously than that of pure data repository as those of Ag_C, only holding courses of computer science. We also noticed that even though all courses of data repository of Ag_C are of computer science, there are still two courses were removed (CS 6740: Advanced Language Technologies; and CS 6764: Reasoning About Knowledge) and were not supposed to be moved from the data repository. We thought that is a problem relating to the accuracy of concept learning algorithm.

5.2.4 Experiment-3: Search with document annotation

From the results of Experiment 2, we concluded that through applying concept learner, search performance improved. However, the trends of ROD evolvement remain the same as in Experiment 1.

In this experiment, we apply the Document Annotator (DA) agent to semantically determine if a document is about the searched concept or not, and to see how search performance would be influenced.

Experiment 3 was carried out based on the data repositories refined in Experiment 2 in which the course description documents of computer science were re-categorized under the directory marked by the concept *Computer Science*. At the beginning of the Experiment 3, each data repository was annotated with the same UIMA type system (i.e. kind of concept hierarchy). An aggregate annotator was established consisting of a series of primitive annotators for annotating terms including *language*, *program*, *C*, *C++*, and *Java*. As all the documents to be scanned and relocated, were already under computer science, we were able to replace those non-domain specific terms (computer, software, and science) with those specific terms of the domain computer science (*C*, *C++*, and *Java*). The following expression illustrates a typical annotation logic of the aggregate annotator:

<Language + Program + [C|C++|JAVA] ➔ Computer Programming Course>

This can be interpreted as: "if a three-concept entity created through some logic built in the annotator has been found in the document, then this document was determined to be a target course, i.e. description of *computer programming language*."

Once the annotation process was completed, the corresponding alteration to current data repositories was made. Documents that had not been annotated successfully were removed from the sub-directory dedicated to computer programming language course description. Hence, the ratios of positive documents were raised and the noise that was

brought in by adding terms to the query was reduced. Table 5-7 shows the changes of data repositories before and after being annotated.

Table 5-7: The States of Data Repositories (A): The Number of Positive Documents; (B) Total Documents Remained after Annotation; (C) The Ratio of Positive Documents Accounting for Total Documents. Lower cases represent the results before annotating

	a	b	c	A	B	C
$\mathcal{A}g_{C,}$	6	14	*43%*	6	6	*100%*
$\mathcal{A}g_{M}$	6	25	*24%*	6	16	*38%*
$\mathcal{A}g_{W}$	6	29	*21%*	6	19	*32%*

Those ratios listed in column C reflect the minimum RODs of queries later on, no matter how many noises were brought in by adding terms to the feature. For instance, after annotation, the total number of documents of $\mathcal{A}g_{C}$ dropped down form 14 to 6, and the ratio increases from 43% to 100%.

Through the Experiment 2 and the annotation process of the Experiment 3, data repositories were structured with two levels instead of initial flat structures as shown in Figure 5-5.

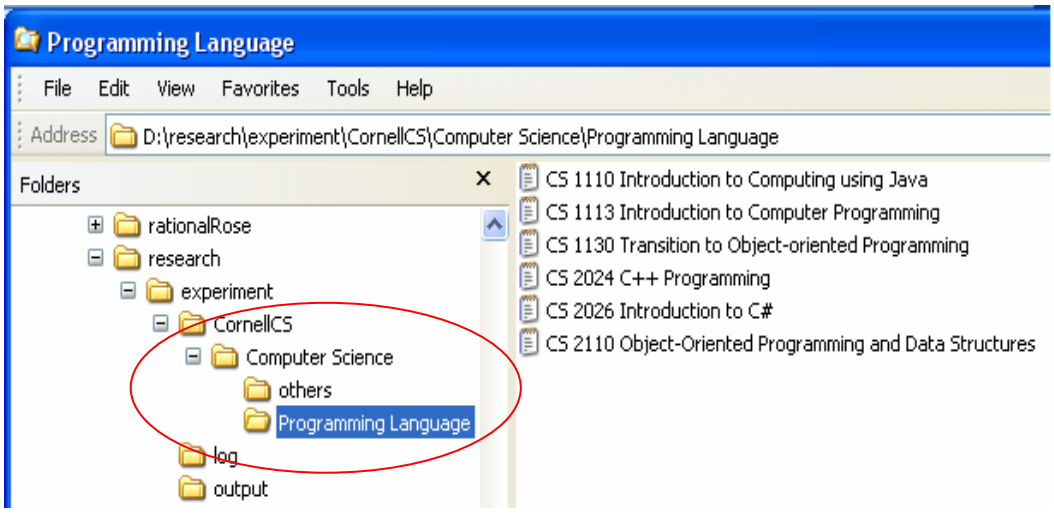


Figure 5-5: Hierarchy of the Data Repository Agc (Cornell University) for Experiment 3

This two-level hierarchy are formed by applying concept learner in Experiment 2 and annotation in Experiment 3. Following this part of the experiment, we continued to query with the same set of features on the structured data repositories in order to compare the results with those of previous experiments.

These queries, similar to those in Experiment 2 triggered a semantic search towards structured data repositories, since some terms in features were treated as concept instead of regular terms, and under the guidance of search handler agents, the query was directly pointed to the correct locations.

The results of queries are listed in Table 5-8, with the visualization of these results presented in Figure 5-6. Compared with Experiment 2, we noticed that RODs for the first two queries (with F1-F2) remained the same as the results of Experiment 2, but the RODs of the last three queries with features F3-F5 achieving improvement and staying the same as F2.

Table 5-8: Summary of the Results in Experiment 3

	Ag_C			Ag_M			Ag_W		
	Pos.	**Neg.**	**%**	**Pos.**	**Neg.**	**%**	**Pos.**	**Neg.**	**%**
F1	4	0	100	4	4	50	4	6	40
F2	6	0	100	6	10	38	6	13	32
F3	6	0	100	6	10	38	6	13	32
F4	6	0	100	6	10	38	6	13	32
F5	6	0	100	6	10	38	6	13	32

The same trends as in Experiment 2 are reflected in Figure 5-6, except that all lines go horizontally beginning from X-coordinate *2*.

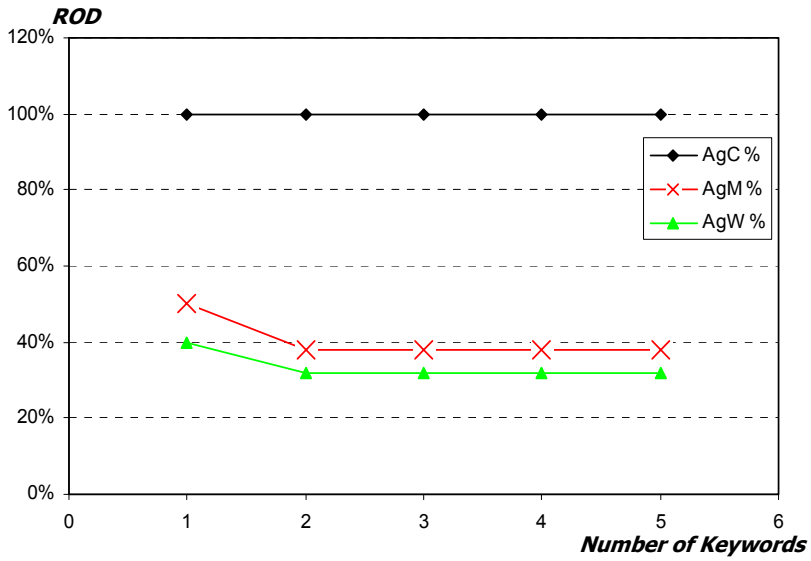
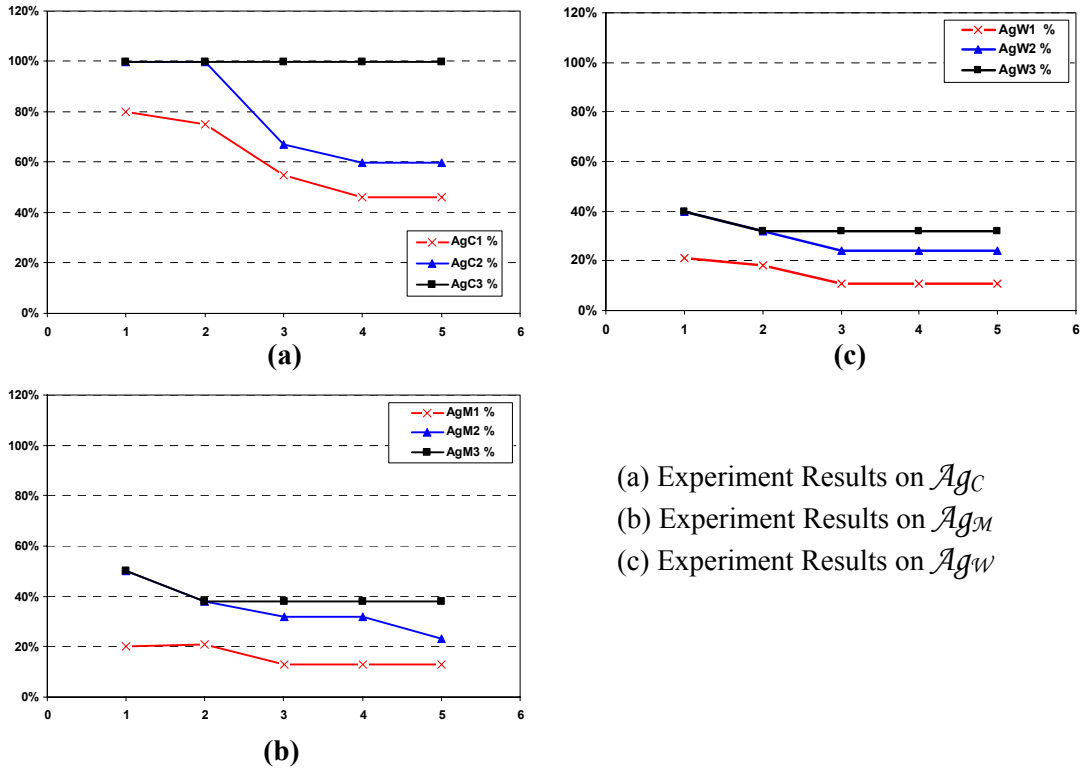


Figure 5-6: Visualization of the Results in Experiment 3

That means adding terms to features no longer brings noises as in previous experiments because the sources of noise (irrelevant documents) have been removed from current locations.

5.3 Experiment Evaluation and Summary

Contribution to the improvement of search results made by both concept learning and annotation is the main concern. Considering that isolated evaluations of either concept learning or annotation would not make sense because the actual working process should be a consecutive generative/spiral process. In order to evaluate contributions made by concept learning and annotation, the percentages of increment of ROD for each query and their average, contributed by concept learning and annotation, are computed respectively. The results are listed in Figure 5-7 and Table 5-9.



(a) Experiment Results on $\mathcal{A}g_C$
(b) Experiment Results on $\mathcal{A}g_M$
(c) Experiment Results on $\mathcal{A}g_W$

Figure 5-7: Visualizations of Experiment Results for Single Data Repositories

As the ΔR1 and ΔR2 indicate in Table 5-9,

- The average rate of increase of ROD achieved through concept learning on $\mathcal{A}g_C$ (28%) is much less than those on $\mathcal{A}g_M$ (121%) and $\mathcal{A}g_W$ (121%).
- The average rate of increase of ROD achieved through annotation on $\mathcal{A}g_C$ (%37) is larger than those on $\mathcal{A}g_M$ (21%) and $\mathcal{A}g_W$ (20%).
- Both concept learning and annotation made almost identical contributions on data repositories $\mathcal{A}g_M$ and $\mathcal{A}g_W$.

The reason is that $\mathcal{A}g_M$ and $\mathcal{A}g_W$ are mixed data repositories; therefore concept learning had more significant effect on them than on $\mathcal{A}g_C$. However, later in the spiral process,

composition of the three data repositories becomes increasingly similar, and consequently, annotation affected the results similarly.

Table 5-9: Quantitative Evaluation of Improvements of ROD (R1: Values of ROD of Experiment 1; R2: Values of ROD of Experiment 2; R3: Values of Experiment 3; ΔR1: (R2-R1)/R1 * 100%; ΔR2: (R3-R2)/R2 * 100%)

Data Repository	R1(%)	R2(%)	R3 (%)	ΔR1(%)	ΔR2(%)
$\mathcal{A}g_C$	80	100	100	25	0
	75	100	100	33	0
	55	67	100	22	49
	46	60	100	30	67
	46	60	100	30	67
	Avg.			**28**	**37**
$\mathcal{A}g_M$	20	50	50	150	0
	21	38	38	85	0
	13	32	38	146	19
	13	32	38	146	19
	13	23	38	77	65
	Avg.			**121**	**21**
$\mathcal{A}g_W$	21	40	40	90	0
	18	32	32	78	0
	11	24	32	118	33
	11	24	32	118	33
	11	24	32	118	33
	Avg.			**121**	**20**

Chapter Six: Conclusion and Future Works

In this chapter we present the conclusions of the thesis, focusing on our semantic search solution that integrates search with concept-learning mechanism. We further identify research that needs to be done in the future to improve our semantic search solution.

6.1 Conclusions

As described in Chapter 1, the goal of this thesis is to devise a semantic search solution by providing a model addressing implementation of multi-agent system (MAS) serving semantic search with the support of annotation and concept-learning, and to implement a prototype MAS system conforming to this model. We had three main objectives:

1. Devising individual autonomous agents that are capable of learning ontological concepts from several teacher agents through interaction with other agents and validating these concepts to better communicate and share information in the future.
2. Developing a semantic search engine capable of dynamically annotating the data repositories that they are handling within MAS.
3. Integrating method or mechanism needed to support and facilitate the implementation of complex interactions among agents.

Contributions of this thesis are:

1. **Revealing the inherent relationship between concept learning and semantic search** in an ontological heterogeneous environment in which concept learning and semantic search play their roles equally evolving in a spiral-like process as shown in Figure 1-2. They support each other to achieve their own goals by enriching the set of ontological concepts and reducing ambiguity of the search results, respectively. We have

clarified that both concept learning (or ontology learning in case of learning both concepts and relationships between concepts) and semantic search should be taken into account together in a comprehensive semantic search solution. In other words, we should reconsider current methods that use concept learning and semantic search in isolation. The results of experiments presented in Chapter 5 showed that through the alternating concept learning and search actions in one round of the spiral process, the search results improved significantly.

2. **Applying layered architecture of semantic interoperability to semantic search and concept learning** allows for incremental improvement of semantic search that goes beyond the lexical and syntactic layers. As the layered architecture is theoretically solid from AI perspective, it makes the implementation of semantic interoperability possible layer by layer through the "divide-and-conquer" strategy, and makes the project possible by focusing on the lexical layer for the first step. This lays the foundation for further study on the rest of the layers.

3. Finally, with regard to the analysis, design, and implementation of the MAS prototype, we **designed and implemented a number of prototype MAS** by exploring useful open sources such as LUCENE, JENA, JXTA, and UIMA. Collection of development experience will benefit further upgrade of such a system.

6.2 Future Works

Based on contributions and experience gained from the implementation of the prototype we present the following suggestions for future research in this area.

- **Test needed to be done to evaluate the scalability and performance of the prototype. As** the MAS addressed in this thesis is working with fundamentally different networking from the current public search engines such as Google and Yahoo which is featured providing centralized and indirect search service, the corresponding measures towards the evaluation of the MAS need to be devised, also.
- **Based on the exercises on the prototype, develop a complete communication protocols on lexical layer** to regulate the semantic interoperations among users, concept learning, and semantic search agents, with the upgrade of algorithms of both concept learning and dynamical annotation. Without it, the layered architecture and MAS application mode suggested in this thesis would not be widely recognized. In this document, the semantic interoperations focus on the lexical layer of semantic interoperability that the concept learning algorithm applied in MAS prototype only supports encoding/decoding concepts in flat-structure, regardless of any hierarchical structure of concepts; and it demands a primitive protocol to regulate the interactions with semantic search engine. In future, with more study on concept learning related algorithms, an interactive session should be generalized, and as result, a complete, practical protocol family will be created. In the near future, for lexical layer, we suggest adding syntax to describe hierarchy of concepts to the message format.
- **Upgrade agent PeerFinder with new version of JXTA resource,** and integrate it with current prototype system to enable MAS to form groups without the helps of central indexing server.
- From the technique perspective, we need **seek an integrated solution to maintain concepts, concept relations, and even complete ontology.** An appropriate solution

would be a repository management system, which supports application modules such as concept learning and query handler, and randomly retrieves each single entry to facilitate access to ontology related information.

- **Once the protocols of lexical layer are getting mature, proceed to higher layers**. In the long run, a complete protocol family guiding semantic interoperations at each layers needs to be established. To achieve it, we suggest a development strategy in which protocols should be developed layer by layer instead of unrealistically designing the protocols for all layers at one time. The one important advantage brought by layered structure is that when developing the protocol of lower layer and proving it solid enough to publish, we do not need to worry about higher layers, as it will not affect development of protocols for higher layers later on because functionalities have already been converged at each layer and relative boundaries between them have already been formed. For instance, the protocol HTTP protocol, a well known Web protocol, was created based on internet protocol such as TCP/IP and was published much later than it; however, it was successfully developed independently.

References

[Afsharchi, 2007] M. Afsharchi, Ph.D. dissertation, "Ontology-Guided Collaborative Concept Learning in Multi-Agent Systems," Department of Electrical and Computer Engineering, University of Calgary, 2007.

[Afsharchi, 2006] M. Afsharchi, B.H. Far, and J. Denzinger, "Ontology Guided Learning to Improve Communication among Groups of Agents," *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'06)*, Hakodate, Japan, pp. 923-930, 2006.

[Afsharchi, 2006A] M. Afsharchi, B.H. Far, and J. Denzinger, "Learning Non-Unanimous Ontology Concepts to Communicate with Groups of Agents," *Proc. IAT'06, IEEE Press*, pp. 211-217, 2006.

[Apache, 2006] http://incubator.apache.org/uima.

[Berners, 1999] T. Berners-Lee and M. Fischetti, "Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor, " *HarperCollins Publisher*, New York, 1999

[Berners, 2001] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, May 2001, pp. 34–43.

[Brickley, 2001] D. Brickley, R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," *W3C Recommendation*. http://www.w3.org/TR/PR-rdf-schema, 2004.

[Corcho, 2005] O. Corcho, "A Layered Declarative Approach to Ontology Translation with Knowledge Preservation," *IOS Press*, ISBN 1-58603-477-4, 2005.

[Daconta, 2003] M.C. Daconta, L. J. Obrst, and K. T. Smith, "The semantic Web. A Guide to the Future of XML, Web Services and Knowledge Management," Indianapolis, Wiley Publishing, ISBN 0-471-43257-1, 2003.

[Dean, 2004] M. Dean, G. Schreiber, "OWL Web Ontology Language Reference. W3C Recommendation," http://www.w3.org/TR/owl-ref, 2004.

[Decker, 2000] S. Decker, S. Melnik, F. V. Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "Semantic Web: The Roles of XML and RDF," *IEEE Internet Computing*, 2000.

[Denzinger, 2002] J. Denzinger and S. Ennis, "Being the New Guy in An Experienced Team: Enhancing Training on The Job," In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS02)*, pages 1246–1253. ACM, 2002.

[Euzenat, 2001] J. Euzenat, "Towards a principled approach to semantic interoperability," *A. Gomez-Perez et al (eds.) IJCAI2001 Workshop on Ontologies and Information Sharing*, Seattle, Washington, 2001.

[Far, 2007] B.H. Far, A.H. Elamy, N. Houari and M. Afsharchi, "Adjudicator: A Statistical Approach for Learning Ontology Concepts from Peer Agents," *The 19th*

Int. Conf. on Software Engineering and Knowledge Engineering SEKE 2007, 2007.

[Fleming, 2005] N. Fleming, "Coping with a Revolution: Will the Internet Change Learning?," Lincoln University, Canterbury, New Zealand, 2005.

[Genesereth, 1987] Michael, R. Genesereth, and N. J. Nilson, "Logical Foundation of Artificial Intelligence," Kauffman Publishers. Inc. Palo Alto. CA, 1987.

[Gomez, 2002] A. Gomez-Perez, "A Survey on Ontology Tools, OntoWeb deliverable D1.3." http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip, 2002.

[Guha, 2003] R. Guha, R. McCool, and E. Miller, "Using the Semantic Web: Semantic Search," In *Proceedings of the 12th international conference on World Wide Web WWW '03', ACM Press*, 2003.

[Horrocks, 2000] I. Horrocks, D. Fensel, F. Harmelen, S. Decker, M. Erdmann, and M. Klein, "*OIL in a Nutshell,"* In: *Dieng R, Corby (eds) 12th International Conference In Knowledge Engineering Management (EKAW'00), Juan-Les-Pins, France. Springer-Verlag, Lecture Notes in Artificial Intelligence (LNAI) 1937*, Berlin, Germany, pp 1-16, 2000.

[Horrocks, 2001] I. Horrocks, F. V. Harmelen, "Reference Description of the DAML+OIL (March 2001) Ontology Markup Language," *Technical Report*. http://www.daml.org/2001/03/reference.html, 2001.

[IBM, 2007] IBM, "Unstructured Information Management Architecture (UIMA)", http://domino.research.ibm.com/comm/research_projects.nsf/pages/uima.index.html, 2007.

[Jim, 2000] K.C. Jim, C.L. Giles, "Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem," *Artificial Life 6(3)*, pp. 237–254, 2000.

[Kant, 1965] I. Kant, "Critique of Pure Reason," *Trans by N. K. Smith, St. Martin's Press,* 1965.

[Lassila, 1999] O. Lassila, R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification," *W3C, Recommendation*. http://www.w3.org/TR/REC-rdf-syntax.

[Maier, 2004] R. Maier, "Knowledge Management System, Information and Communication Technologies for Knowledge Management," Second Edition, ISBN 3-540-20547-0, 2004.

[Oram, 2008] A. Oram, Editor at O'Reilly Media, http://www.oreillynet.com/onlamp/blog/2003/12/the_search_for_searchs_next_ge.html, 2008.

[Peng, 2003] F. Peng and D. Schuurmans, "Combining Naive Bayes and N-gram Language Models for Text Classification," In *Proceedings of The 25th European Conference on Information Retrieval Research (ECIR03)*, 2003.

[Roderick, 1982] C. Roderick, "Knowledge as Justified True Belief," *The Foundations of Knowing. Minneapolis: University of Minnesota Press*, ISBN 0816611033, 1982.

[Russell, 1995] S. Russell and P. Norvig, "Artificial Intelligence A Modern Approach," Prentice Hall, 1995.

[Sahami, 1996] M. Sahami, "Learning Limited Dependence Bayesian Classifiers," In *Proceedings of Second International Conference on Knowledge Discovery in Databases*, 1996.

[Steels, 1998] L. Steels, "The origins of ontologies and communication conventions in multi-agent systems," *Autonomous Agents and Multi-Agent Systems, 1(2):169–194*, 1998.

[Stewart, 1991] Tom Stewart, "Brainpower: How Intellectual Capital is Becoming America's Most Important Asset," *Fortune Magazine,* 1991.

[Studer, 1998] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods," *IEEE Transactions on Data and Knowledge Engineering* 25(1-2): 161-197, 1998.

[TechCrunch] http://www.techcrunch.com/2008/07/14/google-bucket-testing-new-digg-like-search-interface/

[UIl] Illinois Semantic Integration Archive. http://anhai.cs.uiuc.edu/archive/ . As seen on Sep 20, 2006.

[UIMA, 2007] UIMA Overview & SDK Setup Version 2.2.1-incubating, Apache UIMA Development Community, Published December, 2007

[UMi] University of Michigan Academic Units. http://www.umich.edu/units. php. As seen on Sep 20, 2006.

[Williams, 2004] A. B. Williams, "Learning to Share Meaning in a Multi-Agent System," *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.

[Wooldridge, 2000] N. Wooldridge and D. Kinny, "The GAIA methodology for Agent-Oriented Analysis and Design," 2000.

[Yang, 2008] Z. Yang, C. Zhong, B. H. Far, "A Practical Ontology-Based Concept Learning in MAS," *IEEE CCECE'08: Symposium on Computer Systems and Applications*, 2008.

[Zhong, 2008] C. Zhong, Z. Yang, M. Afsharchi, and B. H. Far, "Ontology-learning Supported Semantic Search using Cooperative Agents", *The 20th Int. Conf. on Software Engineering and Knowledge Engineering SEKE 2008*, 2008.

APPENDIX A: ROLE SCHEMAS OF AGENT ROLES UNDER GAIA METHODOLOGY

A.1. The Role Schema of Query Handler

Role Schema: QueryHandler (QH)

Descriptions:

Receives query request and oversees to ensure appropriate documents returned to users.

Protocols and Activities:

GetQueryRequirment, EncodeQueryPhrase, equireYellowPage, SendQuery, ReturnDocuments

Permissions:

reads	**supplied**	*userRequirments*	*// what user wants*
reads	**supplied**	*yellowPage*	*// other peers' accessible addresses*
generates		*formalQueryPhrase*	*// encoded query phrase complying // with certain syntax*
returns		*documents*	*// documents satisfy query // conditions*

Responsibilities

Liveness:

QueryHandler = ((GetQueryRequirement. EncodeQueryPhrase) • [RequireYellowPage] • (SendQuery. ReturnDocuments)) $^{\omega}$

Safety:

- **true**

Figure A-1: Schema for Role of Query Handler

A.2. The Role Schema of Concept Manager

Role Schema: ConceptManager (CM)

Descriptions:

Responsible for maintain a local Concept Repository, including creation of taxonomies of interested domains, integration of new concept, and retrieve concept(s) according to a set of features.

Protocols and Activities:

CreateTaxonomy, GetRequest, AddConcept, DeleteConcept, UpdateConcept, ResponseRequest.

Permissions:

reads *retrieveRequirement* // *what concepts are needed*
returns *concepts* // *concepts belong to taxonomy*

Responsibilities

Liveness:

ConceptManager = (CreateTaxonomy +) • (MaintainConcept) $^{\omega}$ ||
(ResponseRetrieve) $^{\omega}$

MaintainConcept = (AddConcept, DeleteConcept, UpdateConcept)

Safety:

- **true**

Figure A-2: Schema for Role Concept Manager

A.3. The Role Schema of Register Handler

Role Schema: RegisterHandler (RH)

Descriptions:

For centralized registration management, accepts registration applications, maintain registration information, and response queries for other peers; for distributed P2P paradigm, maintain all information to access other peers in the group.

Protocols and Activities:

AdvertiseService, AddRegistryInfor. UpdateRegistryInfor, DeleteRegistryInfor, ReplyQuery.

Permissions:

reads	**supplied**	*registractionApplication*	// registration-related information.
generates:		*entryOfRegistraction*	// an entry of yellow page.
		advertisement	// announcement of yellow page service.

Responsibilities

Liveness:

RequestHandler=((AdvertiseService)+) • (MaintainRegistryInfor, ReplyQuery) $^{\omega}$

MaintainRegistryInfor = (AddRegistryInfor• UpdateRegistryInfor• DeleteRegistryInfor)

Safety:

- **true**

Figure A-3: Schema for Role Register Handler

A.4. The Role Schema of Peer Finder

Role Schema: PeerFinder (PF)

Descriptions:
Responsible for detect central RH agent; or detect potential cooperative peers through wide network connections.

Protocols and Activities:
BroadcastIndividualInfor, CollectPeerInfor, RegisterIndividualInfor

Permissions:

broadcasts:	*personalInfor*	*// personal information to identify itself*
collect/reads:	*peersInfor*	*// other peers identification information*
		// or yellow page service information

Responsibilities:
Liveness:

PeerFinder= [(BroadcastPersonalInfor || CollectPeerInfor)$^{\omega}$] |
[(RegisterPersonalInfor)+]

Safety:

- **true**

Figure A-4: Schema for Role Peer Finder

A.5. The Role Schema of User

Role Schema: User (USER)

Descriptions:
Organization or individual initiating a query.

Protocols and Activities:
RetrieveConcept, InitiateQuery

Permissions:

reads	*concepts.*	*// concepts used to create a query request*
generates	*userRequirements*	*// keywords and concepts used to create a query request*

Responsibilities
Liveness:
User = [RetrieveConcept] • (InitiateQuery) +

Safety:
- **true**

Figure A-5: Schema for Role USER