City University of Hong Kong
Department of Computer Science

BSCS Final Year Project

Project Report

BSCS (FT)

Project Title:

Semantic Web

(Volume 1 of 1)

Student Name : **Kwong King Hei Francis**

Student No. : **50190500**

Programme Code : **BSCS (FT)**

For Official Use Only

Supervisor : **Dr. Chun, Andy H W**

1st Reader : **Dr. Yu, Y T**

2nd Reader : **Prof. Ip, Horace H S**

Acknowledgments

Officially, this is a one man project.
Honestly, this is not.

I would like to devote my sincere thank to Dr. Andy Chun, who is my supervisor of this project. Throughout the year, Dr Chun has given me so much encouragement and flexibility on this project as well as the cultivation on my research attitude.

Secondly, I would like to give my special thanks to my internship supervisor, Mr. Chris Lai, who spotted the subject of this project. Chris had coached me during the internship, the professional practice in project planning, project management, and project documentation. These are all essential skills for this project as well as for all coming projects.

Last but not least, I appreciate very much the effort from people all over the world to make some very nice documentation about their projects and products, but more importantly, their will to share knowledge on the World Wide Web for free. These people include Jena development team from HP Labs, W3C and OilEd development team from University of Manchester.

I thank all of you who have helped me in doing this project.

Abstract

The World Wide Web is probably the richest source of information now. But we have not utilized this source of information to its full potential since the current web is designed for human consumption; machine processing on web documents is not possible (or at least very difficult).

Semantic Web, a state-of-the-art concept of web architecture proposed by W3C, aims to change the current web into a machine processable web.

As a research project report on Semantic Web, this report includes discussion on what, why, when, how questions about Semantic Web. By building a prototype of the Semantic Web, the project evaluates this new concept and its related technologies in a concrete and in-depth manner. The prototype does not only serve as a proof for the feasibility, applicability and new possibility of Semantic Web, it is also one of the very few Semantic Web applications at this moment which can serve as an example for future Semantic Web application development.

Table of Content

1. Project Title

Semantic Web

2. Introduction

2.1 The Problem

Nowadays World Wide Web is widely used in every part of the world to facilitate information exchange. Yet it is a success only in providing information for human manipulation, machine reasoning and processing on web page information can not provide precise and meaningful results. According to Tim Berners-Lee, the inventor of the World Wide Web, the web was designed as an information space, with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help. One of the major obstacles to this has been the fact that most information on the web is designed for human consumption, and even if it was derived from a database with well defined meanings (in at least some terms) for its columns, that the structure of the data is not evident to a robot browsing the web. Leaving aside the artificial intelligence problem of training machines to behave like people, the Semantic Web approach instead develops languages for expressing information in a machine processable form.[1]

2.2 The Solution

Semantic Web is the next generation of World Wide Web. The significance of the Semantic Web is that information on the web will be machine-understandable, so that machine reasoning on web page information is possible. The Semantic Web is a web of data, in some ways like a global database[2]. Semantic Web organizes information in a well defined structure and creates an environment where software agent can readily carry out sophisticated tasks on the web for users.

[1] http://www.w3.org/DesignIssues/Semantic.html
[2] ibid

According to Tim Berners-Lee, some of the design goals of the Semantic Web are [3]:

- To bring structure to the meaningful content of web pages so that logic can be added to the web and make inference in between the web content possible.
- To extend the current World Wide Web to a machine processable form, not to make a separate one.
- To make it as decentralized as possible. The design has to compromise to throw away the ideal total consistency of all its interconnections but allowing versatility and unchecked exponential growth.

2.3 The Chronology

The history of Semantic Web can date back to 1998 when Tim Berners-Lee first wrote the Road map for the Semantic Web. Research group of the World Wide Web Consortium(W3C) started to draft specifications and standards of relevant technologies to support this initiative.

The Semantic Web activities grow gradually until 2001, when Tim-Berners-Lee and his colleagues actively worked out publications, presentations and conferences which promoted more discussions and researches in organizations and universities all over the world.

Today, Semantic Web is still a state of the art technology which is one of the hot topics in distributed knowledge base system development. Yet this technology has not come to the stage that can be widely adopted in the community. More researches and developments have to be done to make it mature before transformation from the current World Wide Web to a World Wide Semantic Web can be done.

[3] http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

3. Project objectives

By doing this project, the followings are meant to achieve:

- Get familiar with the Semantic Web concept and its significance.

- Learn the Semantic Web related technologies such as eXtensible Markup Language (XML), Resource Description Framework (RDF), RDF Schema and the DARPA Agent Markup Language plus Ontology Inference Layer (DAML+OIL).

- Realize and evaluate the Semantic Web by implementing a small Semantic Web for a selected domain and a Java software agent to manipulate the Semantic Web content.

- Study the evolution and new challenges of web as well as the pros and cons of Semantic Web and its related technologies.

4. Scope of project work

The scope of work will include the followings:

1. Study publications and articles about the concept of Semantic Web.
2. Study the language specifications of XML, RDF, RDF Schema and DAML+OIL.
3. Study the ontology design technique
4. Design a ontology for a selected domain using a ontology editor
5. Create RDF documents based on the ontology
6. Design a web application program to analyze the Semantic Web documents
7. Implement the application in Java with Java RDF APIs
8. Carry out experiments on the prototype to evaluate the Semantic Web
9. Evaluate web technologies of different ages
10. Find out the up coming challenges in web development

5. Background research

The Semantic Web Architecture[4]:

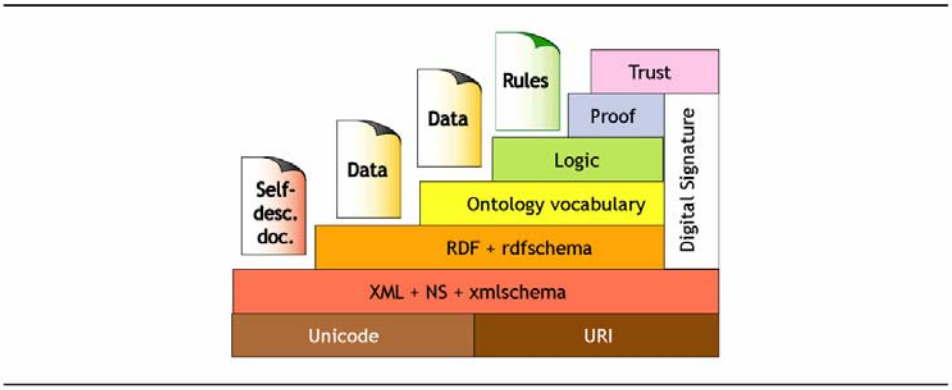


Fig.5.1 Layered architecture of the Semantic Web

As the diagram shown in Fig.5.1, the Semantic Web is built based on a layered architecture. Each layer is built to provide better capabilities to represent knowledge. The development of a layer is possible only with the capabilities supported by the lower layers. As you may notice that this diagram does not only show the architecture, but also it shows the chronological order of the development. The lowest layer, Unicode and URI, are something well developed for long. XML, XML schema and namespace are widely adopted recently in web development. RDF and RDF schema are something that have just beome recommendations of the W3C. For ontology, languages are being finalized, yet it is not yet a standard. For the top three layers, there is so little to tell other than their concepts. Up to this moment, these 3 layers are something to be done in the short future because of their dependency on the lower layers: full scale development of logical layer will proceed only if the ontology is finalized and so on.

Therefore, this project will mainly cover the details about the bottom 4 layers. Some brief concepts of each layer are shown in the following sections.

[4] http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

5.1 Unicode

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. This unique numbering avoids data corruption when data is passed between different encodings or platforms. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others. Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646. It is supported in many operating systems and all modern browsers.[5] This unique character identification serves as a syntactic support to the upper layers in the Semantic Web architecture.

5.2 Uniform Resource Identifiers (URI)

URI is a globally adopted internet addressing technology. URIs are short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources. They make resources available under a variety of naming schemes and access methods such as HTTP, FTP, and Internet mail addressable in the same simple way.[6] URI serves as a unique resources addressing scheme in the Semantic Web architecture, as well as the current World Wide Web architecture.

5.3 eXtensible Markup Language (XML)

XML is a simple, very flexible markup language derived from SGML for documents containing structured information. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents. Unlike well defined structure and semantic in HTML, XML supports arbitrary structure. XML specifies neither semantics nor a tag set. In fact

[5] http://www.unicode.org/unicode/standard/WhatIsUnicode.html
[6] http://www.w3.org/Addressing/

XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by stylesheets.[7] XML serves as a layer for defining arbitrary structures in Semantic Web document without saying anything about the semantic of that structure. XML syntax is used for all languages on the upper layers in the Semantic Web architecture.

5.4 XML schema

XML schema is a better alternative to XML Document Type Definition (DTD) that specifies the structure and data type of elements and attributes in its instance documents. A XML schema can be used to create XML document with certain restrictions or it can be used to validate a XML document to see whether it conforms to those restrictions specified in the schema.

5.5 XML namespace

XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.[8] This is vital to avoid naming conflicts of elements and attributes in XML document with more than one schema. Note that the namespace URI reference does not necessarily point to anything. It could be a “dead link” but it still serves as a valid namespace URI reference.

[7] http://www.xml.com/pub/a/98/10/guide1.html#AEN58
[8] http://www.w3.org/TR/1999/REC-xml-names-19990114/

5.6 Resource Description Framework (RDF)

RDF is a foundation for processing metadata (metadata is "data about data", for example, a library catalog is metadata, since it describes publications); it provides interoperability between applications that exchange machine-understandable information on the web. RDF emphasizes facilities to enable automated processing of web resources.

The syntax presented in RDF is the XML syntax. One of the goals of RDF is to make it possible to specify semantics for data based on XML in a standardized, interoperable manner. RDF and XML are complementary: RDF is a model of metadata. For many of the syntax and encoding issues such as internationalization, character sets, etc., RDF relies on the support of XML.

The foundation of RDF is a model for representing named properties and property values. The RDF data model is a syntax-neutral way of representing RDF expressions. The data model representation is used to evaluate equivalence in meaning. Two RDF expressions are equivalent if and only if their data model representations are the same. This definition of equivalence permits some syntactic variation in expression without altering the meaning. The basic data model consists of three object types: Resources, properties and statements.

Consider as a simple example the sentence:

"Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila."

This sentence has the following parts which form a RDF model statement:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

A RDF statement can be represented pictorially using directed labeled graphs (also called "nodes and arcs diagrams"). In these diagrams, the nodes (drawn as ovals) represent resources and arcs represent named properties. Nodes that represent string literals will be drawn as rectangles. The sentence above would thus be diagrammed as:

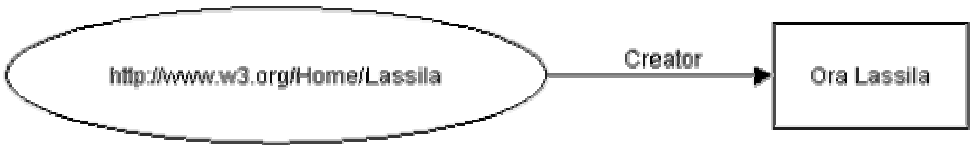


Fig.5.2 Simple node and arc diagram

Note that the direction of the arrow is important. The arc always starts at the subject and points to the object of the statement. The simple diagram above may also be read "*http://www.w3.org/Home/Lassila has creator Ora Lassila*", or in general "*<subject> HAS <predicate> <object>*". [9]

RDF can be considered as the beginning of the story of Semantic Web since it is the first specification to allow unambiguous semantic of statement assertion to be carried in web documents which is not possible using pure XML. This is vital as a base for logic to be imposed to web documents on the upper layers in the Semantic Web architecture.

5.7 RDF schema

Similar to the relationship between XML and XML schema, RDF schema specifies and imposes restriction to resources and properties in RDF model. For example, RDF schema can be used to specify superclass-subclass relationship of resource and to define domain and range of a property restricted to certain classes.

This specification of meaningful use of properties and classes in RDF data is again essential for logic and deduction to be made in the Semantic Web.

[9] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

5.8 Ontology

On the web, the same concept of knowledge could be represented in different ways, in different document structure using different identifiers. A program that wants to compare and analyze these documents must have a way to discover such common meanings.

A solution to this problem is provided by the third basic component of the Semantic Web, collections of information called ontologies. Ontologies describe formal, shared conceptualizations of a particular domain of interest. This description can be used to describe structurally heterogeneous and distributed information sources as found on the web. By defining shared and common domain concepts and vocabularies, ontologies help both people and machines to communicate concisely, supporting the exchange of semantics and not only syntax.

The basic building block for ontologies are concepts, which are typically hierarchically organized in a concept hierarchy. These concepts can have properties which establish named relations to other concepts.[10]

5.9 OIL

During the evolution of web, there is a evolution of web ontology language. The first form of ontology language is called the Ontology Inference Layer (OIL).

OIL is a language built on a long history (over 20 years) of research in description logics. Description Logic is a subfield of knowledge representation and as such aims to provide a vehicle for expressing structured information and for reasoning with the information in a principled manner. Description logics may be viewed as providing a formal foundation for frame-based systems, object-oriented representations, semantic data models, and type systems. OIL is an effort to produce a well defined language for integrating ontologies with web standards (in particular RDF/RDFS and XML/XMLS). It is a web-based representation and inference layer for ontologies

[10] http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/dexa2002.pdf

using the constructs found in many frame languages and reasoning and formal semantics in description logics.

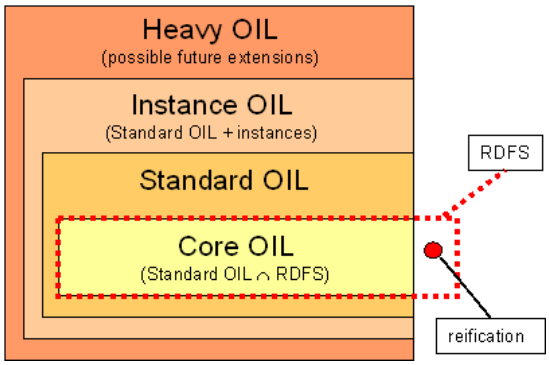


Fig.5.3 Layers of OIL

OIL presents a layered approach to a standard ontology language. Each additional layer adds functionality and complexity to the previous layer. This is done such that agents (humans or machines) who can only process a lower layer can still partially understand ontologies that are expressed in any of the higher layers. The picture above sketches the relation between the OIL dialects and RDFS.

Core OIL coincides largely with RDF Schema (with the exception of the reification features of RDF Schema). This means that even simple RDF Schema agents are able to process the OIL ontologies, and pick up as much of their meaning as possible with their limited capabilities.

Standard OIL is a language intended to capture the necessary mainstream modelling primitives that both provide adequate expressive power and are well understood thereby allowing the semantics to be precisely specified and complete inference to be viable.

Instance OIL includes a thorough individual integration. While the previous layer - Standard OIL - included modelling constructs that allow individual fillers to be specified in term definitions, Instance OIL includes a full-fledged database capability. (note that Instance-OIL has the same schema as Standard-OIL; the instances are described directly in RDF)

Heavy OIL is a future extension of OIL that includes additional representational (and reasoning) capabilities. [11]

5.10 DAML+OIL

DARPA Agent Markup Language (DAML) is a $80M program for Semantic Web by The Defense Advanced Research Projects Agency (DARPA) of United State. DAML is an ontology and inference language based upon RDF. It takes RDF Schema a step further, by giving us more in depth properties and classes. DAML allows one to be even more expressive than with RDF Schema, and brings us back on track with our Semantic Web discussion by providing some simple terms for creating inferences.

DAML inherits many aspects from OIL, and the capabilities of the two languages are relatively similar. In December 2000, DAML and OIL are merged to form a new version as to make DAML more consistent with the European OIL project. The DAML+OIL keeps the OIL's mapping to Description Logic but moves away from the frame-style language of OIL. General axioms are used instead of frame descriptions. DAML+OIL becomes an alternative syntax for Description Logic.[12]

More recently, the WebOntology group of the W3C is developing a universal standard based on DAML+OIL, which is the Web Ontology Language (OWL). OWL is expected to be a W3C recommendation once it reaches a mature stage.

[11] http://www.daml.org/2000/10/daml-oil
[12] http://www.ifs.tuwien.ac.at/ifs/lehre/skripten/metadaten/metadaten_ifs.ppt

As a summary, here it is a language comparison table of common languages used in different layers of the Semantic Web architecture quote from www.daml.org.[13]

	XML DTD	XML Schema	RDF(S)	DAML+OIL	RDF(S) 2002	OWL
bounded lists				X	X	X
cardinality constraints	X	X		X		X
class expressions				X		X
data types		X		X	X	X
defined classes				X		X
enumerations	X	X		X		X
equivalence				X		X
extensibility			X	X	X	X
formal semantics				X	X	X
inheritance			X	X	X	X
inference				X		X
local restrictions				X		X
qualified constraints				X		
reification			X	X	X	X

- bounded lists
  - `rdf:Seq` and `rdf:Bag` don't provide any indication that the list is complete (e.g. "these are 5 children of X" vs. "X is known to have exactly these 5 children")
  - DAML uses a first/rest structure to represent unordered bounded lists, with `nil` representing the end of the list
  - `rdf:parseType="daml:collection"` provides a shorthand and RDF syntax compatibility

[13] http://www.daml.org/langua ge/features.html

- cardinality constraints
  - limit the number of statements with the same subject and predicate
  - the Kleene operators ? (0 or 1), * (0+), and + (1+) in XML DTD provide basic cardinality constraints
  - DAML supports `cardinality`, `minCardinality`, and `maxCardinality`
- class expressions
  - wherever a Class is referenced, DAML+OIL also allows an expression involving `unionOf`, `disjointUnionOf`, `intersectionOf`, or `complementOf`
- data types
  - RDF Literals are essentially strings
  - DAML+OIL (March 2001) adds XML Schema data types
- defined classes
  - DAML allows new classes to be defined based on property values or other restrictions of an existing class (e.g. Child is a Person with age < 18) or class expressions.
- enumerations
  - XML DTD allows specification of a restricted set of values for a given attribute
  - DAML provides `oneOf`
- equivalence
  - to support reasoning across ontologies and knowledge bases, DAML supports `equivalentTo` for classes, properties, and instances
  - additional mapping constructs may be added to future versions of DAML
- extensibility
  - RDF and DAML allow new Properties to used with existing Classes
  - RDF has been used to define DAML+OIL
  - DAML+OIL may similarly be used to define future languages such as DAML-Logic
- formal semantics
  - DAML+OIL semantics have been expressed in both model-theoretic and axiomatic forms

- inheritance
  - XML Schema attribute groups formalize DTD use of parameter entities in attribute definitions, but this isn't full inheritance
  - RDF and DAML support `subClassOf` and `subPropertyOf`
- inference
  - DAML+OIL constructs such as `TransitiveProperty`, `UnambiguousProperty`, `inverseOf`, and `disjointWith` provide additional information for reasoning engines
  - future versions of DAML are expected to support rules, proof-checking, etc.
- local restrictions
  - RDF associates `domain` and `range` constraints with a Property
  - DAML allows `Restriction`s to be associated with a Class/Property pair, e.g. allowing the `color` property to be used for the `Car` and `Eye` classes with different domains
- qualified constraints
  - DAML restrictions allow expressions such as "all children of X are of type Person"
  - the DAML properties `hasClassQ`, `cardinalityQ`, `minCardinalityQ`, and `maxCardinalityQ` allow qualified restrictions such as "at most 3 of the children of X are of type Doctor"
- reification
  - RDF and DAML allow a statement to be the subject of another statement
  - reification provides a standard mechanism for recording data sources, timestamps, etc. without intruding on the data model
  - the DAML+OIL semantics do not currently cover reification
  - in its initial discussions on reification, the Joint Committee has found it useful to distinguish "tagging" (making statements about asserted statements, e.g. source, timestamp, etc.) from "quoting" (making statements about unasserted statements)

5.11 The ultimate three layers

The logic layer consists of rules that enable inferences, e.g. to choose courses of action and answer questions.

The proof layer has been conceived to allow the explanation of given answers generated by automated agents. Naturally, you might want to check the results deduced by your agent, this will require the translation of its internal reasoning mechanisms into some unifying proof representation language.[14]

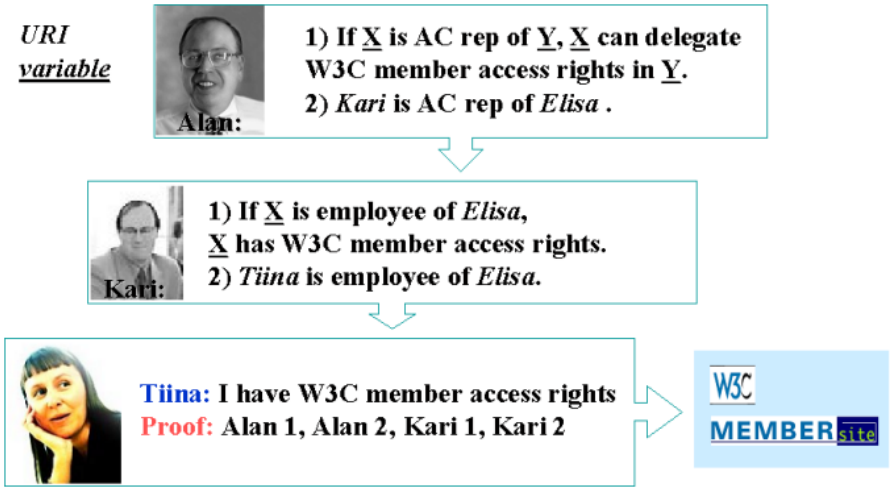


Fig.5.4 Logic and Proof layer illustration from W3C

The trust layer representing the "Web of Trust" is the ultimate goal of Semantic Web. Application needs a mechanism to evaluate the trustworthiness of statements found on the Semantic Web. The basic idea to do this is by digital signature to sign the RDF assertions from a party the application trusts.

These three layers give some hints to us on what the web will finally be. However, this project is dedicated to study how Semantic Web can work out in the context of lowest four layers of the Semantic Web. A workable prototype of a microscopic Semantic Web will be discussed in the coming chapters.

[14] http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/dexa2002.pdf

6. A prototype of Semantic Web – "My First Uni"

To show the Semantic Web working in a realistic scenario, I have selected a specific study case as an example to demonstrate how web information processing today can be automated in future with Semantic Web.

"My First Uni": finding out the most suitable bachelor degree program for Hong Kong high school graduates

For every high school graduate who wants to study in a university, they have to select a list of bachelor degree programs, sorted in priorities, and hand in the list to the *Joint University Programmes Admissions System* (JUPAS).

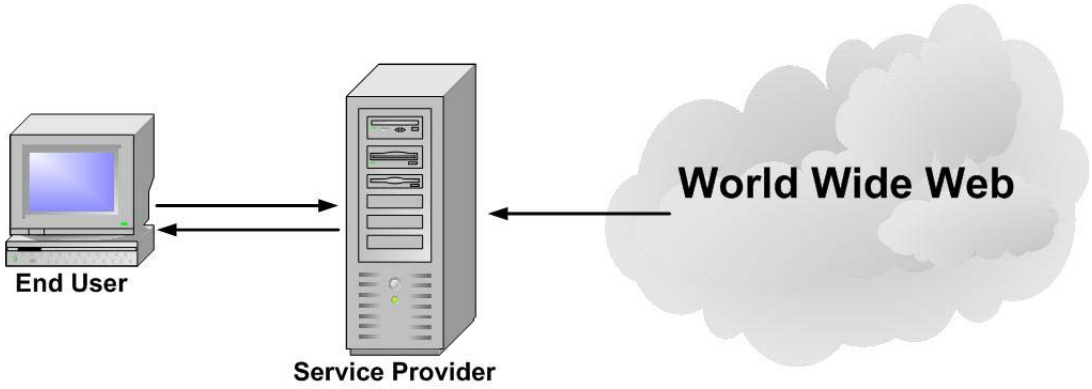
To select bachelor degree programs, students have to consider:

- **The admission requirement of the university**
  For admission in most universities, applicant has to satisfy certain academic requirements, such as qualification in language skills.
- **The admission requirement of the bachelor degree program**
  For admission in some bachelor degree programs, applicant has to satisfy certain academic requirements, such as qualification in related subjects of study.
- **The ranking of the university**
  Students are interested in the reputation/ranking of universities as they prefer to apply for a school with better reputation.
- **Degree of interest in different fields of study**
  Students prefer bachelor degree programs which suit their interest, those they do not interest in are less preferred.

To do this selection manually, students have to search and refer to web pages of every university, every department of that university, every bachelor degree program of that department, as well as the web page of current university ranking. This task is time-

consuming for students which may take days to read through a hundred of web pages before they can come up with a list.

A web application is needed automate this manual process.



Traditional approach I: using database

Using current technology, a way to do it is to have all information collected and consolidated by a third party, the service provider, and then input all the information into a database. After this database is set up, this party can write a web application for this purpose using the information from the database.

The problem is the manual process between the service provider and the information provider. Information has to be processed manually by the service provider when preparing the database. This is not acceptable when the whole thing scale up to a size, say to cater for hundred thousands of different bachelor programs around the world. Furthermore, this is not a one-off process, the database has to be maintained and updated for all information changes which may happen everyday or the service will lose synchronization to up-to-date information.

On the other hand, control of information is absolutely centralized. Once the database is set up, control is done only by the database owner. The information provider is in no position to control the database information for which it has the ownership.

More importantly, the information in the database is designed for this particular purpose only. This information is not accessible, and the semantics is unknown to other programs provided by other parties for other purposes. This is absolutely not the way to make a web of inter-processable information.

Traditional approach II: using XML

An alternative is to use XML. If all information providers provide the information in XML form, the application program can process the information directly from web rather than retrieving information from database. In this way, control can be distributed back to the information provider and the access of information is opened to everyone.

But there are many limitations. To make this possible all the XML document and the application program have to agree to use a common XML structure and tags (hence a common XML schema). This is not practical as it is difficult to enforce all information providers to use the same schema and that it is difficult to define a schema that suits different variations of information from different universities.

Another problem is the discoverability of these XML documents on the web. Say if there is a new university established or some existing XML documents change their URIs, the application program will not be able to discover the location of these XML documents for use. It is difficult for today's search engine to discover these specific XML documents on the web.

In addition, the semantics of the XML documents is not defined within themselves. Applications could interpret the same XML document with different semantics. (and probably most applications will misinterpret the original semantics) That's why XML is not an answer for inter-processable information on the web.

Semantic Web approach : RDF & DAML+OIL

Now we are looking for a new solution that can satisfy the following requirement:

- All information on the web is machine-processable to allow web applications to answer a complicated question by manipulating multiple sources of information from the web and consolidate them to generate a correct answer.
- Control and maintenance of information is distributed solely to the information owners.
- Access of information is opened to public.
- The form of document storing machine-processable information has to allow great flexibility in its structure.
- These documents have to be discoverable and classifiable on the dynamic and opened web.
- All information has to have well defined semantics resided within itself so that it is inter-processable by any machine process.

Building on top of XML, RDF inherits all the benefits of XML but also incorporated clear semantics in the document. Instead of a representation of a syntactic structure, RDF is a syntax-neutral way to represent a model which can be thought as a directed labeled graph model. Every node and arc in the model must have a URI label, which is as critical as the hyperlink of HTML. With the URI label, RDF documents can be linked with other web documents so that web crawlers can traverse through the web to discover these RDF documents. Thus the URI label greatly enhanced the discoverability and the cohesion of RDF documents.

Since RDF is about describing something about something, or describing anything about anything, there is a great flexibility in composing a RDF comparing with XML. XML document structure is bounded by XML schema, the whole tree structure and all its tag names has to be consistent with the XML schema. But in RDF, the document structure is not bounded; author can use whatever vocabulary in the RDF statements. A RDF document is valid as long as all the RDF statements are complete.

With DAML+OIL semantics incorporated, RDF can extend its expressive power of semantics. DAML+OIL gives formal definition to vocabularies used in the RDF documents. Since the DAML+OIL itself is also a RDF document, it is discoverable and accessible by public. This allows any web applications to interpret any RDF document with correct semantics defined in the corresponding DAML+OIL documents. These formal semantics definitions are very useful that enable intelligent agents to make decisions, such as inference and prediction.

In the following chapters, you will see how a semantic web prototype is built for this study case.

7. Prototype design I: Semantic Web content

The Semantic Web prototype in this project consists of two parts, the Semantic Web content and the Semantic Web application.

This chapter discusses how we design web content for the Semantic Web. At the beginning we have to come up with a concept definition called "ontology" as a base model of knowledge of a given domain/context. Ontology is an explicit specification of a conceptualization. That is, ontology is a description (like a formal specification of a program) of the concepts and relationships among these concepts. Similar to class-object relationship in Object Oriented Programming theory, there is a class-instance relationship in ontology: a class in the ontology is a template of concept and instances can be instantiated from the template. On the representation level, the ontology is represented in DAML+OIL, whereas the instances are represented in RDF.

7.1 Ontology design strategy

The ontology is developed following the design strategy from "Ontology Development 101: A Guide to Creating Your First Ontology"[15] which involves the following steps:

Step 1. Determine the domain and scope of the ontology

To define the scope of the ontology by considering the domain it covers, the purpose of the ontology and what information that the ontology should carries for the knowledge base to give answers.

Step 2. Consider reusing the existing ontologies

To consider what someone else has done and checking if it is possible to refine and extend existing sources for one's particular domain and task. Reusing existing ontologies may be a requirement if the system needs to interact with other

[15] http://protege.stanford.edu/publications/ontology_development/ontology101.html

applications that have already committed to particular ontologies or controlled vocabularies.

Step 3. Enumerate important terms in the ontology

To write down a list of all terms which have to make statements about or to explain to a user. In this step it is important to get a comprehensive list of terms without worrying about overlap between concepts they represent, relations among the terms, or any properties that the concepts may have, or whether the concepts are classes or properties.

Step 4. Define the classes and the class hierarchy

There are several ways to develop a class hierarchy: top-down approach, bottom-up approach or the combination of both. Whichever approach is chosen, we usually start by defining classes. From the list created in Step 3, we select the terms that describe objects having independent existence rather than terms that describe these objects. These terms will be classes in the ontology and will become anchors in the class hierarchy. We organize the classes into a hierarchical taxonomy by asking if by being an instance of one class, the object will necessarily (i.e., by definition) be an instance of some other class.

Step 5. Define the properties of classes

The classes alone will not provide enough information. Once we have defined some of the classes, we must describe the internal structure of concepts.

After selecting classes from the list of terms created in Step 3, most of the remaining terms are likely to be properties of these classes.

For each property in the list, we must determine which class it describes. These properties become properties attached to classes.

Step 6. Define the facets of the properties

Properties can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the property can take. In this step we need to define the cardinality (e.g. 1..*), value type (e.g. integer), domain (the subject) and range (the object) of each property.

Step 7. Create instances

To define an individual instance of a class which requires (1) choosing a class, (2) creating an individual instance of that class, and (3) filling in the property values.

The following sections will show the details of work of the ontology "MyFirstUni", corresponding to the steps above.

7.2 Determine the domain and scope of the ontology

In this prototype, the ontology "MyFirstUni" is specific to the concept of university admission, or more specifically, university bachelor degree program admission for Hong Kong secondary school graduates. The ontology should able to give information about which Bachelor programs a student is eligible to admit to, and then sort them according to the university ranking. To trim down the scope to a manageable size for this one man project, most of the supplementary concepts are omitted. Only those essential to the admission eligibility are included in the ontology. Since the scope is defined to include only HKCEE and HKALE examination record as admission criteria, other criteria such as international examination record and extra curricular activities of students are not considered in this ontology.

7.3 Consider reusing the existing ontologies

In terms of consistency and reusability, ontology should be reused if there is an existing one that fits the purpose. However, after searching on Internet and the DAML+OIL ontology library, there is no such ontology for this purpose exists. Not even a general ontology for university admission. That's why it is necessary to build a new ontology from scratch for this specific domain.

Although there is no existing ontology that fits the purpose, there are some ontologies which are about university research and publication. I have chosen one of these ontologies to extend and reuse some of the classes that are relevant to this project domain.

7.4 Enumerate important terms in the ontology

Here it is an unordered list of terms that comes up during the first draft of the ontology:

University	Department	Bachelor degree	Student	Hong Kong High school graduate
HKCEE	HKALE	Email address	Ranking	Ranking type
HKCEE subjects	HKALE subjects	HKCEE grade	HKALE grade	Entry requirement

7.5 Define the classes and the class hierarchy

Fig. 7.1 Class Hierarchy of ontology "MyFirstUni"

top	Denoted the root of the ontology
C	Denoted a class of information in the domain
C #1	Denoted a class defined in ontology "MyFirstUni"
C #2	Denoted a class defined in ontology "ksl-daml-desc" from Stanford University

For classes marked "#1", they are classes defined in this new ontology "MyFirstUni". For classes marked "#2", they are Super classes being inherited in "MyFirstUni". These classes are not originally defined in "MyFirstUni", but in http://www.ksl.stanford.edu/projects/DAML/ksl-daml-desc.daml, which is an existing ontology modeling university publication. In "MyFirstUni" these super classes are added with new properties. (See Fig. 7.2 for details) It is important to link with the existing ontology wherever possible in order to extend the existing knowledge base rather than having these ontologies isolated from each other.

This class hierarchy shows the superclass-subclass relationship. For instance, "Student" is a subclass of "Person" and "Student" is also a super class of "Hong-Kong-High-School-Graduate".

7.6 Define the properties of classes and its facet

Similar to the Object-Oriented concept, each of the classes has its own properties. Yet there is no "method" since we are composing the N-Triple/RDF Triple, the Subject–Property- Property Value relationship. (or the Subject-predicate-object relationship)

Same as the Object-Oriented concept, inheritance is possible in an ontology. The class diagram of the inherited class and the class relationship below is presented using UML notation for better understanding.

Inherited properties

Under this class hierarchy, there are some properties in super class inherited to its sub classes.

For instance, "Hong-Kong-High-School-Graduate" has ancestor class "Person" so that it has inherited all properties from "Person" as shown in the "Person" box in Fig.7.2. In addition to inherited properties, subclasses like "Hong-Kong-High-School-Graduate" may also have their own properties defined.

For design reason, super class like "Exam-Record(single)" has no properties. It serves as a generalization of all exams.

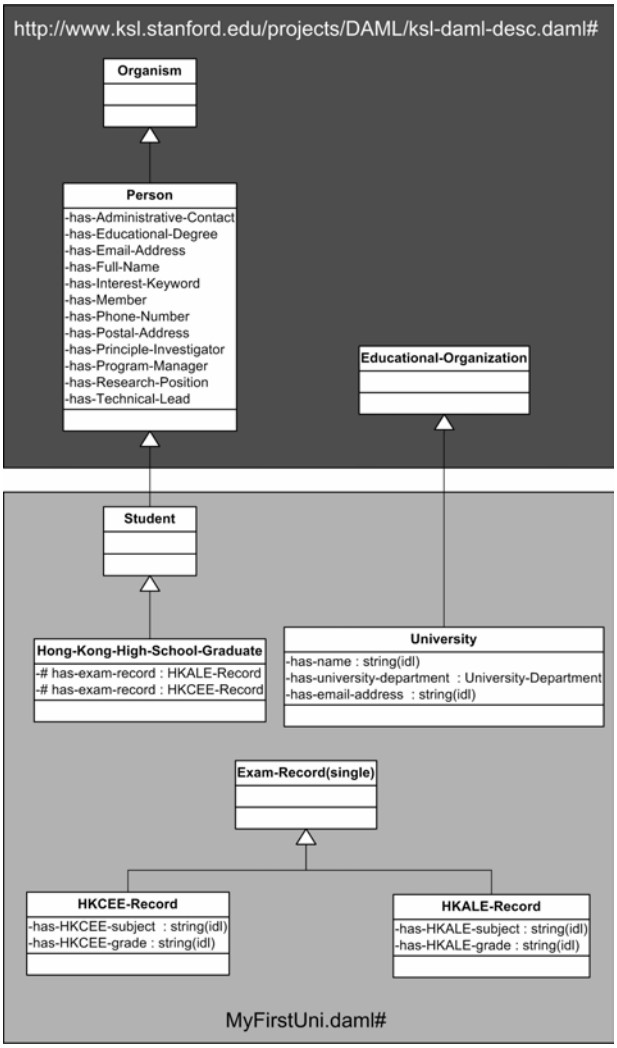


Fig.7.2 Class generalization

7.7 Properties and Class relationships

Here it is the Class relationship diagram:

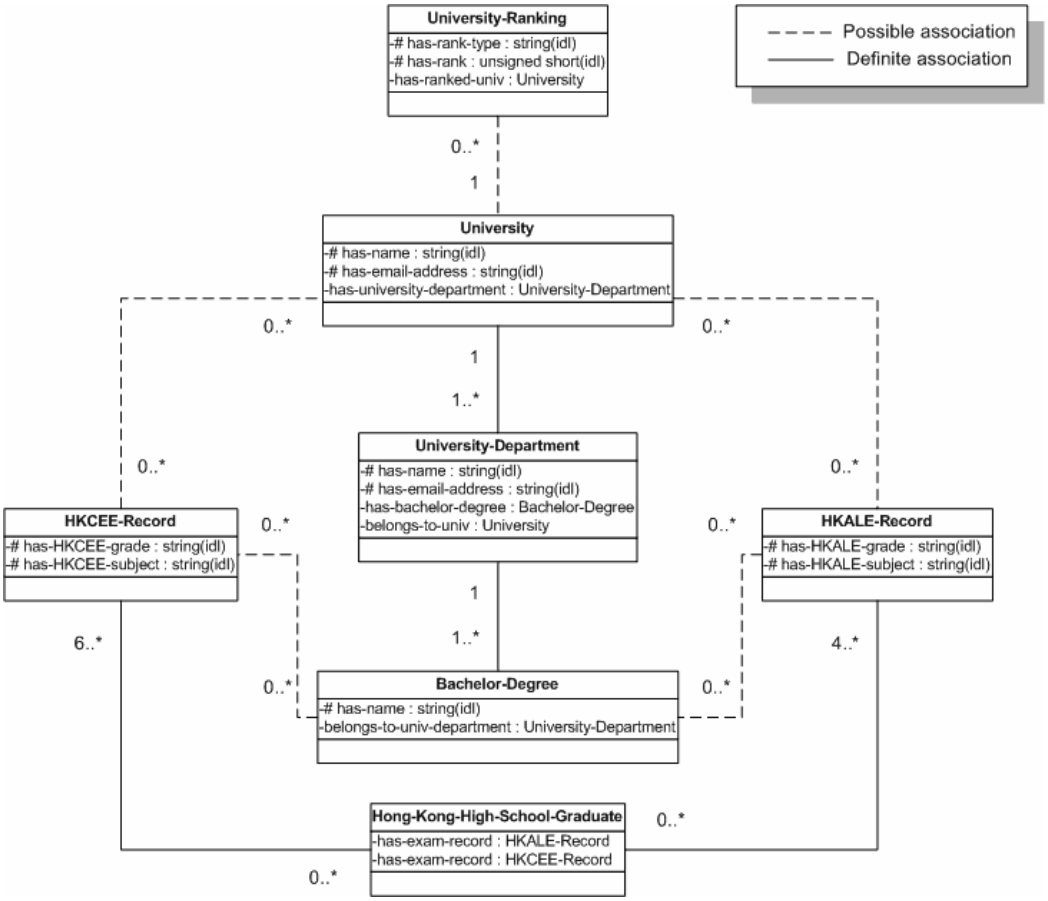


Fig.7.3 Class relationship diagram

In Fig.7.3, inherited properties from super class are hidden. Only class restrictions (necessary properties) are shown in the class boxes. Optional properties are not shown in the diagram. Properties marked with "#" have cardinality "exactly 1" to the class. Other properties (which relate to other classes) have cardinality as shown at the ends of the linkage lines.

Definite Associations:

University-Ranking to University relationship (unidirectional):

Domain	*Property*	*Range*
Class "University-Ranking"	has-ranked-univ	Class "University"

A "University-Ranking" describes the ranking of a "University" in certain area (i.e. the rank-type). A ranking describes one single "University". Yet a "University" may have "University-Ranking", up to many "University-Ranking" as a "University" may have rankings in different areas or ranking by different parties.

University to University-Department relationship (bidirectional):

Domain	*Property*	*Range*
Class "University"	has-university-department	Class "University-Department"
Class "University-Department"	belongs-to-univ	Class "University"

A "University" is defined to have at least one and up to many "University-Department". Each "University-Department" must belong to one single "University" only.

University Department to Bachelor-Degree relationship (bidirectional):

Domain	*Property*	*Range*
Class "University-Department"	has-bachelor-degree	Class "Bachelor-Degree"
Class "Bachelor-Degree"	belongs-to-univ-department	Class "University-Department"

A "University-Department" may offer at least one and up to many "Bachelor-Degree" programs. Each "Bachelor-Degree" must belong to one single "University-Department" only.

Hong-Kong-High-School-Graduate to HKALE-Record relationship (unidirectional):

Domain	*Property*	*Range*
Class "Hong-Kong-High-School-Graduate"	has-exam-record	Class "HKALE-Record"

A "Hong-Kong-High-School-Graduate" is defined as a student who has taken at least 4 subject examinations in HKALE, and thus has at least 4 "HKALE-Record". It is the requirement of some universities in Hong Kong that their local applicant should have taken at least 4 subject examinations in HKALE.

Hong-Kong-High-School-Graduate to HKCEE-Record relationship (unidirectional):

Domain	*Property*	*Range*
Class "Hong-Kong-High-School-Graduate"	has-exam-record	Class "HKCEE-Record"

A "Hong-Kong-High-School-Graduate" is also defined to have taken at least 6 subject examinations in HKCEE, and thus has at least 6 "HKCEE-Record". It is very common that a local university applicant to have taken 6 to 10 subjects in their HKCEE.

Possible Associations:

University to HKALE-Record / HKCEE-Record relationship (unidirectional) :

Domain	*Property*	*Range*
Class "University"	has-entry-requirement	Class "Exam-Record(single)"

It is common that universities in Hong Kong require their local applicants to have certain grade in some of the subjects in their HKALE & HKCEE record. That's why "University" may relate to zero or more "Exam-Record(single)" (which could be a "HKCEE-Record" or "HKALE-record") as the university's admission requirement. It is an optional relation here as to leave some flexibility to the admission requirement.

Bachelor-Degree to HKALE-Record relationship

Domain	*Property*	*Range*
Class "Bachelor-Degree"	has-entry-requirement	Class "Exam-Record(single)"

In addition to the university admission requirements, individual Bachelor degree program may require additional HKALE & HKCEE grades from the applicant who applies for that degree. This is again an optional relation.

7.8 Create instances

After the completion of class and property definitions, the ontology can be used to create instances which are represented in RDF. Here the property values of those instances and the mapping between the instances and the RDF documents are shown in tabular format. Each table is about instances in one RDF document.

ranking.rdf			
Class	*Instance*	*Property*	*Property value*
University-Ranking	RANKING-OVERALL-BU	has-rank	3
		has-web-page	ranking.htm
		has-rank-type	overall
		has-ranked-univ	BU.rdf#BU
	RANKING-OVERALL-CTU	has-rank	1
		has-web-page	ranking.htm
		has-rank-type	overall
		has-ranked-univ	CTU.rdf#CTU
	RANKING-OVERALL-PLU	has-rank	2
		has-web-page	ranking.htm
		has-rank-type	overall
		has-ranked-univ	PLU.rdf#PLU

BU.rdf			
Class	*Instance*	*Property*	*Property value*
University	BU	has-name	Baptess University
		has-email-address	email@bu.edu.hk
		has-web-page	BU.htm
		has-university-department	BU-CS.rdf#BU-CS
		has-university-department	BU-JN.rdf#BU-JN
		has-entry-requirement	BU.rdf#genid001
HKALE-Record	*genid001	has-HKALE-subject	Chinese
		has-HKALE-grade	E

BU-CS.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	BU-CS	has-name	Computer Science Department, Baptess University
		has-web-page	BU-CS.htm
		has-email-address	email@cs.bu.edu.hk
		belongs-to-univ	BU.rdf#BU
		offer-bachelor-degree	BU-CS-CS.rdf#BU-CS-CS

BU-JN.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	BU-JN	has-name	Journalism Department, Baptess University
		has-web-page	BU-JN.htm
		has-email-address	email@jn.bu.edu.hk
		belongs-to-univ	BU.rdf#BU
		offer-bachelor-degree	BU-JN-COMM.rdf#BU-JN-COMM

BU-CS-CS.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	BU-CS-CS	has-name	BSc Computer Science program, Baptess University
		has-web-page	BU-CS-CS.htm
		has-academic-field	Computer Science
		Has-entry-requirement	BU-CS-CS.rdf#genid002
		offer-by-univ-department	BU-CS.rdf#BU-CS
HKCEE-Record	*genid002	has-HKCEE-subject	Mathematics
		has-HKCEE-grade	E

BU-JN-COMM.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	BU-JN-COMM	has-name	BSoc Communication program, Baptess University
		has-web-page	BU-JN-COMM.htm
		has-academic-field	Communication
		Has-entry-requirement	BU-JN-COMM.rdf#genid003
		Has-entry-requirement	BU-JN-COMM.rdf#genid004
		offer-by-univ-department	BU-JN.rdf#BU-JN
HKALE-Record	*genid003	has-HKALE-subject	English
		has-HKALE-grade	D
	*genid004	has-HKCEE-subject	Chinese
		has-HKCEE-grade	C

CTU.rdf			
Class	*Instance*	*Property*	*Property value*
University	CTU	has-name	Cit Tee University
		has-email-address	email@ctu.edu.hk
		has-web-page	CTU.htm
		has-university-department	CTU-CS.rdf#CS
		has-university-department	CTU-EE.rdf#EE
		has-entry-requirement	CTU.rdf#genid005
		has-entry-requirement	CTU.rdf#genid006
HKALE-Record	*genid005	has-HKALE-subject	Chinese
		has-HKALE-grade	E
	*genid006	has-HKALE-subject	English
		has-HKALE-grade	D

CTU-CS.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	CTU-CS	has-name	Computer Science Department, Cit Tee University
		has-web-page	CTU-CS.htm
		has-email-address	email@cs.ctu.edu.hk
		belongs-to-univ	CTU.rdf#CTU
		offer-bachelor-degree	CTU-CS-CS.rdf#CTU-CS-CS

CTU-EE.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	CTU-EE	has-name	Electronic Engineering Department, Cit Tee University
		has-web-page	CTU-EE.htm
		has-email-address	email@ee.ctu.edu.hk
		belongs-to-univ	CTU.rdf#CTU
		offer-bachelor-degree	CTU-EE-EE.rdf#CTU-EE-EE

CTU-CS-CS.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	CTU-CS-CS	has-name	BSc Computer Science program, Cit Tee University
		has-web-page	CTU-CS-CS.htm
		has-academic-field	Computer Science
		Has-entry-requirement	CTU-CS-CS.rdf#genid007
		offer-by-univ-department	CTU-CS.rdf#CTU-CS
HKCEE-Record	*genid007	has-HKCEE-subject	Mathematics
		has-HKCEE-grade	C

CTU-EE-EE.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	CTU-EE-EE	has-name	BEng Electronic Engineering program, Cit Tee University
		has-web-page	CTU-EE-EE.htm
		has-academic-field	Electronic Engineering
		Has-entry-requirement	CTU-EE-EE.rdf#genid008
		offer-by-univ-department	CTU-EE.rdf#CTU-EE
HKALE-Record	*genid008	has-HKALE-subject	Physics
		has-HKALE-grade	E

PLU.rdf			
Class	*Instance*	*Property*	*Property value*
University	PLU	has-name	Pauline University
		has-web-page	PLU.htm
		has-email-address	email@plu.edu.hk
		has-university-department	PLU-CS.rdf#CS
		has-university-department	PLU-EE.rdf#EE
		has-entry-requirement	PLU.rdf#genid009
HKALE-Record	*genid009	has-HKALE-subject	English
		has-HKALE-grade	D

PLU-CS.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	PLU-CS	has-name	Computer Science Department, Pauline University
		has-web-page	PLU-CS.htm
		has-email-address	email@cs.plu.edu.hk
		belongs-to-univ	PLU.rdf#PLU
		offer-bachelor-degree	PLU-CS-CS.rdf#PLU-CS-CS

PLU-EE.rdf			
Class	*Instance*	*Property*	*Property value*
University-Department	PLU-EE	has-name	Electronic Engineering Department, Pauline University
		has-web-page	PLU-EE.htm
		has-email-address	email@ee.plu.edu.hk
		belongs-to-univ	PLU.rdf#PLU
		offer-bachelor-degree	PLU-EE-EE.rdf#PLU-EE-EE

PLU-CS-CS.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	PLU-CS-CS	has-name	BSc Computer Science program, Pauline University
		has-web-page	PLU-CS-CS.htm
		has-academic-field	Computer Science
		Has-entry-requirement	PLU-CS-CS.rdf#genid010
		Has-entry-requirement	PLU-CS-CS.rdf#genid011
		offer-by-univ-department	PLU-CS.rdf#PLU-CS
HKCEE-Record	*genid010	has-HKCEE-subject	Mathematics
		has-HKCEE-grade	C
	*genid011	has-HKCEE-subject	Physics
		has-HKCEE-grade	E

PLU-EE-EE.rdf			
Class	*Instance*	*Property*	*Property value*
Bachelor-Degree	PLU-EE-EE	has-name	BEng Electronic Engineering program, Pauline University
		has-web-page	PLU-EE-EE.htm
		has-academic-field	Electronic Engineering
		Has-entry-requirement	PLU-EE-EE.rdf#genid012
		Has-entry-requirement	PLU-EE-EE.rdf#genid013
		offer-by-univ-department	PLU-EE.rdf#PLU-EE
HKALE-Record	*genid012	has-HKALE-subject	Physics
		has-HKALE-grade	E
HKCEE-Record	*genid013	has-HKCEE-subject	Mathematics
		has-HKCEE-grade	C

** Instances with prefix "genid" are anonymous nodes which they are part of other instances. They are not representing an entity but are part of the description of an entity.*

7.9 Linking the RDF document with HTML document

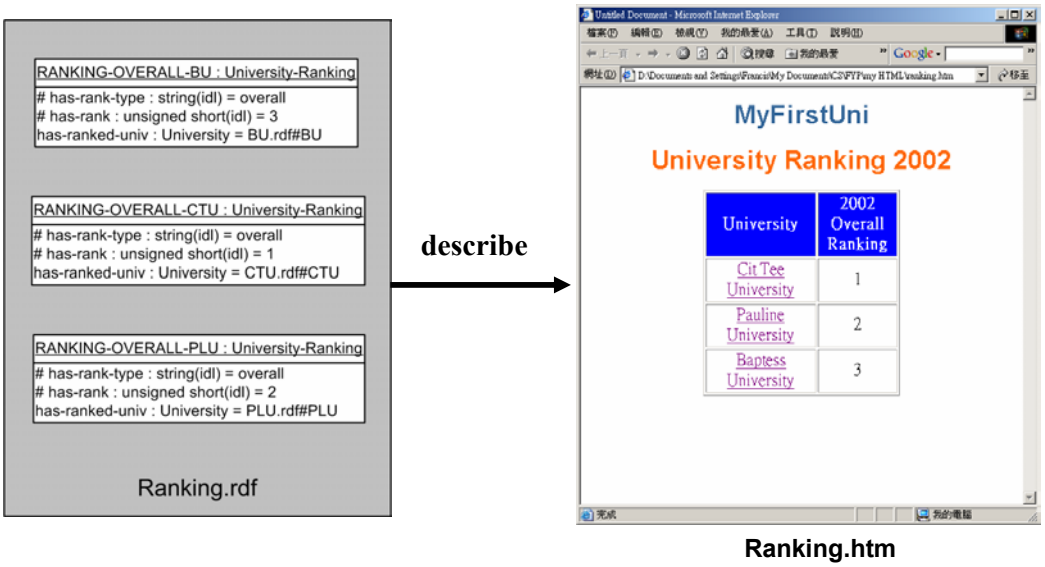


Fig.7.4 RDF & HTML relationship

To link a RDF document to a HTML document, the HTML URI has to be put in the <rdf:Description> tag in the RDF file such that :

```
<rdf:Description rdf:about="http://www.URI_of_Ranking.htm">
</rdf:Description>
```

Alternatively, like all the RDF documents in this project, a property “has-web-page” can be created for this purpose such that:

```
<ns0:has-web-page>
   <xsd:anyURI xsd:value="http://personal.cityu.edu.hk/~50190500/myFirstUni/html/PLU-CS.htm"/>
</ns0:has-web-page>
```

To link a HTML document to a RDF document, the RDF URI has to be put in the <LINK> tag in the HTML file such that :

```
<LINK rel="meta" href="http://www.URI_of_Ranking.rdf">
```

Note that all the URI presented in any RDF document and DAML+OIL documents are not validated by the parser. They do not necessarily point to anything. Yet in my project they all point to some document.

8. Prototype design II: Semantic Web application

In this chapter, the design of a web application for "My First Uni" is described. However, since software development is not a major focus in this project, the application design will not be as detailed as a design specification. Only component level design and major procedures will be described here.

8.2 "My First Uni" components overview

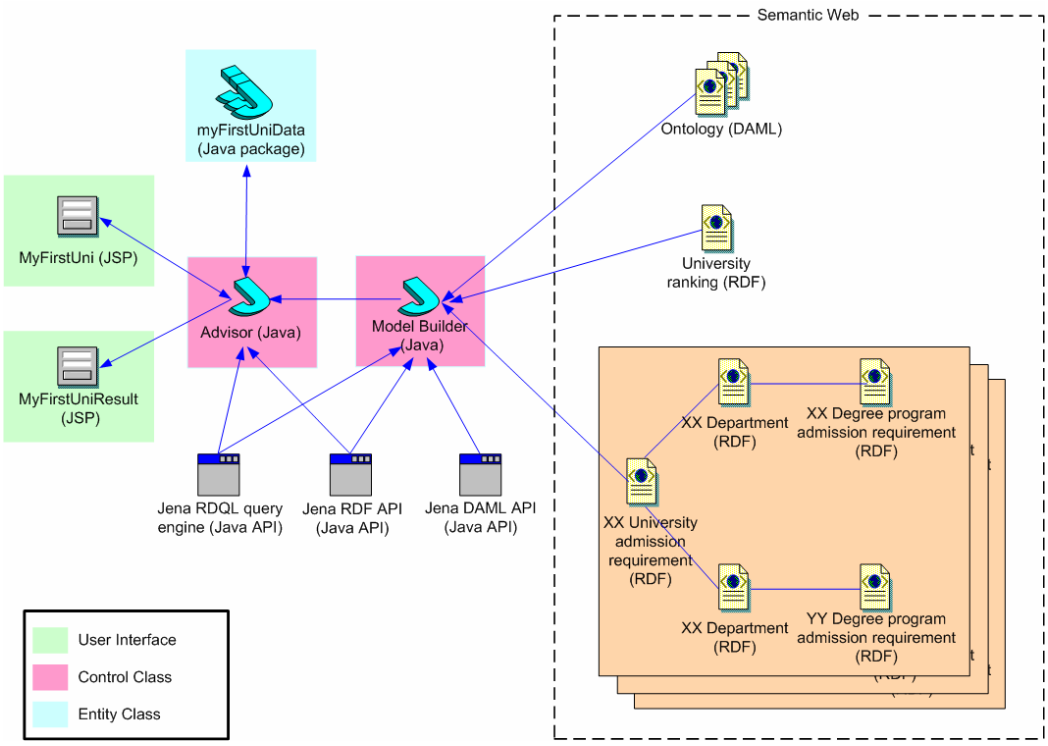


Fig.8.1 Major components overview

This application basically follows a simple UML software design structure which includes user interfaces, control classes and entity classes.

8.2 Components description

User interfaces

Two user interfaces are implemented in JSP for user input and system output respectively.

MyFirstUni.jsp – On this page, user has to input his examination records for the Hong Kong Certificate Examination and the Hong Kong Advanced Level Examination. In addition, user has to specify his preferences among different academic fields (returned from "Advisor") according to his interest. Client-side input validation is done in this JSP. Validated input is passed to the "Advisor".

MyFirstUniResult.jsp – As a computer generated result, a list of recommended Bachelor degree programs and their corresponding information (returned from "Advisor") are sorted from the most recommended program to the least recommended one. The list is displayed in a HTML table.

Control classes

Advisor – It invokes its "Model Builder" to build a RDF model starting from a specified root URI. After the model is ready, it performs various queries to the model for information including the academic fields of the available degree programs (which are passed to MyFirstUni.jsp) and the list of recommended degree programs (which is passed to MyFirstUniResult.jsp). It takes user inputs as well as query results and then stores them using corresponding entity classes.

Model Builder – It is used to retrieve Semantic Web documents (.rdf and .daml) and build a RDF model using retrieved RDF documents. Initially, the Model Builder starts crawling from a root URI given by its "Advisor".

myFirstUniData

A package of entity classes implemented to support operations of the "Advisor". They are used to store information temporary. Most of the entity classes are modeled to resemble the classes in the ontology.

Jena API

Jena RDF API – It is a Java library that support retrieval, parsing and model manipulation for RDF. It implements the RDF model as a set of RDF Triples and provide resource centric methods for manipulating an RDF model as a set of resources with properties.

Jena RDQL Engine – Just like SQL, RDQL is a query language for RDF in Jena models. This RDQL engine matches the RDF statement pattern specified in a RDQL query against the directed graph in the RDF model to yield a set of matches.

Jena DAML API – It is a library to support a special RDF model for DAML. This model provides accessor methods to get DAML classes and properties in the model and it is also supported by the RDQL.

8.3 Sequence of major procedures

The following diagram (Fig. 8.2) shows the sequence of some critical procedures called between objects. Since the JSP pages are not classes, and only one instance is instantiated from each of the *advisor* class and the *modelBuilder* class, for simplicity, class names and page names are used instead of object names.

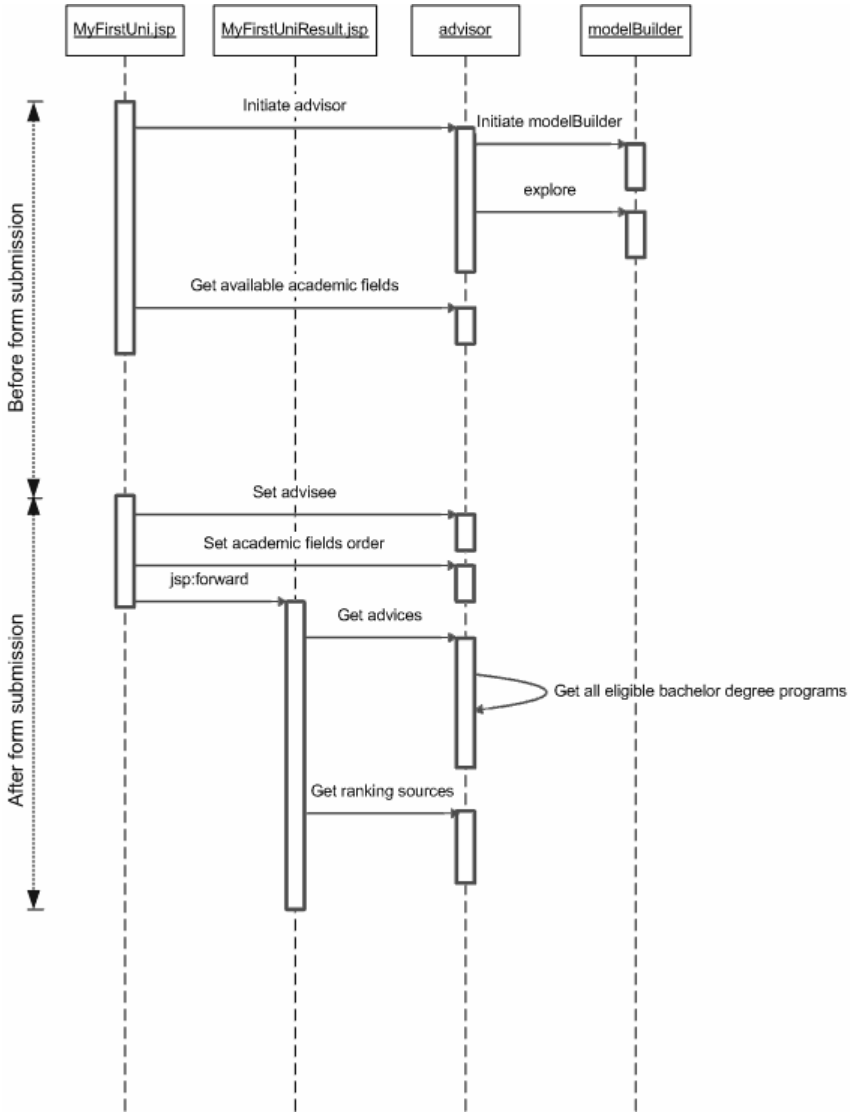


Fig.8.2 Sequence diagram of major procedures

Procedure description

- Initiate advisor

  `advisor()` , class **`advisor`** :

  When MyFirstUni.jsp is loaded, it automatically initiates its bean class *advisor* by calling the default constructor of class advisor. This constructor will invoke the constructor of its class member of *modelBuilder*, with a root URI as parameter. After that, another procedure call to this *modelBuilder* object will be made to retrieve RDF documents starting from the root URI.

- Initiate modelBuilder

  `modelBuilder(java.lang.String rootURI)`, class **`modelBuilder`** :

  This *modelBuilder* constructor is called by the *advisor* constructor. It takes a root URI as the start point to retrieve its interlinked RDF. The RDF content of this root URI is read in this constructor. Then the *modelBuilder* object is ready with model depth at 0.

- Explore

  `exploreByClass(java.lang.String wantedClass, int depth)` / **`exploreByDepth(int depth)`**, class **`modelBuilder`** :

  There are two implementations of Explore : "explore by class" and "explore by depth."

  The default Explore used by the *advisor* class is "explore by depth", which is a bounded depth first search to retrieve all interlinked RDF starting from the model root up to the specified depth.

  The "explore by class" method is an intelligent search. It performs the bounded depth first search up to the specified depth, and then extends the search to retrieve additional RDF resources of the specified class, at a known cost. For details about the search algorithm, please read chapter 9.

- Get available academic fields

  `getProgramFields()`, class **`advisor`** :

  This is a query to the RDF model to get the academic fields of all the bachelor degree programs in the model. The result is used to generate dynamic content on MyFirstUni.jsp.

- Set advisee

  `setAdvisee(myFirstUniData.student newStudent)` , class **`advisor`** :

  This is called when the form is being submitted. The caller stores all the examination records of the user in an object of entity class *student.* This object is passed to the *advisor* object by this procedure call.

- Set academic fields order

  `setFieldOrder(java.util.ArrayList newOrder)` , class **`advisor`** :

  This is also called during form submission. The caller stores the academic fields in an *ArrayList* following the order/priority specified by the user. This *ArrayList* is passed to the advisor object by this procedure call.

- Jsp:forward

  This is a simple JSP script to start another JSP page, MyFirstUniResult.jsp, and get it loaded in the browser.

- Get advices

  `getFinalAdvice()` , class **`advisor`** :

  This calls the getEligiblePrograms() method in the advisor object to yield a set of eligible bachelor degree programs for the advisee. This set of programs is then sorted in ascending order of score according to the simple formula:

  For N number of rankings (R_1, R_2, …. R_N) found for the university of a bachelor degree program and user priority P of the academic field of that program,

  $$\text{Score} = 0.3\left(\sum_{K=1}^{N} R_K / N\right) + 0.7P$$

  Such that the weight ratio of student's interest to average university rank = 7:3.

- Get all eligible bachelor degree programs

  `getEligiblePrograms()` , class **`advisor`** :

  This procedure invokes a series of other methods of the *advisor* object to return a set of bachelor degree programs that the advisee is eligible for. The internal process sequence is :

  1) get all universities that the advisee is eligible for,

  2) get all bachelor degree programs of universities from (1) and

  3) from (2), get all bachelor degree programs that the advisee is eligible for

  All these process are done by query to the RDF model using RDQL.

- Get ranking sources

  (**`getRankSourceWeb()`** , class **`advisor`**) :

  This procedure returns a set of URI of all the university ranking web site for which the ranking is retrieved and used by the *advisor* object. Again, this is done by querying the RDF model for all such URI exists in the model.

8.4 Screen design

The starting page:

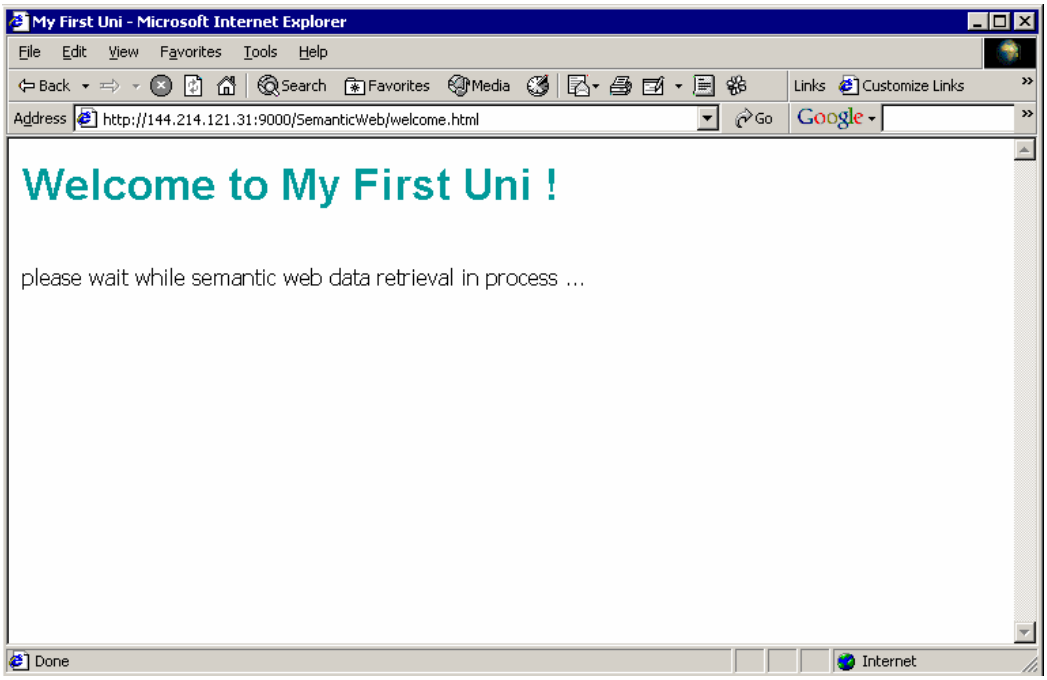


Fig.8.3 Starting page for My First Uni

The page shown in Fig 8.3 is the entry page of the "My First Uni" application. When it is loaded, it is in fact loading the *MyFirstUni.jsp* in the same browser. The starting page is used because when the *MyFirstUni.jsp* page starts, it takes a few seconds for the *modelBuilder* object on the server side to build the RDF model.

Screen design for MyFirstUni.jsp:

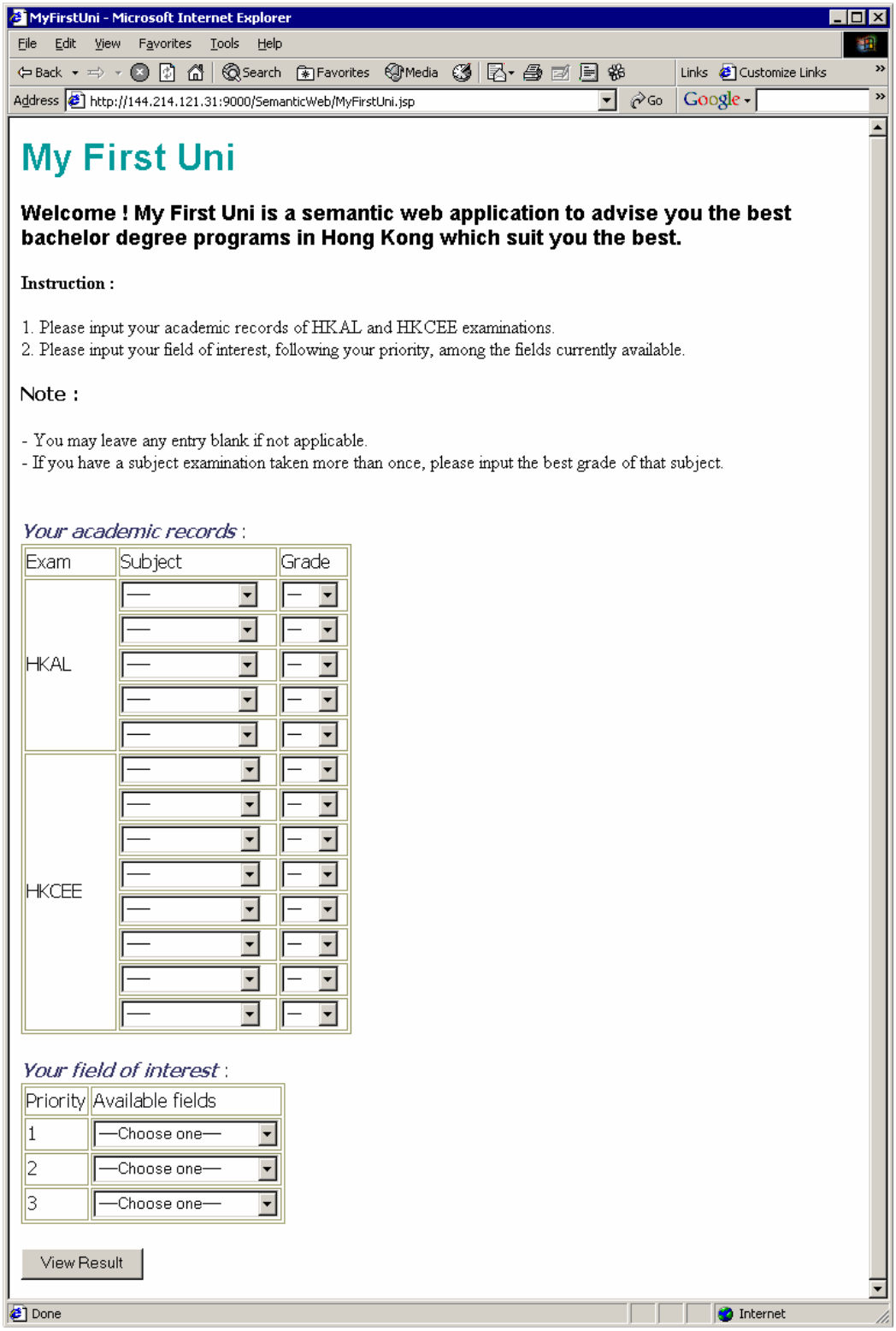


Fig.8.4 MyFirstUni.jsp

On the MyFirstUni.jsp, there is one static table and one dynamically generated table.

The static table shown in Fig.8.4 is the one for user to input his examination records. Since this is only a prototype, only part of the examination subjects of HKAL and HKCEE is available.

The dynamic table shown in Fig.8.5 is for user to input his field of interest in different academic subjects. User should complete the table according to their priority. The row of priority 1 is for the field that the user is interested the most.

Since the table is dynamically generated, the number of rows in the table depends on the number of academic fields of the bachelor degree programs available in the RDF model. All available academic fields in the RDF model will be shown in each of these drop-down boxes.

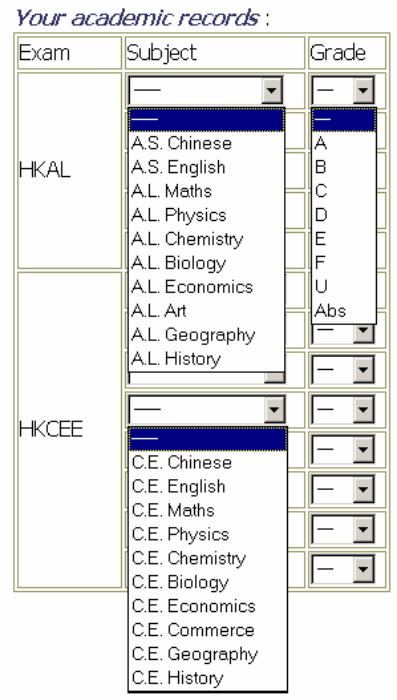


Fig.8.4 academic records table

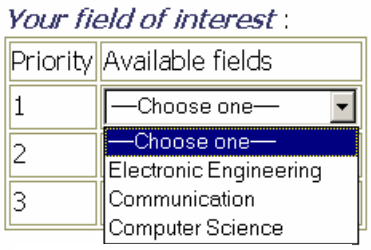


Fig.8.5 academic fields table

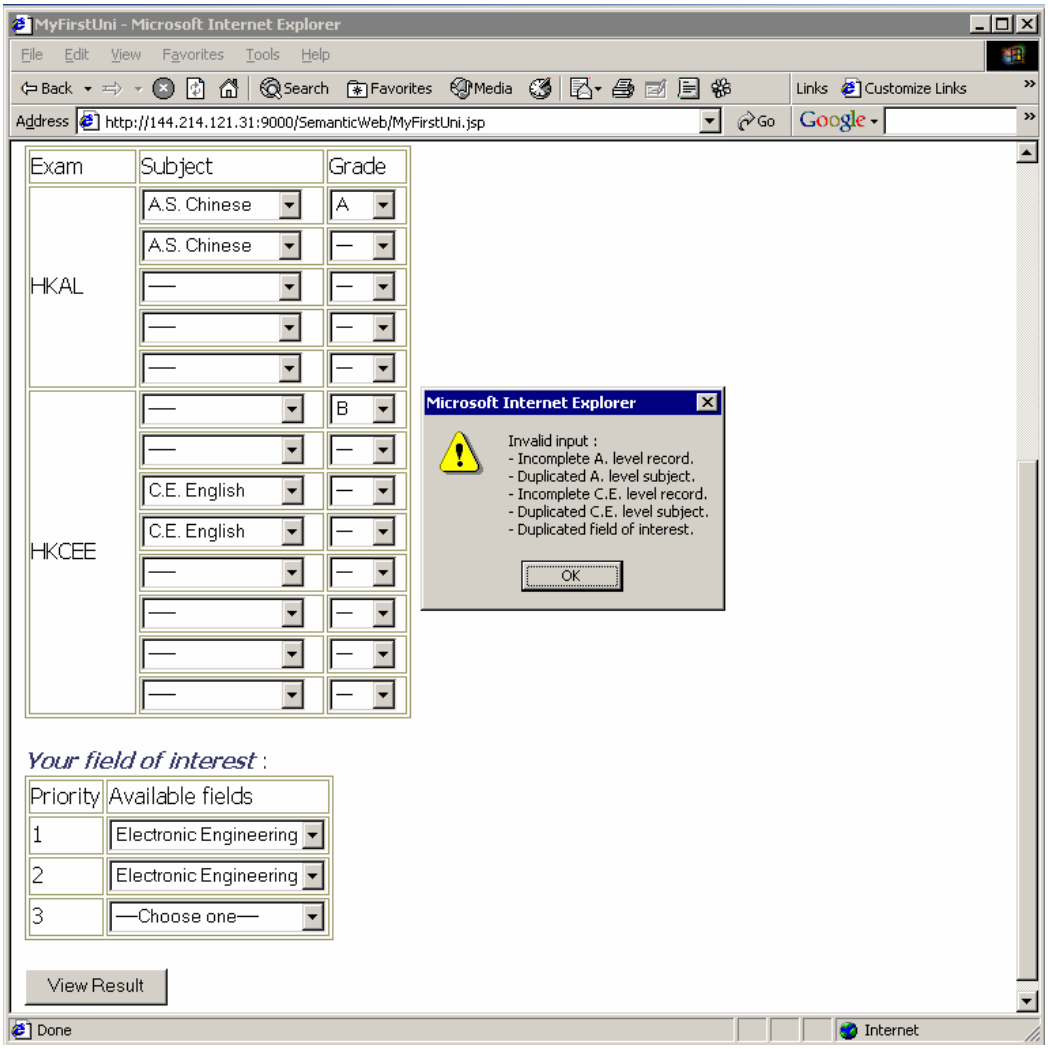


Fig.8.6 input validation

Fig.8.6 shows a message box informing user that there are input errors in the form when the form is submitted. There are totally 5 types of invalid input :

- Incomplete A. level record
- Duplicated A. level subject
- Incomplete C.E. level record
- Duplicated C.E. level subject
- Duplicated field of interest

For any occurrence of these inputs the corresponding error message will be included in the message box content. No message box will appear if all inputs are valid.

Screen design for MyFirstUniResult.jsp:

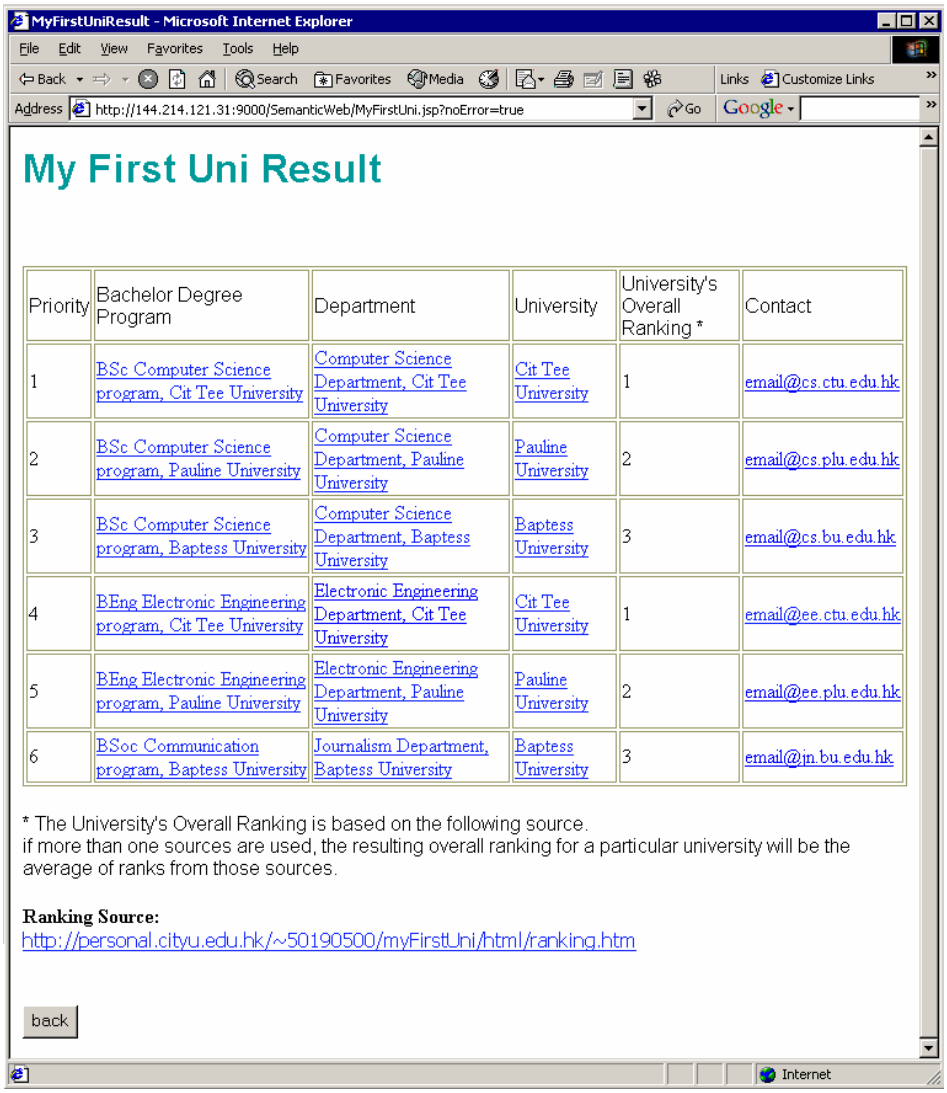


Fig.8.7 MyFirstUniResult.jsp

Fig.8.7 is the screen of *MyFirstUniResult.jsp* after the form in *MyFirstUni.jsp* is submitted. The content of the table is dynamically generated from information returned by **`getEligiblePrograms()`**. The links in the table are pointing to the HTML documents of the corresponding information.

The links under "Ranking Source" at the bottom are generated by **`getRankSourceWeb()`**, which show the URI of the referenced sources of ranking information.

8.5 application use cases

Here are screen captures of a few use cases. The following is a summary of admission requirements for reference.

University admission requirements			
University	Baptess University (BU)	Cit Tee University (CTU)	Pauline University (PLU)
University ranking	3	1	2
Minimum Admission Requirements	-Chinese (HKAL) Grade E	- English (HKAL) Grade D - Chinese (HKAL) Grade E	- English (HKAL) Grade D

Bachelor degree program admission requirements		
University	Program	Minimum Admission Requirements
BU	BSc Computer Science	- Mathematics (HKCEE) Grade E
	BSoc Communication	- English (HKAL) Grade D - Chinese (HKCEE) Grade C
CTU	BSc Computer Science	- Mathematics (HKCEE) Grade C
	BEng Electronic Engineering	- Physics (HKAL) Grade E
PLU	BSc Computer Science	- Physics (HKCEE) Grade E - Mathematics (HKCEE) Grade C
	BEng Electronic Engineering	- Physics (HKAL) Grade E - Mathematics (HKCEE) Grade C

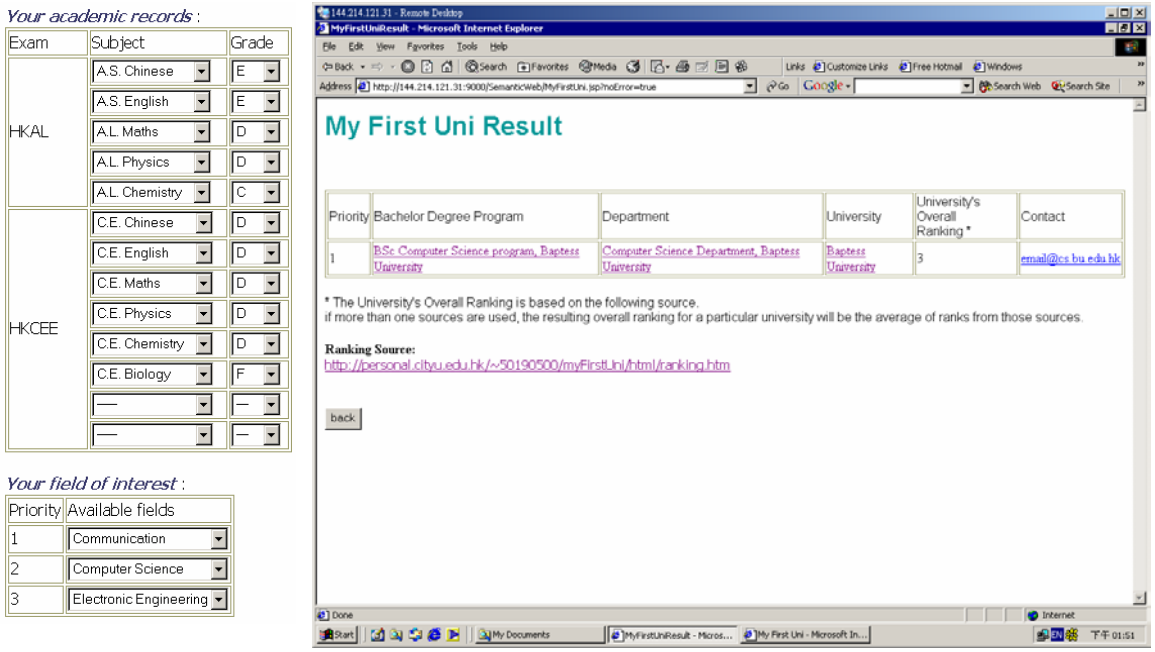


Fig.8.8 Use case 1

Your academic records :

Exam	Subject	Grade
HKAL	A.S. Chinese	C
	A.S. English	C
	A.L. Maths	D
	A.L. Physics	D
	A.L. Biology	D
HKCEE	C.E. Chinese	A
	C.E. English	A
	C.E. Maths	D
	C.E. Physics	D
	C.E. Chemistry	D
	C.E. Biology	D
	C.E. Economics	D
	C.E. Commerce	D

Your field of interest :

Priority	Available fields
1	Computer Science
2	Electronic Engineering
3	—Choose one—

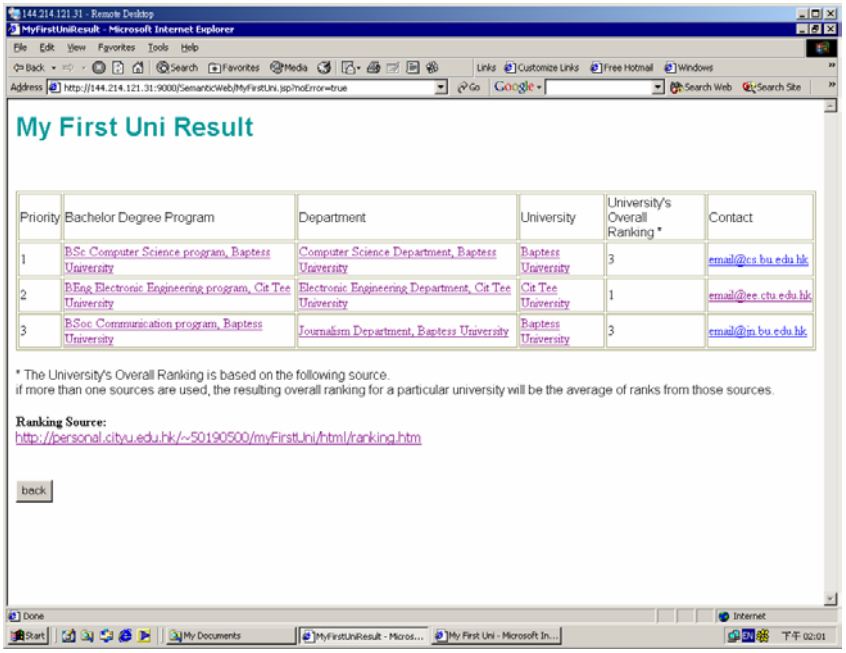


Fig.8.9 Use case 2

Your academic records :

Exam	Subject	Grade
HKAL	A.S. Chinese	A
	A.S. English	A
	A.L. Maths	A
	A.L. Physics	A
	—	—
HKCEE	C.E. Chinese	D
	C.E. English	D
	C.E. Maths	D
	C.E. Physics	D
	C.E. Biology	D
	C.E. Economics	D
	—	—
	—	—

Your field of interest :

Priority	Available fields
1	Electronic Engineering
2	Computer Science
3	Communication

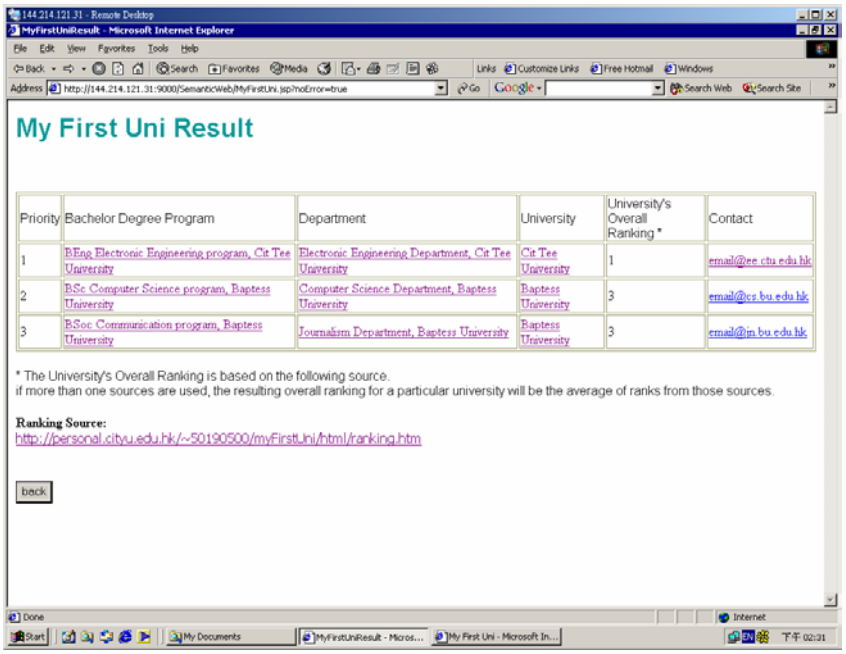


Fig.8.10 Use case 3

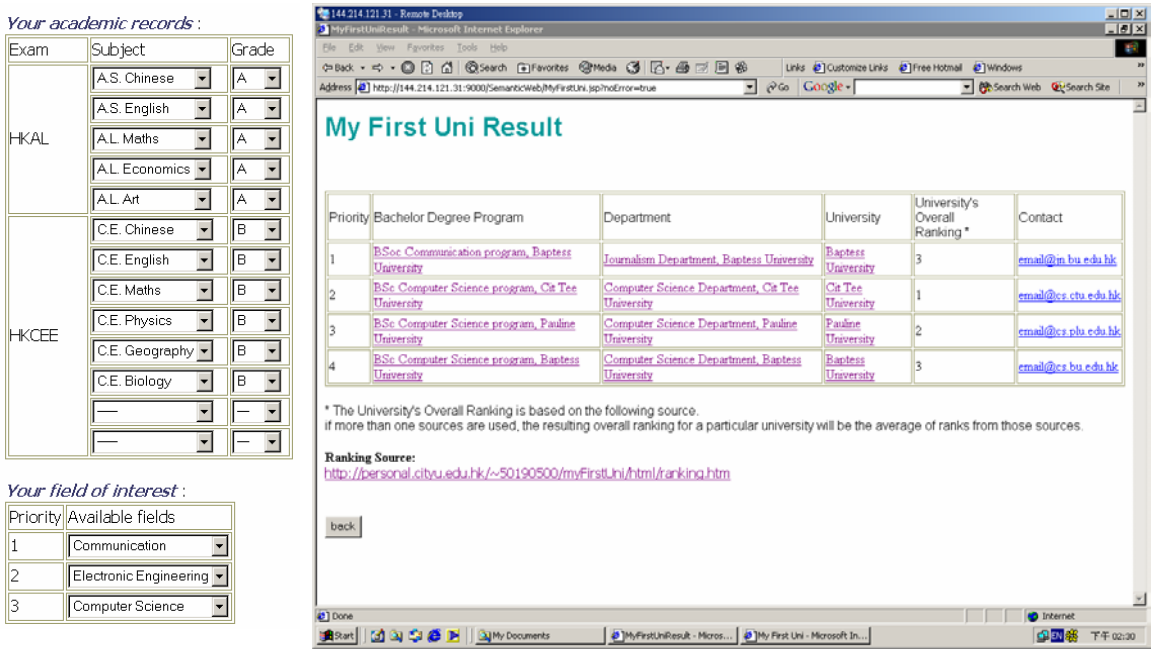


Fig.8.11 Use case 4

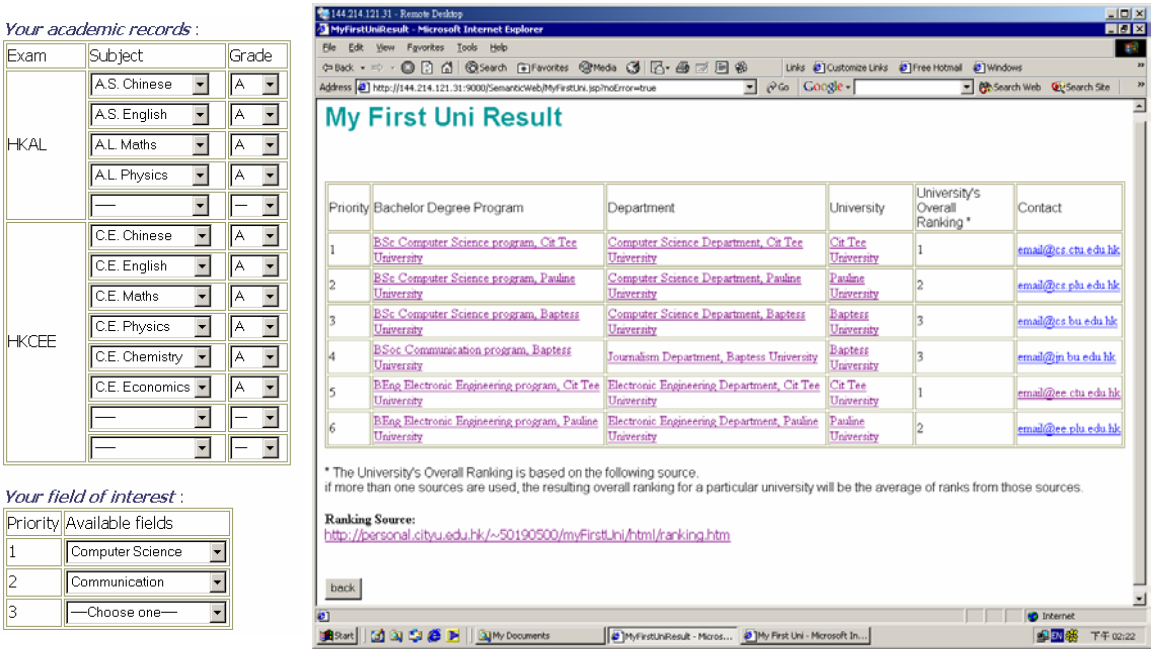


Fig.8.12 Use case 5

9. Search algorithm for Model Builder

To start building a RDF model, the Model Builder has to retrieve RDF documents from the web. To do this, a root URI has to be assigned to the Model Builder so that it can explore for other RDF documents linked directly/indirectly starting from the root URI. This technique is commonly used by most of the web crawlers today. The gist for effective web-crawling, however, is not about how the crawling starts, but how the crawler explores the links once it is started.

As mentioned in Section 8.3, the "explore" action by the Model Builder, which is a search process for RDF documents, has two implementations "Explore by depth" and "Explore by class".

Explore by depth

The first implementation "explore by depth" is a traditional bounded depth first search. It explores all the links to Semantic Web documents (.rdf and .daml files) blindly from the root URI up to depth N, (N is a method argument). This implementation, though it is not intelligent, guarantee all documents within the bound are retrieved. After the search, two models are built in the memory, one for all retrieved RDF and the other for all retrieved DAML.

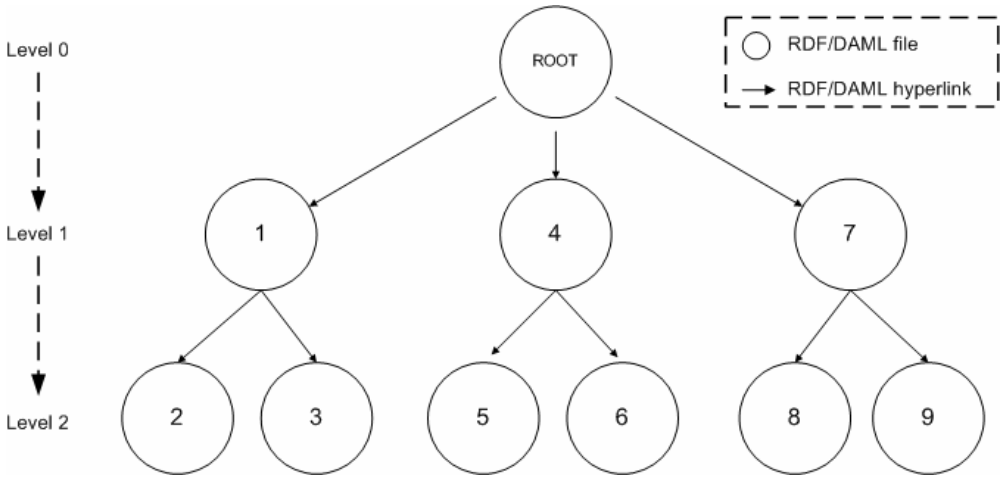


Fig.9.1 Explore by depth up to depth 2

Explore by class

The second implementation "explore by class" is a new intelligent approach building on top of the bounded depth first search. It is like a focused search that maximizes the number of retrieval of RDF resources of a specific class. To use "explore by class", a URI of a wanted class and a search depth have to be supplied as arguments.

The basic idea can be explained by an analogy of treasure hunting. Say a treasure hunter wants to hunt for ancient treasure. He can do it by either searching for treasure directly or searching for a map showing the location of the treasure. He starts his search at the centre of a circular boundary and search through everywhere within the boundary. When finished, he has found some treasures and some maps pointing to other treasures out of the searched boundary. The treasure hunter then follows those maps to hunt those out-of-the-boundary treasures by traveling a known distance given from the maps.

The "explore by class" search is similar to treasure hunting, with the search depth as the circular boundary, the instances (RDF) of the wanted class as treasures and the ontologies (DAML) as maps. This search algorithm involves three phases:

In the first phase, the ontology of the wanted class (which is the XML namespace URI of that class) is retrieved and stored in a DAML pool. Then, a bounded depth first search is performed up to the specified search depth. In this phase all RDF and DAML documents within the bound (plus the ontology of the wanted class) are retrieved and stored in the RDF pool and the DAML pool respectively.

In the second phase, an associated-class search is performed. All the "concepts" in the retrieved ontologies (DAML documents) are analyzed to look for cardinality constraints (min/exactly equal to N, where N>=1) of any property in class restrictions, which the range of the property is the wanted class.

The term "cardinality constraint" is the number of occurrence of a property. For example, one may define in an ontology a cardinality constraint for property "has-age", that "If anything has age, that thing has exactly one age". The formal expression for this in Description Logic is: =1 has-age

The term "class restriction" refers to one or more conditions that are always true for a class. For example, one may define in an ontology a class restriction to the class "human couple", that "for any human couple, if it has a child, the child must be a human". This is always true for all child of any human couple that they must be human but not alien! The formal expression for this in Description Logic is:
human couple $\sqsubseteq$ $\forall$ has-child.human

The term "range of a property" refers to the object in a subject-predicate-object triple. For example, one may define in an ontology a range for the property "has-age", that the range of "has-age" must be of class "number". The formal expression for this in Description Logic is: $\forall$ has-age.number

So the search is looking for a set of associated-class A, where

$A = A_1 \sqcup A_2$,
$A_1 \sqsubseteq$ (=N ANY.C) ,
$A_2 \sqsubseteq$ ($\geq$ N ANY.C) ,
N >= 1,
ANY = any properties
C = wanted class (for association level 1 search)

After the association level 1 search(the search for direct associated classes), an iterative operation of the same search will be performed. For every associated-class *a* in Set A obtained from the search above, the associated-class search is done again with wanted class C equals to *a.* This is the association level 2 search. It looks for all associated-class of an associated-class of the wanted class. In such way, this iterative search goes on up to level n, where there is no more associated-class of any associated-class found from the ontologies in the DAML pool.

At the end of the second phase, the Model Builder gets the maps. The cost to find a RDF resource of the wanted class starting from a RDF resource of any associated-class is the level number of that associated-class. Say if the wanted class C has an associated-class Y, and Y has an associated-class Z. Then from any RDF resource of class Z, the cost to find a RDF resource of class C is exactly equals to Z's association level, which is 2.

In the third phase, after a bunch of maps are processed in the second phase, the Model Builder starts to retrieve "out-of-the-boundary" RDF documents with a known cost. It search for all RDF resources of any associated-class in the set of leave nodes in its search tree, and explore it with the known cost if such resource is found in the leave node set.

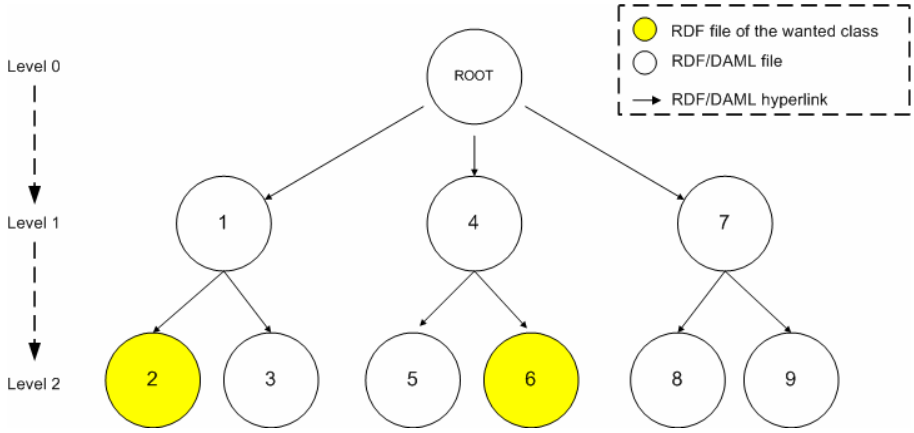


Fig.9.2 Explore by class : first phase

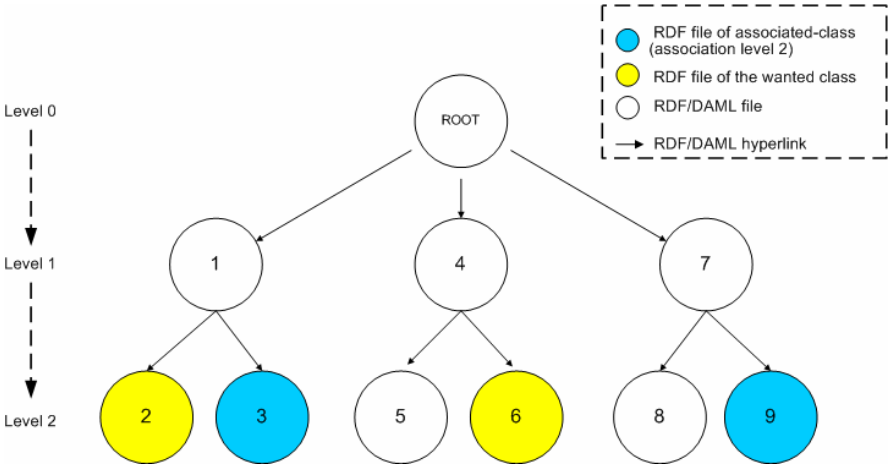


Fig.9.3 Explore by class : post-second phase

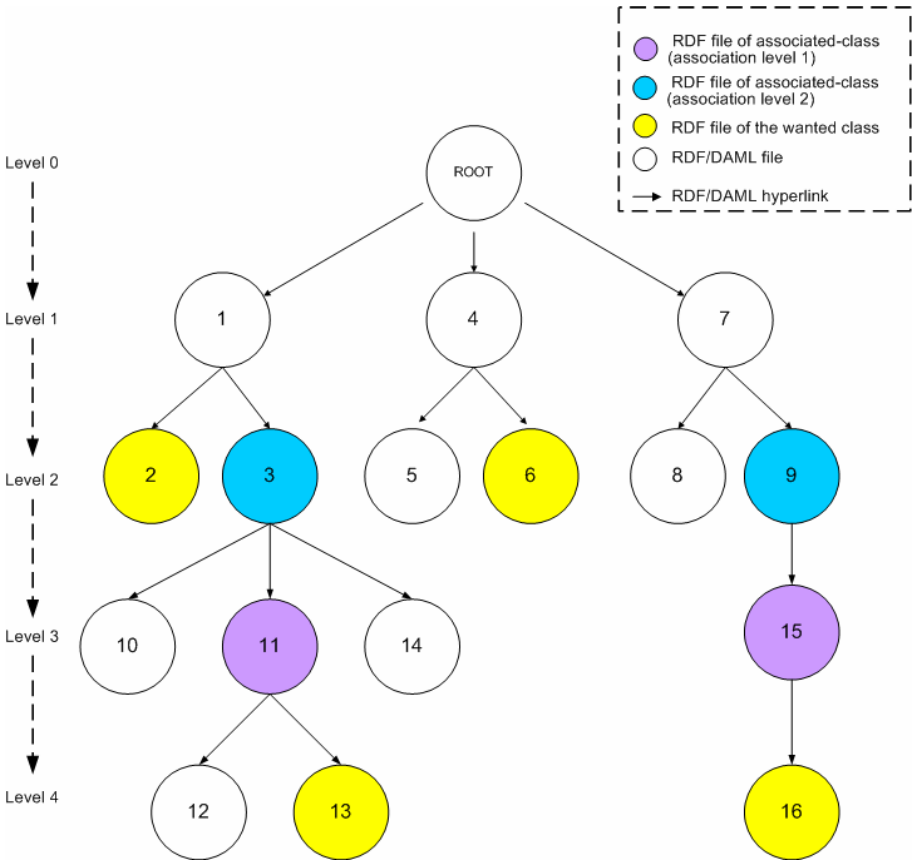


Fig.9.4 Explore by class : third phase

There are two assumptions that govern the effectiveness of this search algorithm:

1. In an ontology, some classes have necessary association with other classes by mean of class restrictions (Some classes have associated classes)
2. Instances (RDF documents) must obey the definitions in its ontology (DAML documents)

If (1) is not satisfied, this “explore by class” search is just the same as a bounded depth first search. The more the associated classes exist, the better the effectiveness of this search. If (2) is not satisfied, this new search is even worse than a bounded depth first search since extra cost is paid without getting any wanted documents.

10. Experiments showing the benefits of Semantic Web

By show the result of the following four experiments, this chapter aims to show the Semantic Web provides a better architecture for application to adapt to the open and dynamic nature of web and extends the web to its full potential.

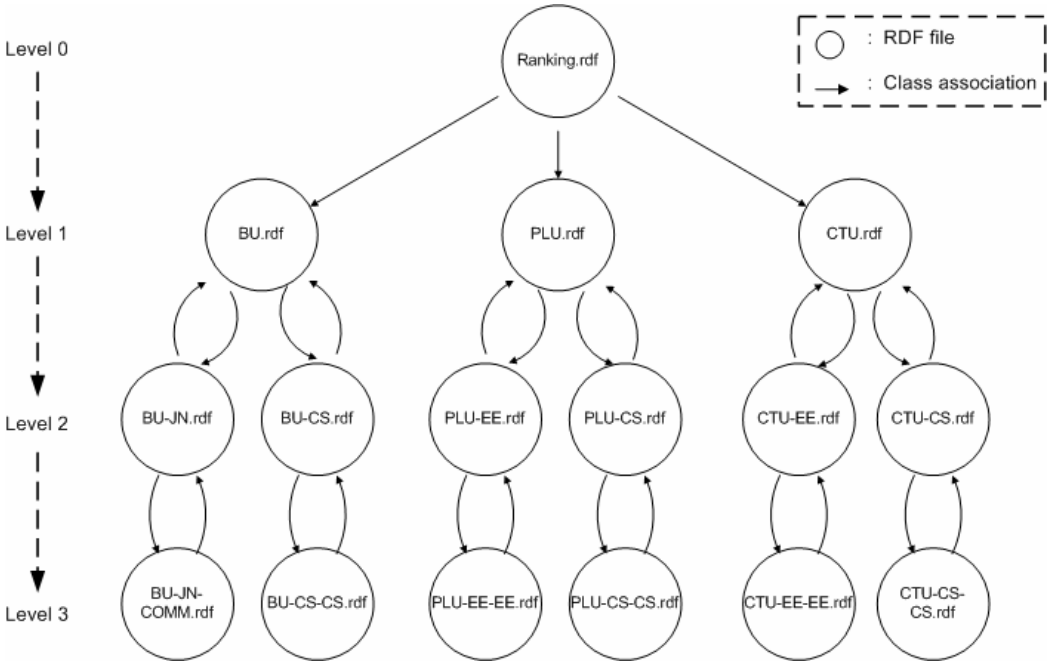


Fig.10.1 Basic set of RDF documents for "My First Uni"

Basic document set description

In Fig.10.1 the basic set of RDF documents used is shown. The nodes represent the RDF documents of 6 bachelor degree programs, 6 university departments, 3 universities and a ranking. It is common to all the up coming diagrams that the node at level 0 is always the root for building the model. The arrow of class association indicates that the class of a resource in a RDF document is an associated-class of the class which a resource, pointed by an arrow, belongs to.

For experiment 1 to 3, a modified set of RDF documents is used based on this basic set of "My First Uni" RDF documents.

Experiment 1 : Adaptation to newly available resources

This experiment mimics a situation that somebody has made a new RDF document about university ranking (INT-RANKING.rdf) and that the CS department of Cit Tee University has opened a new bachelor degree program (CTU-CS-IS.rdf). The root has been set to ROOT.rdf, which it has links to both ranking RDF documents.

In experiment 1, a few new RDF documents are added to the basic set as in Fig.10.2.

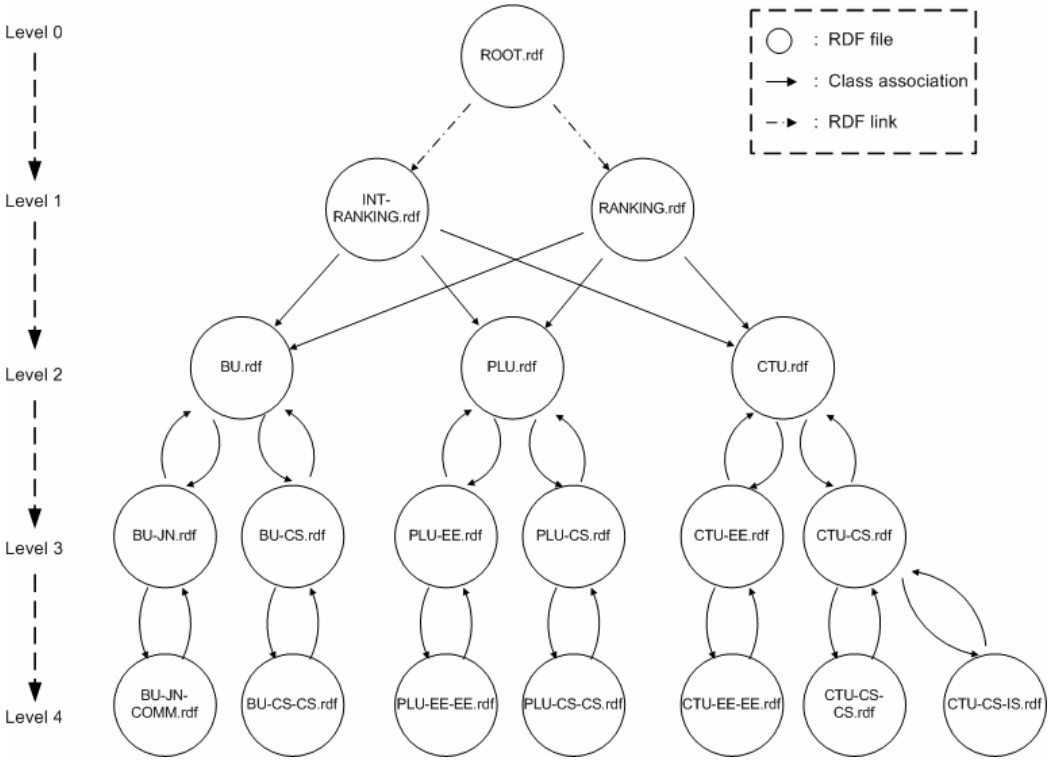


Fig.10.2 set of RDF documents for experiment 1

The experiment result shows that the "My First Uni" application can automatically adapt to, and make use of new resources of classes the application is looking for.

Fig.10.3 shows the information in CTU-CS-IS.rdf is added to the model so that a new academic field is available. Fig.10.4 shows that the information in INT-RANKING.rdf is added to the model so that the rankings of the universities changed to a 2-digits number. The "Ranking Source" at the bottom also shows a new entry.

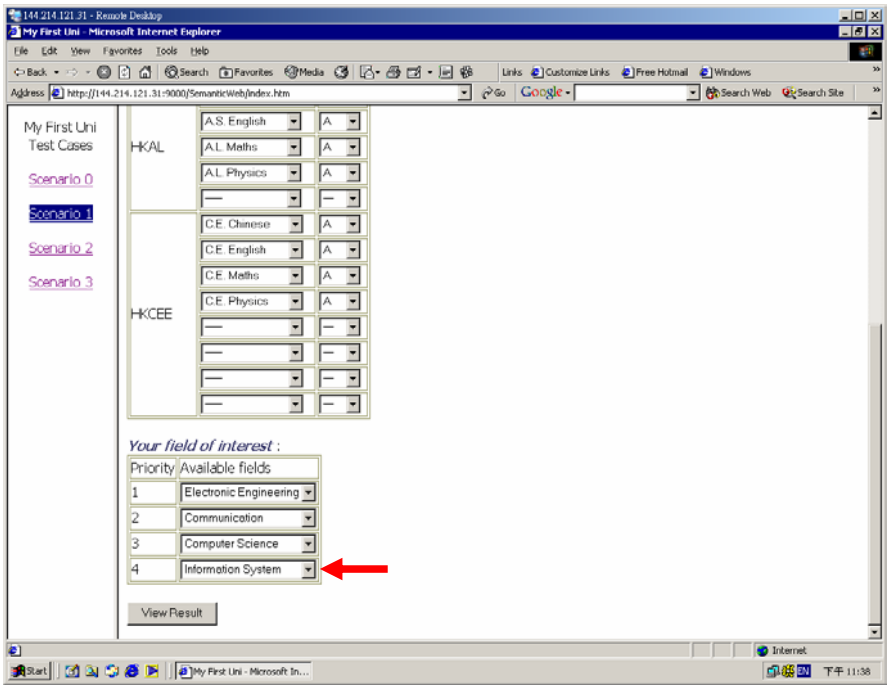


Fig.10.3 a new academic field is added

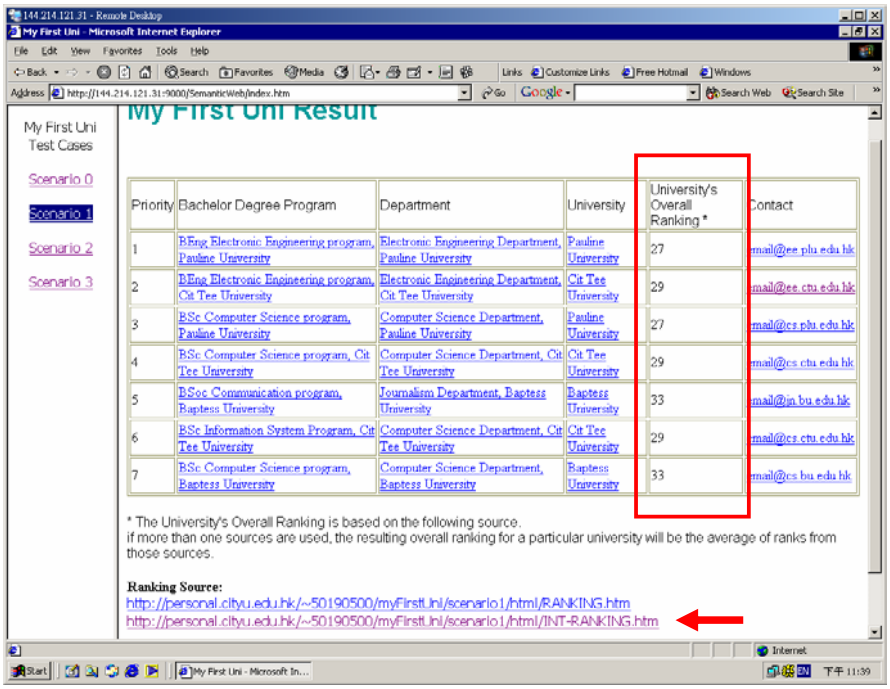


Fig.10.4 a new ranking source is used

Experiment 2 : Anyone can say anything about anything anywhere

This experiment mimics a situation that some new vocabularies from some unknown ontologies are added to the existing RDF documents. In addition, the complete description of a RDF resource is distributed into more that one RDF documents.

In experiment 2, a few new RDF documents are added and some are modified in the RDF set used in experiment 1 as in Fig.10.5.

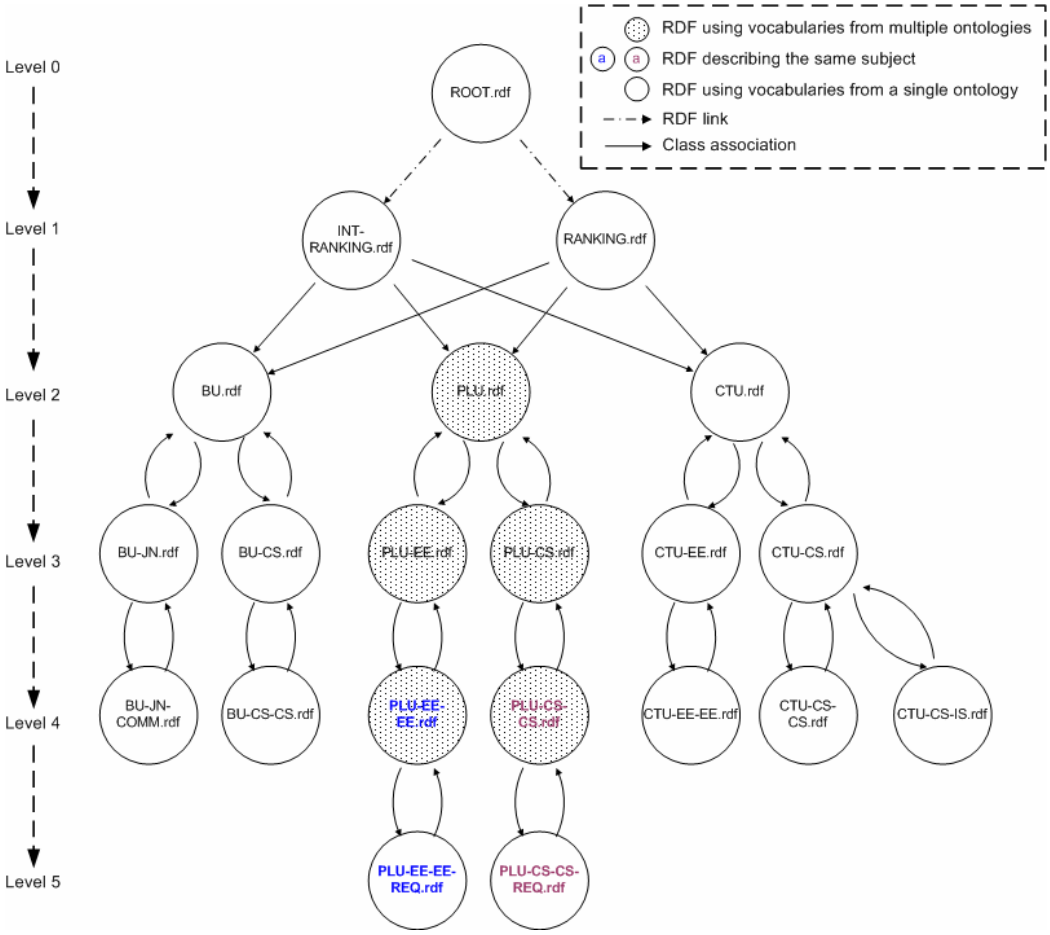


Fig.10.5 set of RDF documents for experiment 2

Taking the PLU-CS-CS.rdf as an example, new vocabularies (which are highlighted in yellow in Fig.10.6) are introduced as the RDF author can describe any resource with any vocabulary from any ontology. The author can even describe the admission requirements of PLU-CS-CS in a separate file PLU-CS-CS-REQ.rdf instead of putting everything in PLU-CS-CS.rdf.

```
<rdf:RDF xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
 xmlns:ns0="http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#"
 xmlns:ns1="http://derpi.tuwien.ac.at/~andrei/cerif-rdf-dc-mn.daml#"
 xmlns:ns2="http://grcinet.grci.com/maria/www/CodipSite/Onto/WebSite/WebSiteOntology_V27Aug2001.daml#"
 ..... (omitted)
<rdf:Description rdf:about="http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario2/rdf/PLU-CS-CS.rdf#PLU-CS-CS">
..... (omitted)
<rdf:type>
..... (omitted)
<ns0:has-name>
  <xsd:string xsd:value="BSc Computer Science program, Pauline University" />
  </ns0:has-name>
<ns1:CERIF.ev_desc_language>
  <xsd:string xsd:value="English" />
  </ns1:CERIF.ev_desc_language>
  <ns2:page-content rdf:resource="http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario2/rdf/PLU-CS-CS-REQ.rdf" />
</rdf:Description>
</rdf:RDF>
```

Fig.10.6 RDF content of PLU-CS-CS.rdf

The result in Fig.10.7 shows that the modified RDF documents are still processable by the application without change in program codes. In Semantic Web, description of a resource by multiple RDF documents using different vocabularies can be integrated in the RDF model. Semantic Web application interprets the model selectively. It takes relevant information for processing and ignores the rest.

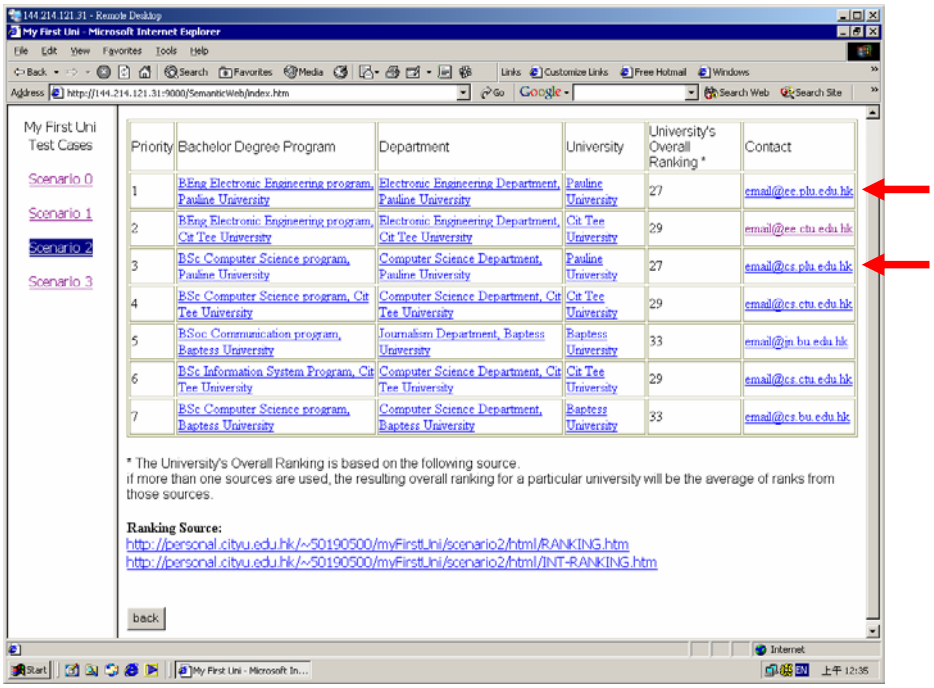


Fig.10.7 all RDF documents are processable in experiment 2

Experiment 3 : Intelligent web crawling

This experiment aims to show how the intelligent search, *explore by class*, mentioned in chapter 9 can apply to a real situation to maximize the retrieval of RDF resources of a wanted class for a small bound depth first search.

In experiment 3, new RDF documents using new ontologies are added to the RDF set used in experiment 1 as in Fig.10.8.

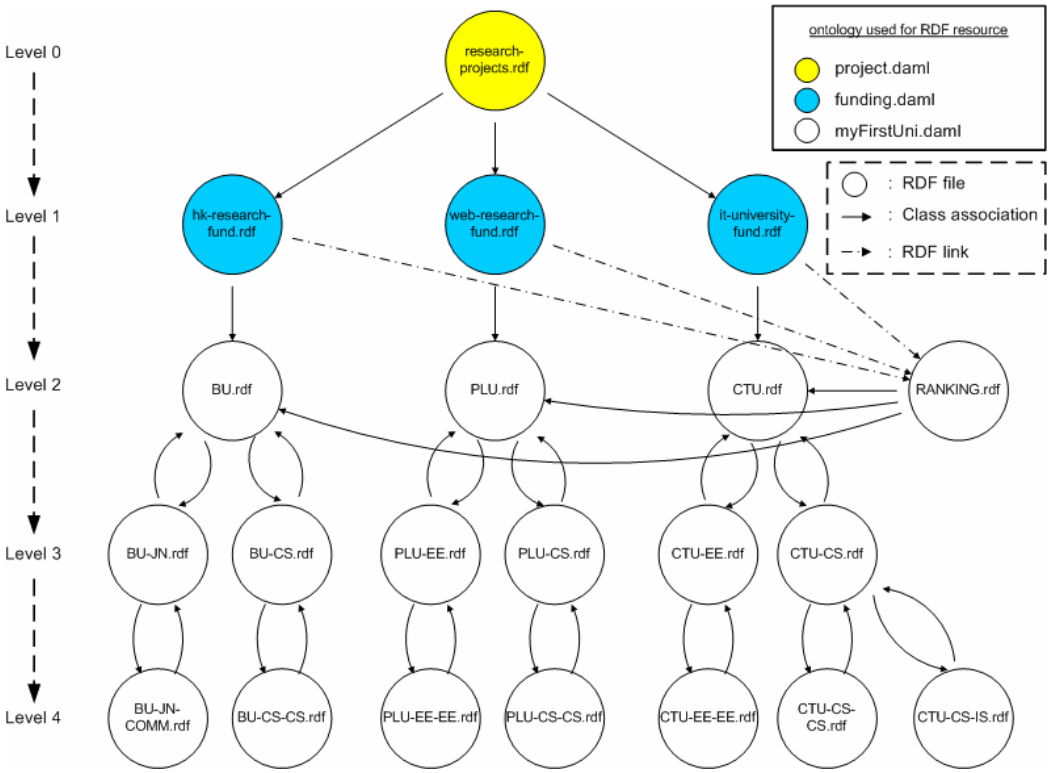


Fig.10.8 set of RDF documents for experiment 3

In this experiment:

- By ontology “project.daml”, class “Research-Project” is an associated-class of class “Research-Funding” of ontology “funding.daml”.
- By ontology “funding.daml”, class “Research-Funding” is an associated-class of class “University” and “University-Ranking” of ontology “myFirstUni.daml”.
- By ontology “myFirstUni.daml”, classes, “University” and “University-Ranking” are associated-class of class “Bachelor-Degree” of the same ontology.

Such that, it is only necessary to do a bounded depth first search up to depth 2 (the minimal depth deep enough to retrieve all related DAML documents), and the intelligent search will get all needed RDF documents straight-away down to level 4.

The search process is shown in Fig.10.9. The root URI for search is stated in line one, first phase of the search is in line 2-3, second phase is in line 4 to 12. Note that "interlinked class" means the associated-class of the wanted class or other associated-class. The rest of lines are the files successfully retrieved by the Model Builder.

The result shows that, an intelligent search with search depth 2 can achieve the same result of a bounded depth first search with search depth 4 in retrieving a specific class of RDF resource, provided that some class associations exist between classes.

```
Model root node : http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/research-projects.rdf
Retrieving RDFs and DAMLs using bounded depth-first search to depth 2 , please wait ...
Ready.
Extending search to look for class "http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#Bachelor-Degree" RDFs , please wait ...
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#University
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#Bachelor-Degree
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/project.daml#Research-Project
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/funding.daml#Research-Funding
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#University-Ranking
interlinked class :   http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml#University-Department
Ready.
-------RDF files in model-------
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/PLU.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/PLU-EE.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU-CS-IS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/BU-CS-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/BU-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/BU-JN.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/it-university-fund.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU-EE.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/BU-JN-COMM.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/hk-research-fund.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/PLU-CS-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU-EE-EE.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/research-projects.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU-CS-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/RANKING.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/CTU.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/PLU-EE-EE.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/PLU-CS.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/web-research-fund.rdf
http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/BU.rdf
-------DAML files in model------
http://personal.cityu.edu.hk/~50190500/myFirstUni/funding.daml
http://www.ksl.stanford.edu/projects/DAML/ksl-daml-desc.daml
http://personal.cityu.edu.hk/~50190500/myFirstUni/project.daml
http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml
------Leave nodes in model------
No. of RDF Resource in model :21
model depth :4
```

Fig.10.9 message dump on server side when using "explore by class" in experiment 3

Experiment 4 : Foundation for inference

This experiment aims to show the capability of Semantic Web in ontology inference. To do it, a simplified version of the original ontology "myFirstUni.daml" is created. It consists of the following classes and definitions:

Class	Class property*	Class restrictions (informal)
Associate-Degree	SubclassOf	"Associate-Degree is only offered by a Technical-Institute"
Bachelor-Degree	SameClassAs	"Bachelor-Degree is only offered by a University-Department" "Bachelor-Degree has a length"
Qualified-Degree	SameClassAs	"Qualified-Degree is only offered by a University-Department"
Hons-Degree	SameClassAs	"Hons-Degree has exactly one honor type of Honors"
HK-Bach-Degree	SubclassOf	"HK-Bach-Degree is only offered by a University-Department" "HK-Bach-Degree has a length" "HK-Bach-Degree has exactly one honor type of Honors"
Technical-Institute	SubclassOf	--
University-Department	SubclassOf	--
Honors	SubclassOf	--

*Class property of a class can be one of the followings:

SubclassOf – SubclassOf property defines that the class restrictions are necessary for class membership, but not sufficient to describe membership. Logically, let P be the member of class X having SubclassOf property and Q be the class satisfying all class restrictions of class X, such that $P \rightarrow Q$

SameClassAs – SameClassAs property defines that the class restrictions are necessary for class membership, and sufficient to describe membership. Logically, let P be the member of class X having SubclassOf property and Q be the class satisfying all class restrictions of class X, such that $P \leftrightarrow Q$.

The FaCT description logic classifier (SHIQ reasoner) that integrated with *OilEd* editor is used as the reasoner for this experiment. The ontology is opened in *OilEd* and the ontology is classified by the reasoner to decide the class memberships.

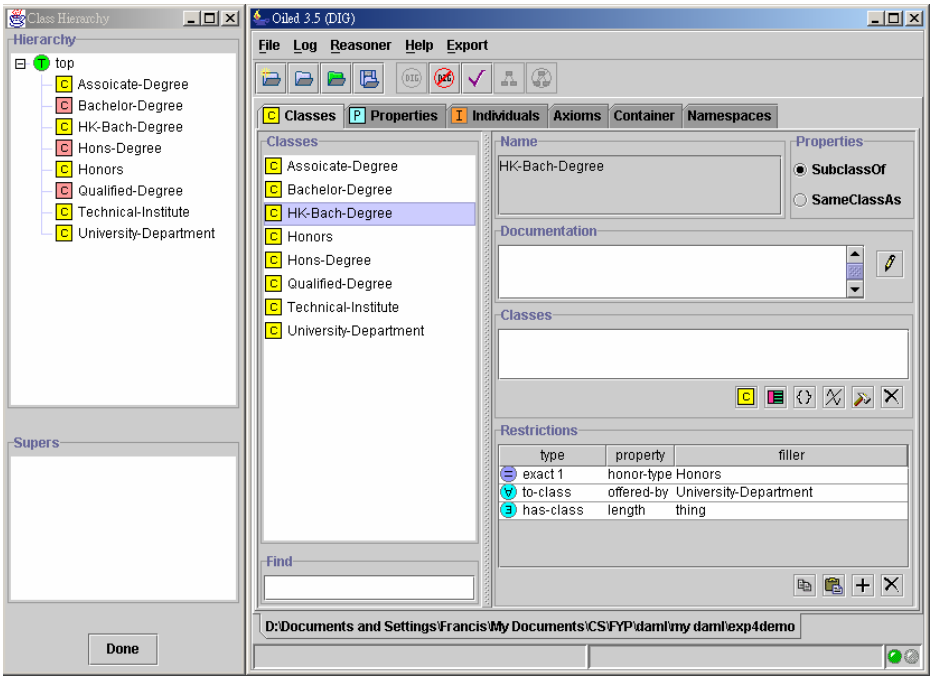


Fig.10.10 Class relationship before classification

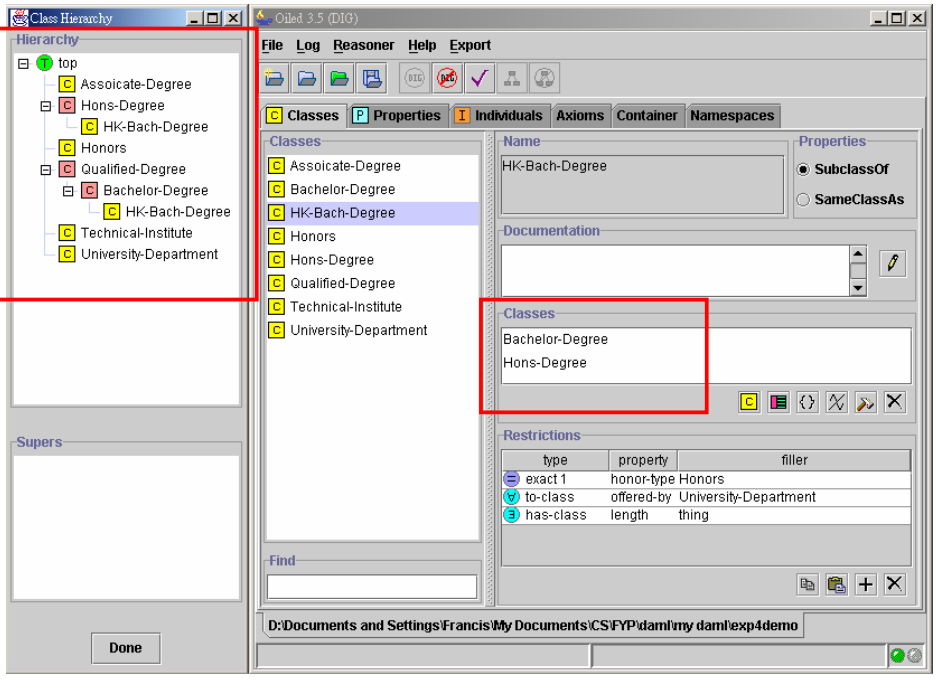


Fig.10.11 Class relationship after classification

From the result shown in Fig.10.11 that the reasoner is capable to process the ontology and infers the following conclusion:

- HK-Bach-Degree is a subclass of Hons-Degree.
- Bachelor-Degree is a subclass of Qualified-Degree
- HK-Bach-Degree is a subclass of Bachelor-Degree

From this experiment, it is clear that the semantics described in DAML+OIL, which follows the Description Logic theory, is processable by inference engines.

With such capability to decide class relationships, programs can map and recognize concepts from different ontologies, this is an essential support to achieve the goals of adapting to dynamic and opened web environment, and allow anyone to say anything about anything, without sacrificing the machine processability.

We foresee that in the age of Semantic Web, applications and agents will incorporate with reasoning engines to work and make decisions using knowledge from the web documents.

11. Summary

11.1 Proved benefits of Semantic Web

By doing this project, a number of new possibilities have been explored which will greatly extend the capability of web technology today. Here it is a summary:

- All information on the web is machine-processable to allow web applications to answer a complicated question by manipulating multiple sources of information from the web and consolidate them to generate a correct answer.
- Control and maintenance of information is distributed to the information owners.
- Access of information is opened to public.
- RDF and DAML documents storing machine-processable information allow great flexibility in structure, so that anyone can say anything about anything.
- RDF and DAML documents can be discovered and classified effectively and precisely on a dynamic and opened web.
- All information has to have well defined semantics resided within itself, not with any application, so that it is inter-processable by any machine process.
- Concepts represented in DAML+OIL are possible to be inferred and mapped between each other.
- It is an extension to the current web, not a reconstruction. Existing web pages, which are for human consumption, can be retained. It is just an effort of deriving a RDF version of the existing information, and linking it to the existing web page.

Though it seems to be a perfect solution to the web, its research and development is still in progress. The Semantic Web was found in 1998, but it takes much longer time than XML to be widely adapted simply because it is far more complicated to weave the Semantic Web than adding XML to the current web.
In the rest of this chapter, evaluation of the Semantic Web from various points of view are brought to discussion.

11.2 RDF vs XML

XML and RDF are two complimentary technologies being used to build an internet that is more intelligent and machine processable. It is a frequently asked question that why don't we just stick to XML as programs can read XML document. The short answer is the difference between syntax and semantics, structure and meaning. The long answer is elaborated as follows.[16]

1) Semantics to syntax mapping is many-to-many in XML

Recall that XML model is a tree-like structure. It is good for representing parent-child relationship in a document. But it is not a way to express meaning which is more that parent and child. Like English, meaning is conveyed basically through subject-verb-object grammar. Given the following XML tree fragment:

```
<love>
    <man>beta</man>
    <woman>alpha</woman>
</love>
```

```
<man href="beta">
  <love>alpha</love>
</man>
```

```
<woman>
  <details>
   <name>alpha</name>
   <love>
      <man>beta</man>
   </love>
   </details>
</woman>
```

```
<man>
   <love>woman</love>
   <details>
      <alpha>beta</alpha>
   </details>
</man>
```

It is hard to tell whether "beta loves alpha" or "alpha loves beta" or "beta loves man" or "woman loves alpha" or "beta love is man" or "man loves woman is an alpha of beta" …. It is possible for all four trees to represent the same meaning. Yet it is also possible for any one of them to convey different meanings when it is interpreted by different applications.

Since it is impossible to answer "who loves beta" from the tree, it is impossible to develop a query mechanism on XML to answer such question.

[16] http://www.semaview.com/d/RDFandXML.pdf

But in RDF, everything is a triple, the subject-predicate-object semantics is enforced:

```
<Description about="alpha" love ="beta" />
```

```
<Description about=”alpha”>
        <love>beta</love>
</Description>
```

```
<Description about=”alpha”>
        <love ="beta" />
</Description>
```

Although a triple can be represented in different syntaxes, there is only one way to interpret a triple of a particular syntax. Thus syntax to semantics mapping is many-to-one. Therefore, querying a RDF model is possible. In reality, query language like RDQL has proven this possibility.

2) XML trees are difficult to be integrated

For every XML document, it represents a single XML tree. For representation of a complex data structure, it has to be represented in one complete tree in one XML document. It is impossible, or at least very difficult, to have a complex data partially represented by one tree and the rest by another. The complete description of something has to be completed in one single tree.

In contrast, all information in RDF are represented as triples, thus they can be integrated. Descriptions of a single resource from different triples in different RDF documents can be integrated as long as they are describing the same resource. (See experiment 2 in Chapter 10)

3) XML documents are difficult to be discovered from one another on the web

It is a simple point that data values in a XML document do not necessarily be URIs. This makes XML documents isolated from one another.

In RDF, all resources (as subjects or objects) and predicates in triples have to be represented as URI and the triple itself is also identifiable by URI. Therefore everything in RDF can be linked up with each other, and thus can easily be discovered on the World Wide Web, provided that the URI is resolvable on the web. (See experiment 1 in Chapter 10)

4) Order matters in XML

In XML, the order which data are listed is very important to the application. If the order of elements is altered, the parser will not be able to work with the document.

Using RDF model, however, the order of triples does not affect the ability the application process the data.

5) Understanding of XML schema is essential

To use any of the data in XML, developer of an application must understand the complete XML schema governing the document.

It is not necessary for an abstract RDF model, that application can just make use of those with known vocabularies, and discard the rest. (See experiment 2 in Chapter 10)

6) Meaning of vocabularies in XML is not known to computer

For XML, all the program can know about is the position of the vocabulary in the XML tree defined in the schema. There is no way to infer or deduce any meaning about a vocabulary.

For RDF, every vocabulary is linked to a concept definition, the ontology. The ontology describes the concept of a vocabulary in formal logic notation which is understandable by computer. By the ontology, machine gains the capability to infer and deduce about concepts and knowledge. (See experiment 4 in Chapter 10)

7) Getting two XML applications to talk requires a common syntax

For applications using different syntax in their own XML data, data exchange between them is impossible unless their documents are converted to follow a common schema.

Theoretically, for Semantic Web applications, two parties may communicate with data, in different syntax and vocabularies but of an equivalence concept, directly with the help of inference logic. Yet research and development on this is still in progress.

8) Reusability of XML schema

It is hard to define a XML schema to suit the needs of several different applications. Thus every XML application tends to develop its own schema, which is one of the cause of problem (7).

It is a fact that the number of concepts in this world is much smaller than the number of representations of these concepts. So, for Semantic Web application, the ontology (which is the "schema" of vocabularies) is much more reusable as people mostly talks to each other with common concepts. Therefore, developer seldom needs to develop a new ontology unless the concept is new to the ontology world.

Why Not RDF?

Despite the fact that RDF provides machine with better capability, there is some reasons why we keep using pure XML:

1) Rapid development

To use RDF, developers have to spend more effort to plan how to model the concept they want to represent even reusing an existing ontology. Development is particularly complicated if it is necessary to develop a new ontology.

In contrast, one may make things easier with pure XML, as the effort is just to define the structure and format of the document.

2) Popularity and support

XML is new, but RDF is even newer. Currently there are little people who know about RDF out of the research community. It may not be a good time for a company to use RDF if its business partners do not step forward. Furthermore, tools for Semantic Web development are very limited. Not even every browser supports the RDF standard.

3) Strict structure and forced validation against schema

For security reason, some applications may not prefer the great flexibility in structure proved by RDF. There could be vulnerability if an unauthorized party makes RDF description to describe a resource which alters the meaning of the original description of that resource.

In XML, the data structure has to be complete within a single XML document. This gives better control and management to the user. In addition, XML document is forced to be validated against its schema for consistency. Therefore, XML ensure better integrity of documents.

4) Hidden semantics

Another security concern is the confidentiality of document's semantics. It is not desirable for confidential data exchange to allow semantics to be embedded in the document. It is more secure to first make an agreement on document's semantics with all exchange parties and then exchange the documents in XML.

Conclusion

Not every project is well suited for using RDF. RDF means to wide spread the knowledge enabling highly intelligent and automated processing. Meanwhile, XML is good in terms of information security and suitable for rapid development. As a developer, it is important to decide which language standard to be used to bring the most benefits to the project. In general, XML is the choice for internal applications with restricted access whereas RDF is the answer to make documents more useful on the World Wide Web.

11.3 Evolution of the World Wide Web

Here are tables to show the differences between three ages of World Wide Web in terms of five areas: human usage, searching, machine processing, development and significance.

Original WWW with HTML	
Human usage	- A rich information source. - Human deal with input, review, assimilation, integration and action using the web information. - All web documents are for human consumption
Searching	- Global documents and terms indexed. - Results are generated by pattern matching and string matching - Non-text resources (multimedia) are difficult to search
Machine processing	- Limited to string matching and screen formatting. - Simple program execution with program script in the document.
Development	- Easy and rapid: It is just the web designer's job to format the content in HTML format
Significance	- An amazing creation which is the first successful attempt to share huge amount of information in a world-wide scale. - Every documents are interlinked to each other by URI forming a huge network of documents

Current WWW with XML	
Human usage	- Still a rich information source - With XML, part of the effort in processing web page content is transferred to XML enabled web applications. - Most documents are for human consumption
Searching	- Precise searching within a XML enabled web site is possible. - Global searching is not improved.
Machine processing	- Within individual XML web site, application is able to extract data from XML documents for processing. - Web Services, which was made possible by XML, further enhances the capability of machine processing on web as service can be shared with each other. - Since semantics does not reside on the XML document, these XML documents are not processable by third party's application.
Development	- Effort 1: Design of a XML schema by system designer - Effort 2: Production of XML documents conforming to the schema by editor - Effort 3: Design of a style sheet for XML to be displayed on screen by web designer - The XML syntax has to be agreed by all parties using the XML documents. - The semantics of the XML has to be made known to all parties using the XML documents.

Significance	- A meta language used to define other domain or industry specific language without the complexity of SGML - First machine processable web language - An extension to previous web architecture for machine processing - A concept to separate machine processing information and human consumed information

Semantic WWW with RDF, DAML+OIL …	
Human usage	- Will still be rich information source - Web document processing will be fully automated. Human simply gives command to the web agents to obtain answers. - Documents for machine processing will be as much as documents for human consumption.
Searching	- Local and global searching can both be done intelligently and precisely. - A concept matching process. - Expected to combine with syntax matching technique to yield highly accurate result. - With RDF describing multimedia files, they become searchable
Machine processing	- To a machine, the web is just like a OODB for query and process. - To a machine, the web is not only a database, but also a huge knowledge base. Intelligent agents can learn different concepts from ontologies as they traverse across the web to help them to make intelligent decisions. - Every web document can be processed by any parties as semantics is resided within the document - New knowledge, as the deduction from existing concepts, can be automatically asserted on the web -

Development	- Effort 1: Find or to create one or more suitable ontology by a domain expert together with an ontology engineer - Effort 2: Integrate different concepts in a RDF document to avoid contradiction between concepts by ontology engineer - Effort 3: Complete the RDF document with property values by editor - Effort 4 (can be automated in the future): Check the consistency between a RDF document with its ontology
Significance	- Web documents that carry formal unambiguous semantics. - Largely automated web content processing and enabling intelligent services. - Provide a large pool of knowledge for machine learning and data mining. - Accurate and intelligent web searching is possible.

11.4 Challenges for Semantic Web

Weaving the Semantic Web is a grant vision. Its research and development is happening rapidly but it seems that we have not yet reached a mature stage. There are quite a number of challenges for researchers to work on. Some of these examples are listed as an ending of this report.

1) ***Ontology validator for RDF document***

   As mentioned in previous chapter, currently there is no validator for RDF to validate RDF documents like XML validator for XML documents. The validator is essential to check the consistency between the RDF triples and their ontology. For example, an ontology defines that cat must have exactly four legs. But there is no mechanism to enforce a cat" RDF to conform to the class restriction of cat. Therefore, a validator is needed to guarantee the correctness of information for machine processing.

2) ***Namespace resolvability***

   For a RDF validator and other application to learn about the ontology, the namespace URI of the DAML document (stated in the RDF document) must be resolvable. Yet currently there is no guarantee that the URI is pointing to anything. It remains a question that whether we should enforce this resolvability because the design principle of web is to leave it unbounded, we have to sacrifices link integrity for scalability. However, If we allow "404 Semantics not found" error in Semantic Web, does Semantic Web still work?

3) ***Ontology mapping***

   It is an edge of Semantic Web that a single concept represented by different vocabularies is known to be equivalence to machine by inference. But what if there is no sufficient evident to make such inference?

4) Incorrect knowledge

As anyone can say anything about anything on the web, people may intentionally or unintentionally assert incorrect knowledge to the Semantic Web. Intelligent applications have no ability to evaluate the correctness of such knowledge and will have it learnt just like other knowledge on the web. This could make a intelligent application "sick" and impair its ability in making correct decision.

5) Description integrity

As mentioned in section 11.2, there could be vulnerability if an unauthorized party makes RDF description to describe a resource which alters or contradicts with the meaning of the original description of that resource. For example, Aaron has a file on web and he makes a RDF description "This file is confidential". But no one can stop Bob to make another description for that file "This file is open to public." As these two descriptions forms a contradiction in concept, it may stop programs to work if both descriptions are received. How can we ensure the integrity of RDF describing a resource without stopping "anyone can say anything about anything"? The answer of this and for (4) is on the "trust" layer. For statement from A, we trust. From B, we don't. We will see how we can do this in the near future.

6) Ontology maintenance

Knowledge and concepts evolve from time to time. It is necessary to update old ontologies to reflect the up-to-date concept. The problem is how to maintain the compatibility between different versions of the same ontology, as well as how to maintain the compatibility between a new version and the instance data of an old version.

7) Semantic Web Services

Web Services (based on XML) is a hot topic today. Yet the development of standards for Semantic Web Services has already begun. The goal of Semantic Web Services is to achieve automated Web service discovery, execution, composition, monitoring and interoperation by software agents. A new ontology language DAML-S has been developed under the DAML family for Semantic Web Services development.

8) Scalability of Semantic Web application

The "My First Uni" application created in this project read and process only 16 RDF documents plus a few DAML documents and it takes a few seconds to build up a RDF model in the main memory.

In reality this implementation will not be practical for processing a huge number of RDF documents. We don't have that much memory to build a huge RDF model in it and we can't afford a couple of days to process the documents before the application can be used. This could also be a problem for intelligent agents that there would be too much knowledge to "learn" before an agent can make a correct decision. In another word, we have to avoid applications spending too much time and resources to process the whole web of processable documents before it can give a desired output.

9) Marketing issues

It is an interesting question why XML can be so popular in 5 years time but not RDF. It seems to be due to the complexity in developing Semantic Web documents. To develop a good RDF, you need a good ontology. To develop a good ontology, you need a domain expert and probably an ontology engineer. To have these people working together, you need more time and money. For typical companies, they will leave it until the rest of the world is using it.

However, Semantic Web never works if there is too little machine processable knowledge on the web. For research and development, it is hard to draw a conclusion if we don't have an enormous number of ontologies on the web as a test bed for experimenting software agents. Perhaps the Semantic Web community needs to push a little bit harder to foster the season of Semantic Web.

12. Conclusion

As a conclusion to this project, Semantic Web is, in no doubt, the future of World Wide Web. In this project, it is proven by a workable prototype that the web can be used as a world wide database and knowledge base for machine processing in Semantic Web. It is a feasible, beneficial, on-demand technology that enables automation of information processing and intelligent services on web which can never be done before. It is an evolution of web, that the minimalist design or least power principle is conserved: "make the simple things simple, and the complex things possible".[17]

For the Semantic Web at this moment, we can integrate and process distributed heterogeneous web data by RDF, we can define the formal semantics of the concepts used in the RDF by DAML+OIL so that software agents can reason about these concepts. For the Semantic Web tomorrow, we expect a web of trust and proof can be made possible. However Semantic Web is regarded as a vision. It is not something that can be totally done in the future because the requirement for machine processing on web information is also moving forward while we are advancing our web technology.

Therefore, Semantic Web will continuously evolve beyond our expectation.

[17] http://www.w3.org/Talks/2002/01/10-video/

13. Project review

13.1 Merits of this project

As to make this a high caliber research-based Final Year Project, the following are accomplished:

Research:

- Extensive research on the concept and theory of this state-of-the-art subject
- Three new languages for Semantic Web document, XML, RDF and DAML+OIL are studied in depth and used for prototype development

Contribution of original work:

- The first heavy ontology (ontology defined with restriction and axiom) for the domain of Hong Kong university admission is developed
- A new search algorithm "explore by class" is invented for selective document crawling on Semantic Web
- The prototype developed is one of the very few Semantic Web applications exist at this moment, which can serve as an example for future Semantic Web application development.

Experiment and critical review:

- Some new possibilities brought by Semantic Web are proved to be feasible by experiments
- In depth reviews include comprehensive comparison of RDF and XML, major changes in web evolution and possible challenges to Semantic Web at the current stage

13.2 Further enhancement

Due to the limited time frame of this project, the followings are left as future enhancements:

- Incorporate a reasoner server to the prototype application to enable inference capability
- Enhance the search algorithm to make an iterative “explore by class” search
- Take in depth study on the mathematical theory behind DAML+OIL

13.3 Project experience

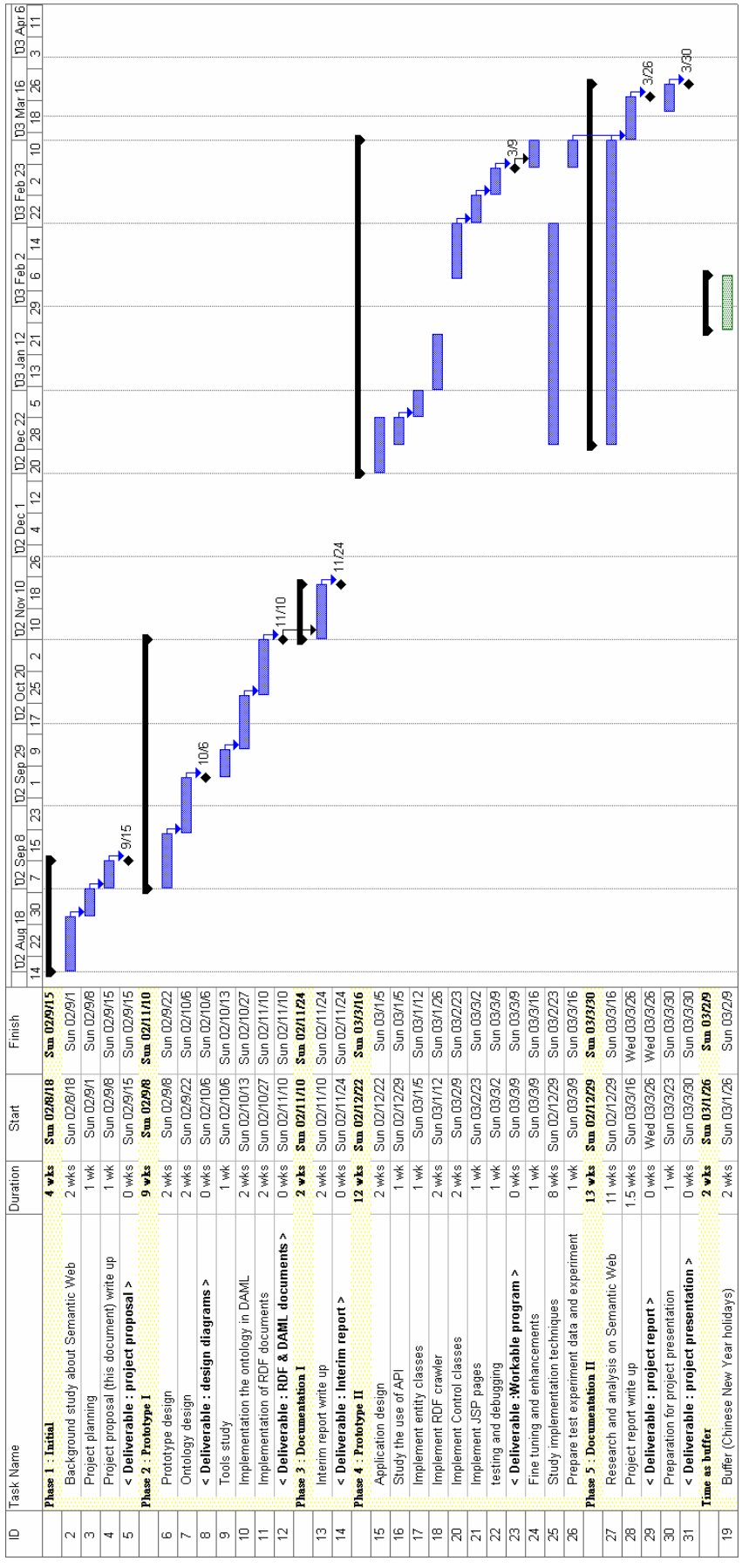
Personally this project is a great challenge to me. I remember at the time I submitted the project proposal, all I knew about Semantic Web were a name of the future web and the concept of searching on web by meaning, not by string matching. These limited information captured my interest but they did tell me how to do it.

I was so frustrated in the first a few months as I knew nothing beyond HTML! Since Semantic Web is such a novel topic, no classmates and tutors can help me as they didn’t even heard of Semantic Web. I started to train my self-study ability by reading piles of articles, specifications, tutorials from web and learnt everything from scratch. I am not sure if it was a magic, five months after my initial proposal, I started weaving the Semantic Web using new languages, new editors, new tools and new programming API. It turns out that without taking lessons in university, one can still master advance topic by just browsing the web. (of course, this takes time !)

Somebody had made an effort to make knowledge sharing possible in articles and documents on the web to let me learn about Semantic Web. Now perhaps it is my turn to put my effort to contribute to knowledge sharing for machine in the coming age of Semantic Web.

13.4 Project schedule

ID	Task Name	Duration	Start	Finish
	Phase 1 : Initial	**4 wks**	**Sun 02/8/18**	**Sun 02/9/15**
2	Background study about Semantic Web	2 wks	Sun 02/8/18	Sun 02/9/1
3	Project planning	1 wk	Sun 02/9/1	Sun 02/9/8
4	Project proposal (this document) write up	1 wk	Sun 02/9/8	Sun 02/9/15
5	**< Deliverable : project proposal >**	0 wks	Sun 02/9/15	Sun 02/9/15
	Phase 2 : Prototype I	**9 wks**	**Sun 02/9/8**	**Sun 02/11/10**
6	Prototype design	2 wks	Sun 02/9/8	Sun 02/9/22
7	Ontology design	2 wks	Sun 02/9/22	Sun 02/10/6
8	**< Deliverable : design diagrams >**	0 wks	Sun 02/10/6	Sun 02/10/6
9	Tools study	1 wk	Sun 02/10/6	Sun 02/10/13
10	Implementation the ontology in DAML	2 wks	Sun 02/10/13	Sun 02/10/27
11	Implementation of RDF documents	2 wks	Sun 02/10/27	Sun 02/11/10
12	**< Deliverable : RDF & DAML documents >**	0 wks	Sun 02/11/10	Sun 02/11/10
	Phase 3 : Documentation I	**2 wks**	**Sun 02/11/10**	**Sun 02/11/24**
13	Interim report write up	2 wks	Sun 02/11/10	Sun 02/11/24
14	**< Deliverable : Interim report >**	0 wks	Sun 02/11/24	Sun 02/11/24
	Phase 4 : Prototype II	**12 wks**	**Sun 02/12/22**	**Sun 03/3/16**
15	Application design	2 wks	Sun 02/12/22	Sun 03/1/5
16	Study the use of API	1 wk	Sun 02/12/29	Sun 03/1/5
17	Implement entity classes	1 wk	Sun 03/1/5	Sun 03/1/12
18	Implement RDF crawler	2 wks	Sun 03/1/12	Sun 03/1/26
20	Implement Control classes	2 wks	Sun 03/2/9	Sun 03/2/23
21	Implement JSP pages	1 wk	Sun 03/2/23	Sun 03/3/2
22	testing and debugging	1 wk	Sun 03/3/2	Sun 03/3/9
23	**< Deliverable :Workable program >**	0 wks	Sun 03/3/9	Sun 03/3/9
24	Fine tuning and enhancements	1 wk	Sun 03/3/9	Sun 03/3/16
25	Study implementation techniques	8 wks	Sun 02/12/29	Sun 03/2/23
26	Prepare test experiment data and experiment	1 wk	Sun 03/3/9	Sun 03/3/16
	Phase 5 : Documentation II	**13 wks**	**Sun 02/12/29**	**Sun 03/3/30**
27	Research and analysis on Semantic Web	11 wks	Sun 02/12/29	Sun 03/3/16
28	Project report write up	1.5 wks	Sun 03/3/16	Wed 03/3/26
29	**< Deliverable : project report >**	0 wks	Wed 03/3/26	Wed 03/3/26
30	Preparation for project presentation	1 wk	Sun 03/3/23	Sun 03/3/30
31	**< Deliverable : project presentation >**	0 wks	Sun 03/3/30	Sun 03/3/30
	Time as buffer	**2 wks**	**Sun 03/1/26**	**Sun 03/2/9**
19	Buffer (Chinese New Year holidays)	2 wks	Sun 03/1/26	Sun 03/2/9



Reference

■ Cited references in the report:

[1]. Tim Berner-Lee, *Semantic Web Road map* http://www.w3.org/DesignIssues/Semantic.html

[2]. Ibid

[3]. Tim Berner-Lee, *The Semantic Web,* http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

[4]. Tim Berner-Lee, World Wide Web Consortium, *Semantic Web - XML2000* http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

[5]. Unicode, *What is Unicode?* http://www.unicode.org/unicode/standard/WhatIsUnicode.html

[6]. World Wide Web Consortium, *Naming and Addressing: URIs, URLs, ...* , http://www.w3.org/Addressing/

[7]. Norman Walsh, *What is XML?*, October 03, 1998 http://www.xml.com/pub/a/98/10/guide1.html#AEN58

[8]. World Wide Web Consortium 14-January-1999, *Namespaces in XML*, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[9]. W3C Recommendation 22 February 1999, *Resource Description Framework (RDF) Model and Syntax Specification*, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[10]. Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, Ljiljana Stojanovic, Nenad Stojanovic, Rudi Studer, Gerd Stumme, York Sure, Julien Tane, Raphael Volz, Valentin Zacharias, *KAON - Towards a large scale Semantic Web*, http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/dexa2002.pdf

[11]. Deborah McGuinness, *DAML-ONT and OIL,* http://www.daml.org/2000/10/daml-oil

[12]. Vienna-Austria, *Integrating Heterogeneous Resources for Web-based Application*, http://www.ifs.tuwien.ac.at/ifs/lehre/skripten/metadaten/metadaten_ifs.ppt

[13]. Daml.org, *Language Feature Comparison*, http://www.daml.org/language/features.html

[14]. Supra, note 10

[15]. Natalya F. Noy, Deborah L. McGuinness, *Ontology Development 101: A Guide to Creating Your First Ontology*, http://protege.stanford.edu/publications/ontology_development/ontology101.html

[16]. Semaview, *XML and RDF Illustrated*, http://www.semaview.com/d/RDFandXML.pdf

[17]. Tim Berners-Lee, Eric Miller, *The Semantic Web*, http://www.w3.org/Talks/2002/01/10-video/

[18]. Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens, *OilEd: A Reason-able Ontology Editor for the Semantic Web*, http://potato.cs.man.ac.uk/slides/oiled-ki2001.ppt

[19]. HP Labs, *Jena Semantic Web Toolkit*, http://www.hpl.hp.com/semweb/jena.htm

[20]. HP Labs, *RDQL - RDF Data Query Language,* http://www.hpl.hp.com/semweb/rdql.htm

Reference books:

1. Tim Berners-Lee, Mark Fischetti, Michael L. Dertouzos, *Weaving the Web The original Design and Ultimate Destiny of the World Wide Web by its Inventor,* Harper SanFrancisco, September 22, 1999, ISBN: B00006B5XJ

2. Johan Hjelm, *Creating the Semantic Web with RDF: Professional Developer's Guide (With CD-ROM)*, John Wiley & Sons, May 11, 2001, ISBN: 0471402591

3. Ian Horrocks, James Hendler (eds.), *Proceedings of the semantic Web-ISWC 2002 : First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002*

4. Bhavani Thuraisingham, Bhavani Thuraisingha, *XML Databases and the Semantic Web*, CRC Press, 1st edition, March 27, 2002, ISBN: 0849310318

5. Rob Pooley, Perdita Stevens, *Using UML: Software Engineering with Objects and Components,* Addison Wesley, 1999, ISBN: 0201360675

6. Henry Hamburger, Dana Richards, *Logic and Language Models for Computer Science*, Prentice Hall, 2002, ISBN: 0130654876

Reference web resources:

1. W3C, *RDF Vocabulary Description Language 1.0: RDF Schema,* W3C Working Draft 12 November 2002, http://www.w3.org/TR/2002/WD-rdf-schema-20020430/
2. Tim Berner-Lee, *what the semantic web can represent,* http://www.w3.org/DesignIssues/RDFnot.html
3. Tim Bray, *What is RDF?,* http://www.xml.com/pub/a/2001/01/24/rdf.html
4. The Diffuse Project, *Guide to the Semantic Web*, http://www.diffuse.org/semantic-web.html
5. Renato Iannella , *An Idiot's Guide to the Resource Description Framework,* http://archive.dstc.edu.au/RDU/reports/RDF-Idiot/
6. Ronald Bourret, *XML.com: Namespace Myths Exploded,* http://www.xml.com/pub/2000/03/08/namespaces/
7. Roxane Ouellet, Uche Ogbuji, *XML.com: Introduction to DAML: Part I,* http://www.xml.com/pub/a/2002/01/30/daml1.html
8. Roxane Ouellet, Uche Ogbuji, *XML.com: Introduction to DAML: Part II,* http://www.xml.com/pub/a/2002/03/13/daml.html
9. Roxane Ouellet, Uche Ogbuji, *XML.com: DAML Reference,* http://www.xml.com/pub/a/2002/05/01/damlref.html

10. Daml.org, *Annotated DAML+OIL (March 2001) Ontology Markup,* http://www.daml.org/2001/03/daml+oil-walkthru
11. The Semantic Web Community Portal, *Inference Engines for the Semantic Web*, http://www.semanticweb.org/inference.html
12. Jim Hendler, *Knowledge Acquisition on the Semantic Web: A scruffy perspective*, http://www.cs.umd.edu/~hendler/presentations/EKAW02.ppt
13. Eric Prudhommeaux, *Semantic Web,* http://www.cs.umbc.edu/671/fall01/class-notes/Semantic-Web.ppt
14. Hui Cai, Professor Wesley Chu, *The Semantic Web and Ontology,* http://www.cs.ucla.edu/~wwc/course/cs244a/sem-ont.ppt
15. Ian Horrocks, *DAML+OIL: an Ontology Language for the Semantic Web,* http://www.cs.man.ac.uk/~horrocks/Slides/daml-pi-feb-01.pdf
16. Natalya F. Noy, *Ontology Engineering for the Semantic Web and Beyond,* http://www.semanticweb.org/SWWS/program/tutorials/tutorial1/OntologyEngineering.ppt
17. On-To-Knowledge project, *OIL ontology inference and exchange,* http://www.ontoknowledge.org/oil/downl/ecai00.ppt
18. Ian Horrocks, *SIG2: Ontology Language Standards – WebOnt Briefing,* http://www.cs.man.ac.uk/~horrocks/OntoWeb/SIG/OWL-overview.ppt
19. John Akinyele, *Multi-agent system for web services,* http://www.cs.umbc.edu/~cost/src/f01/www/presentations/akinyele.ppt
20. Deborah McGuinness, *The SemanticWeb (State of the art and implications for language processing)*, http://www.ncar.ucar.edu/cyber/mcguinness.ppt
21. Lee McCluskey, *The Semantic Web: Implications for Future Intelligent Systems,* http://scom.hud.ac.uk/scomtlm/seminars/semweb.ppt
22. Ontobroker, *Suggestions on RDF Authoring (Pragmatic Approach),* http://ontobroker.semanticweb.org/rdfcrawl/help/rdf.html
23. Ontobroker, *Usage of the RDF Crawler of 2000-11-27,* http://ontobroker.semanticweb.org/rdfcrawl/help/
24. Sean Bechhofer, *OilEd 3.4 Manual*, http://oiled.man.ac.uk/docs/Manual.pdf
25. Angus Roberts, *An Introduction to OilEd,* http://oiled.man.ac.uk/tutorial/
26. Hewlett Packard Laboratories, Bristol, *Jena release 1.6.0 documentation*, http://www.hpl.hp.com/semweb/doc/
27. KRSS group, Description-Logic Knowledge Representation System Specification, http://dl.kr.org/krss-spec.ps
28. Ian Horrocks, *Reasoning with Expressive Description Logics: Theory and Practice,* http://www.cs.man.ac.uk/~horrocks/Slides/leipzig.pdf

Appendix

A. Helpful tools in weaving the Semantic Web

An ontology editor – OilEd

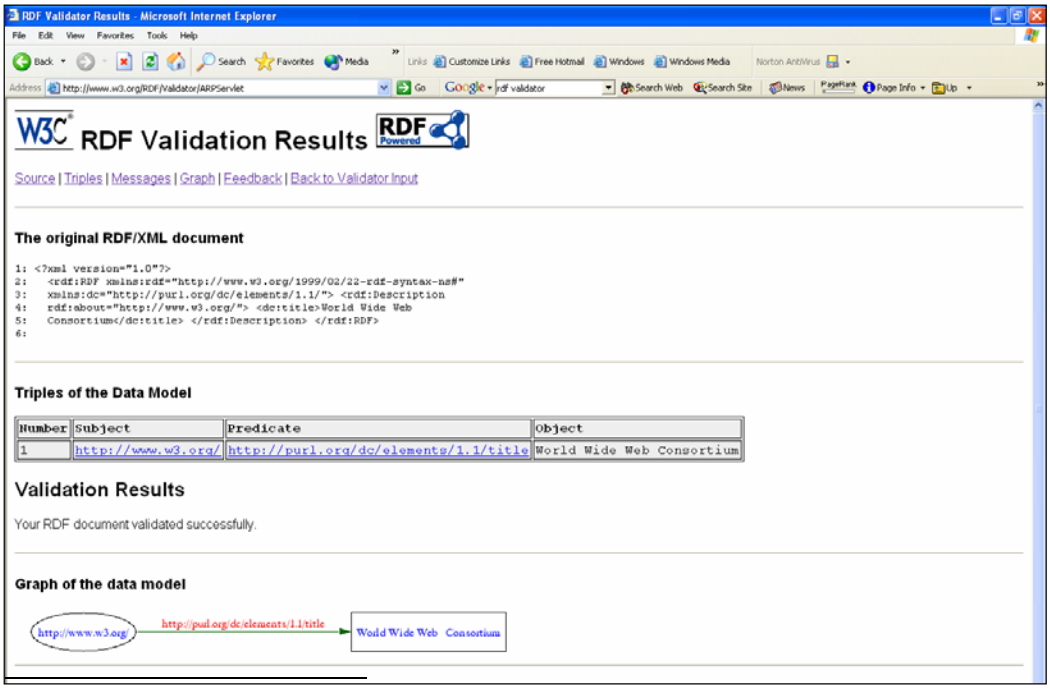
Web site: http://oiled.man.ac.uk/

OilEd is an ontology editor for DAML+OIL document editing. It is the best ontology editor supporting DAML+OIL standard that I have found so far. It supports arbitrary class expressions as slot fillers, primitive and defined classes, various slot constraint types, axioms, concrete type expressions and most importantly, it can connect to a reasoning engine for ontology reasoning and classification.[18]

A RDF validator and model visualizer by W3C

Web site: http://www.w3.org/RDF/Validator/

A great online tool to validate the syntax of a RDF document and visualize the RDF model represented.



[18] http://potato.cs.man.ac.uk/slides/oiled-ki2001.ppt

A methodology – Object Oriented Methodology

For Semantic Web application development, Object Oriented methodology is found to be very applicable. During system design, designer simply defines entity classes and its properties which resemble the classes and properties from the ontology and then finish the job by defining the corresponding accessor methods. Once property values from the RDF document are loaded into object of the corresponding entity class, programming becomes straight forward.

A programming API – Jena

Web site: http://www.hpl.hp.com/semweb/jena.htm

Without a robust RDF API with comprehensive functionality of Jena, this project could probably be a failure. Jena is a java API for manipulating RDF models. Its features include:[19]

- statement centric methods for manipulating an RDF model as a set of RDF triples
- resource centric methods for manipulating an RDF model as a set of resources with properties
- cascading method calls for more convenient programming
- built in support for RDF containers - bag, alt and seq
- enhanced resources - the application can extend the behaviour of resources
- integrated parsers (ARP and David Megginson's RDFFilter)

The current jena release integrates a number of additional components:

- ARP parser compliant with latest working group recommendations
- integrated query language (RDQL)
- support for storing DAML ontologies in a model
- persistent storage module based on Berkeley DB
- support for persisting jena models in relational databases
- open architecture supporting other storage implementations

[19] http://www.hpl.hp.com/semweb/jena.htm

A RDF model query language – RDQL

Web site: http://www.hpl.hp.com/semweb/rdql.htm

RDQL is an implementation of an SQL-like query language for RDF. It treats RDF as data and provides query with triple patterns and constraints over a single RDF model.[20] The RDQL query engine is included in the Jena RDF API package. It features:

- SQL-like language for retrieving sets of values
- Java query engine for Jena models
- Command line support for exploring data sets

A Java web programming IDE – JBuilder 7

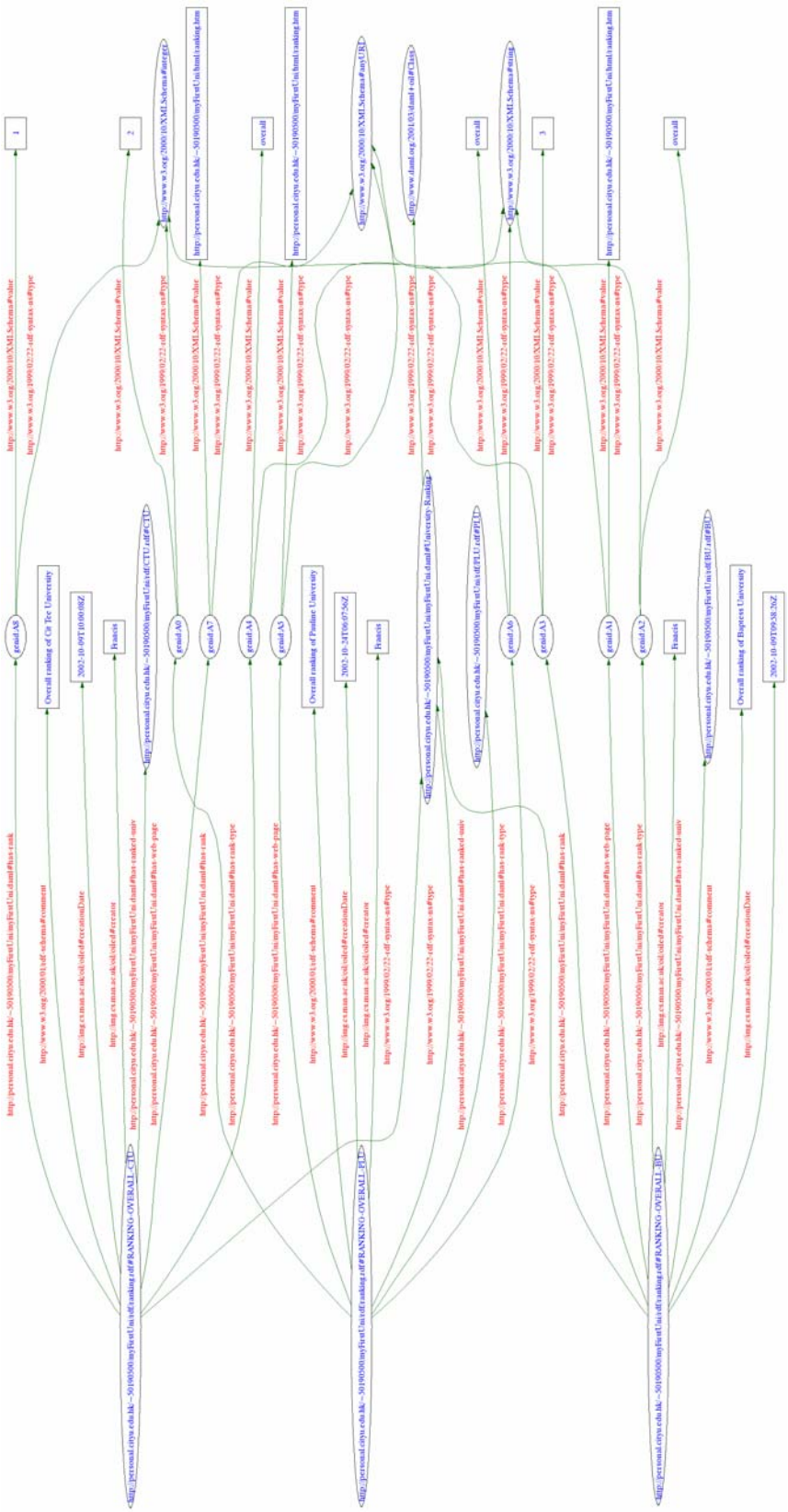
Web site: http://www.borland.com/jbuilder/

As it is so much pain to develop web application in a Notepad, a Java programming IDE is used in this project. JBuilder is an all-in-one solution for web application development. It is an IDE, a web server, a web browser, a GUI designer and a UML visualizer. Tutorials and documentations shipped with JBuilder are so helpful for beginner users. Plus the CodeInsight™ and ErrorInsight™ technologies, web programming in Java becomes so cost-effective.

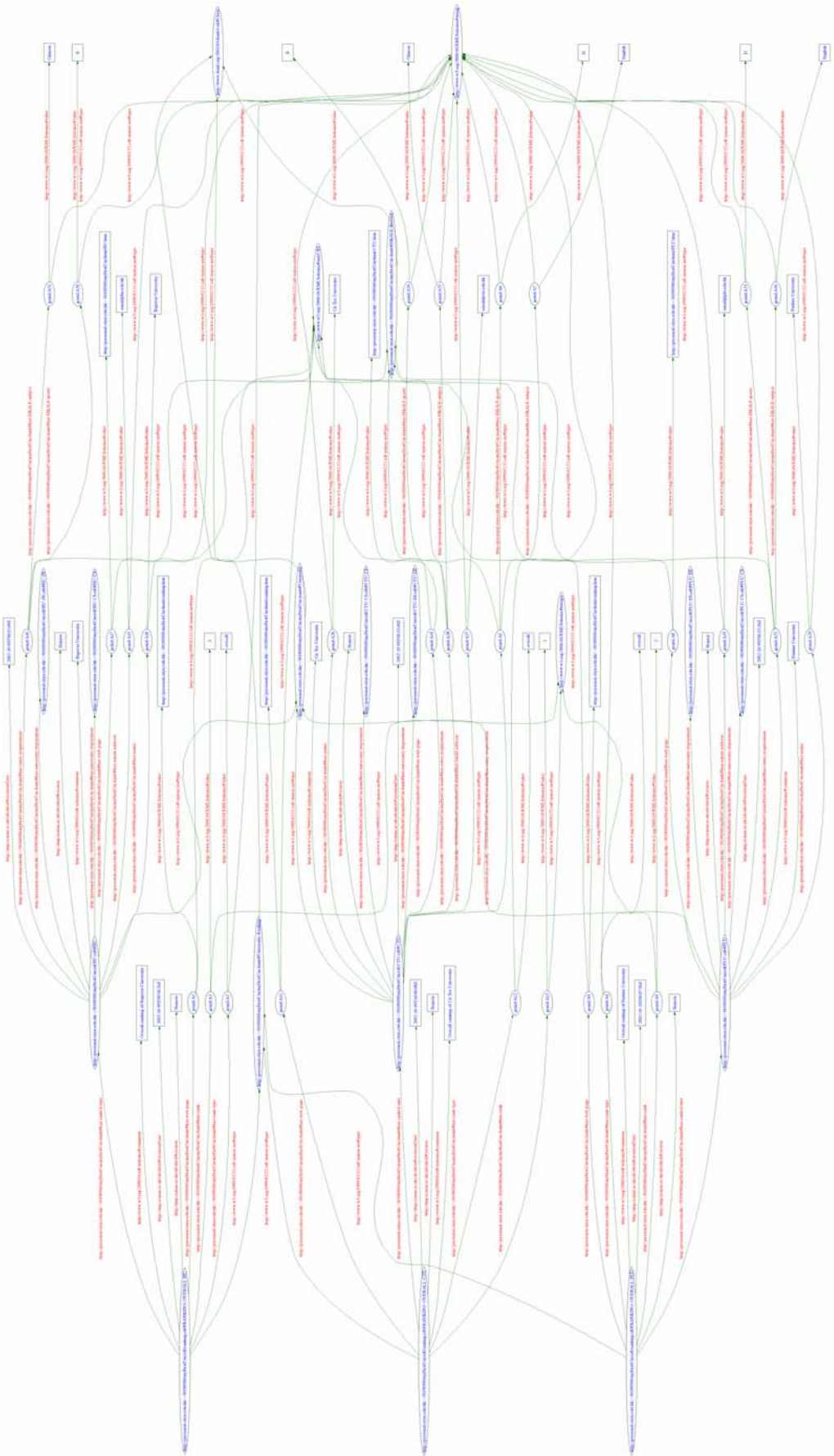
[20] http://www.hpl.hp.com/semweb/rdql.htm

B. Directed graph of "My First Uni" RDF model

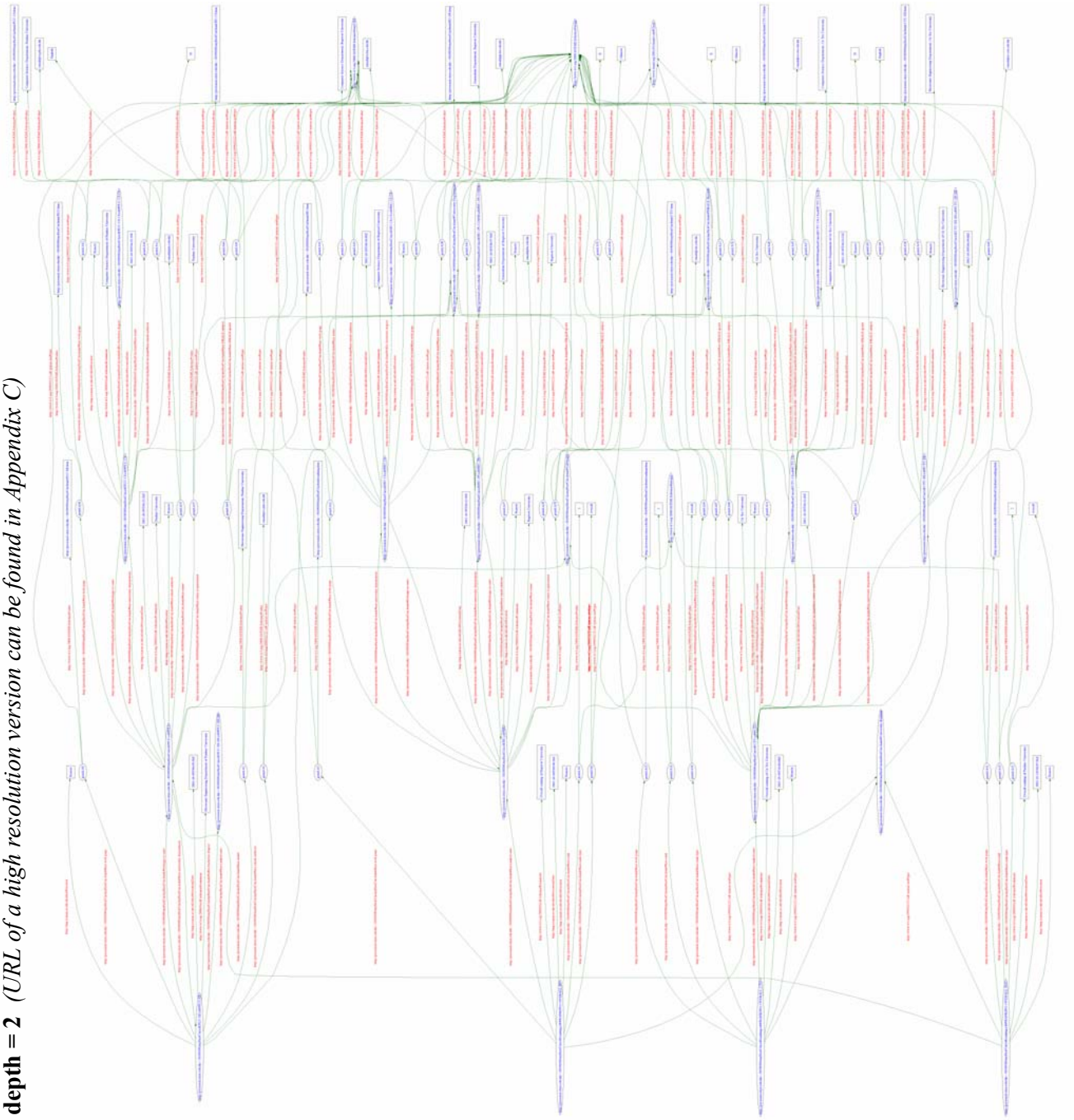
B1. model with depth = 0 *(URL of a high resolution version can be found in Appendix C)*

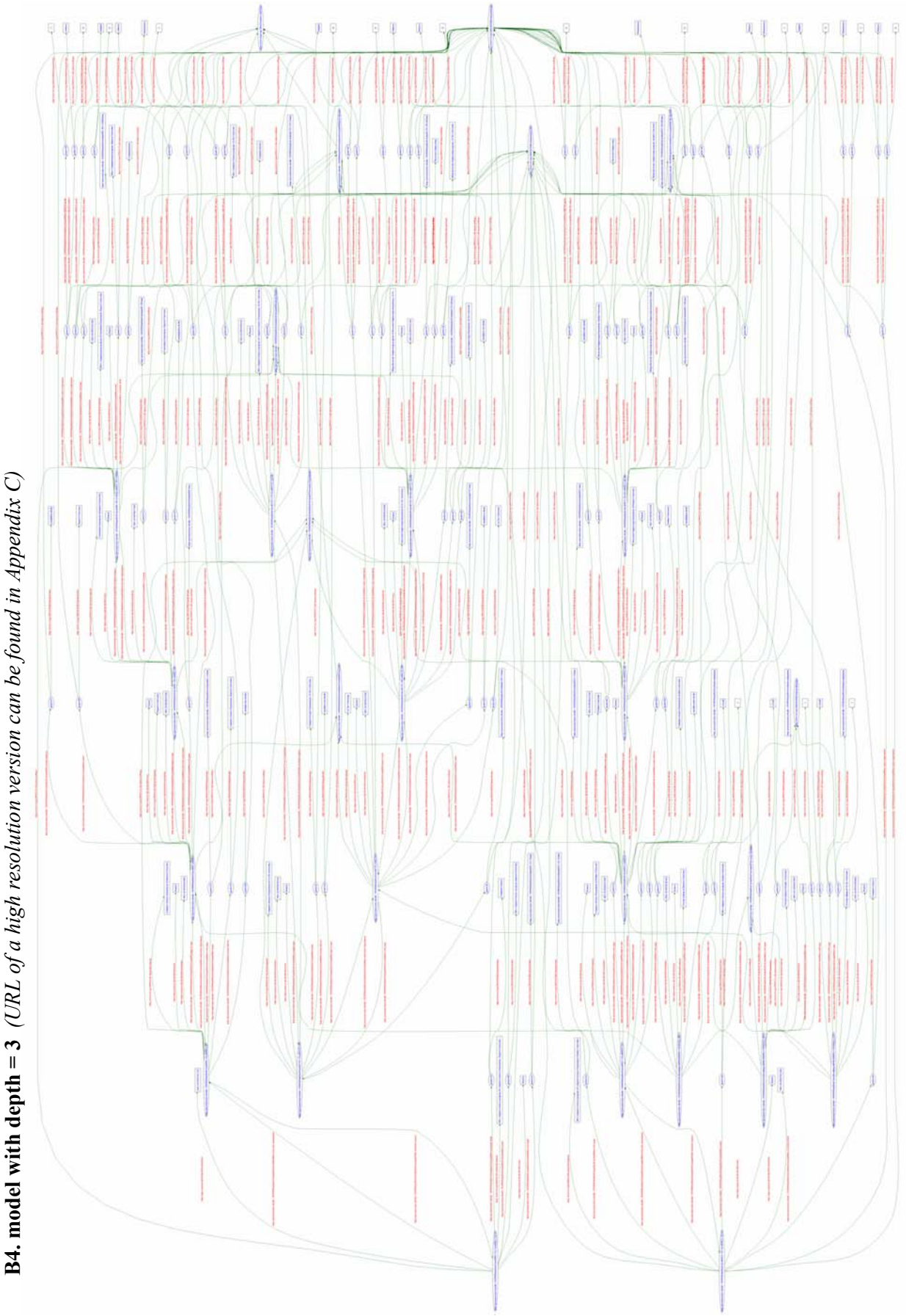


B2. model with depth = 1 *(URL of a high resolution version can be found in Appendix C)*



B3. model with depth = 2 *(URL of a high resolution version can be found in Appendix C)*





B4. model with depth = 3 *(URL of a high resolution version can be found in Appendix C)*

C. URL of all materials of this project

URL of this project report:

http://franciskwg.tripod.com/SemanticWeb/finalReport.pdf

URL of the interim report:

http://franciskwg.tripod.com/SemanticWeb/interimReport.pdf

URL of "My First Uni" source code:

http://personal.cityu.edu.hk/~50190500/myFirstUni/src/myFirstUniSource.zip

URL of "My First Uni" ontology:

http://personal.cityu.edu.hk/~50190500/myFirstUni/myFirstUni.daml

URL of labeled directed graph of "My First Uni" basic RDF set:

http://personal.cityu.edu.hk/~50190500/myFirstUni/img/level0.png

http://personal.cityu.edu.hk/~50190500/myFirstUni/img/level1.png

http://personal.cityu.edu.hk/~50190500/myFirstUni/img/level2.png

http://personal.cityu.edu.hk/~50190500/myFirstUni/img/level3.png

Directory URL of "My First Uni" basic RDF set:

http://personal.cityu.edu.hk/~50190500/myFirstUni/rdf/

(please concatenate the RDF file name to the end of this directory URL for browsing)

Directory URL of "My First Uni" experiment 1 RDF set:

http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario1/rdf/

(please concatenate the RDF file name to the end of this directory URL for browsing)

Directory URL of "My First Uni" experiment 2 RDF set:

http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario2/rdf/

(please concatenate the RDF file name to the end of this directory URL for browsing)

Directory URL of "My First Uni" experiment 3 RDF set:

http://personal.cityu.edu.hk/~50190500/myFirstUni/scenario3/rdf/

(please concatenate the RDF file name to the end of this directory URL for browsing)

URL of "My First Uni" experiment 3 related ontologies:

http://personal.cityu.edu.hk/~50190500/myFirstUni/funding.daml

http://personal.cityu.edu.hk/~50190500/myFirstUni/project.daml

URL of "My First Uni" experiment 4 related ontology:

http://personal.cityu.edu.hk/~50190500/myFirstUni/exp4.daml