Abstract

Semantic Web, a state-of-the-art concept of web architecture proposed by W3C, means to change the current web into a machine processable web. This project aims to investigate the values and new capabilities of Semantic Web. The short-comings of traditional web technologies, as well as the literatures of Semantic Web technology are studied. To realize its capabilities, a major part of the project is to work out a Semantic Web prototype and carrying out experiments on this prototype. The prototype built in this project automates web information processing and provides intelligent advice to students in selecting the most appropriate universities and subjects to apply to. Without Semantic Web technology, students need to manually browse through pages and pages of information in Web sites of different Universities to find a list of universities, departments and programs that they qualify to apply for. Our prototype demonstrates that Semantic Web can automate and streamline this whole process. All the students need to do is enter their qualifications and a list will be generated automatically. The prototype uses a shared ontology based on RDF and DAML+OIL to describe admissions requirements of universities in Hong Kong. One of the objectives of this research project is to design such an ontology and to test its limits by encoding admission requirements for several of our local universities into our prototype. This report also presents a novel "explore by class" search algorithm I developed to optimize search performances in this university Semantic Web problem domain comparing to traditional web crawling techniques.

1. Introduction

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 [1], 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. In this project, we study different approaches in implementing a web application. We explore how such a software agent might be developed to retrieve precise university admission requirements from different university websites, match them with high school student's qualifications and present a filtered list of candidate universities for a student to consider applying to. The main objective of this project is to assess the feasibility of using Semantic Web as well as analyze its usefulness for such a generic application. To achieve these objectives, I designed and developed a Semantic Web-based “My First Uni” (MFU) application.

2. The Problem

High school graduates in Hong Kong who are continuing their studies in a university will have to submit a list of potential bachelor degree programs that they are interested in to the Joint University Programmes Admissions System (JUPAS), which is a semi-government organization that is responsible for
centralizing and coordinating all admissions-related processing for the 8 tertiary institutes in Hong Kong.

Graduating high school students can submit a prioritized list of up to 25 different programs (potentially from different universities) to JUPAS for automatic computerized admissions allocation [3]. To produce this list of 25 programs, students have to consider:

- **Unique university requirements**
  Each university may have their own unique set of admission requirements. For example, an applicant may need to satisfy certain language requirements in one university but not in another.

- **Unique program requirements**
  Each bachelor degree program within even the same university may have additional and different requirements. For example, different programs may require qualification in related subjects of study.

- **University ranking**
  Students may have perceived the ranking of the university based on reputation of the universities from various sources.

- **Interest in fields of study**
  Students will also need to consider which bachelor programs will best suit their interests and abilities.

To do this selection manually, students have to laboriously search and study numerous web pages of every university, every department, and every bachelor degree program as well as perform research on websites related to university rankings. This task is extremely time-consuming and may lead to days of online research through over a hundred web pages before a student can produce the required prioritized list of 25 programs that he/she thinks is best for him/her to study as well as will give him/her the best chance of being accepted.

My “My First Uni” (MFU) application is a research experiment to investigate whether Semantic Web can be used to help streamline this program selection process.

Before I explain the Semantic Web approach, I first describe how one might approach this problem without Semantic Web.

### 3. Alternative 1 – Centralized Database

One way to solve the Hong Kong university admissions problem is to use a centralized database that collects all information related to universities, departments, programs and their admission requirements. This information will need to be collected and consolidated by a third party, a service provider, who will manage this centralized database as well as develop Web applications to give advice to high school students.

The problem with this approach is in the constant need to ensure all information is up-to-date and accurate. This can either be done manually, which will be too labor intensive and error prone, or through sophisticated middleware and integration technology to integrate data from the 8 universities, which will be quite impossible as most of the data reside as free-form textual content.

### 4. Alternative 2 – XML

An alternative is to use XML for data exchange. If all universities provide admission requirements information in XML form, the application program can easily process information directly from the Web rather than retrieving them from a centralized database. In this way, control over the data remains with the information provider and the access of information is opened to everyone.

But there are many limitations. To make this possible all the XML documents and the application program have to agree to use a common XML structure and tags (hence a common XML schema [4]) [5]. 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 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. [6]

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).

5. Alternative 3 – Semantic Web with RDF & DAML+OIL

The ideal technical features we are looking for (for our MFU application) can be summarized as follows:

- Information on the Web should be machine-processable to allow Web applications to service user questions by manipulating multiple sources of information from the Web and consolidate them to generate an answer.
- Control and maintenance of information should remain with the information owners.
- Access of information should be opened to public.
- Document storing machine-processable information should be given great flexibility in how it is structured.
- These documents should be discoverable and classifiable in the dynamic and opened Web.
- All information should have well-defined semantics resided within itself so that it is inter-processable by any machine process.

Building on top of XML, RDF [7-9] 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 [10] 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 [11-16] 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 [17] and prediction.

Alternative 3 is the approach I took in designing the MFU application. The following Sections explain the MFU prototype design in two parts – the Semantic Web content and then the application.

6. Designing the Semantic Web Content

The first step is to define an appropriate ontology to use to express knowledge of universities, departments, programs, and their admission requirements. Ontology is
an explicit specification of a conceptualization. It is a
description (like a formal specification of a program) of
the concepts and relationships among these concepts
[18]. On the representation level, our ontology is
represented in DAML+OIL and the instances are
represented in RDF. The ontology was developed
following the design strategy from “Ontology
Development 101: A Guide to Creating Your First
Ontology” [19] which involves the following steps:

1. Determine domain and scope of ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define properties of classes
6. Define facets of the properties
7. Create instances

In terms of consistency and reusability, ontology
should be reused if there are existing ones that fit the
purpose. Unfortunately, there are none that deals with
university admissions in the DAML+OIL ontology library
[20]. Therefore the MFU ontology had to be built from
scratch. However, there are some ontologies which are
about university research and publication. We have
based our MFU ontology on these works.

The following is a sample of some of the terms
included in the first draft of the MFU ontology:

<table>
<thead>
<tr>
<th>University</th>
<th>Dept</th>
<th>Bachelor degree</th>
<th>Student</th>
<th>HK High school graduate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKCEE</td>
<td>HKALE</td>
<td>Email address</td>
<td>Ranking</td>
<td>Ranking type</td>
</tr>
<tr>
<td>HKCEE</td>
<td>subjects</td>
<td>HKCEE grade</td>
<td>HKALE grade</td>
<td>Entry requirement</td>
</tr>
</tbody>
</table>

Figure 1 is the MFU ontology class hierarchy
developed using ontology editor OilEd [21].

Classes marked “#1” are classes defined in our new
MFU ontology. Some of them inherit super-classes
marked “#2” developed by Stanford University
(http://www.ksl.stanford.edu/projects/DAML/ksl-daml-des
c.daml), which is an existing ontology modeling university
publication. In MFU these super classes are added with
new properties. 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.

Similar to the Object-Oriented classes, each of the
ontology classes has its own properties. There is no
“method” since we are composing the N-Triple/RDF
Triple, the Subject–Property- Property Value relationship.

Figure 2 illustrates the MFU class design using UML
notations [22].

The Class relationships are shown in Figure 3. In the
diagram, 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.

![Class Generalization in MFU Ontology](http://www.ksl.stanford.edu/projects/DAML/ksl-daml-desc.daml#)

**Fig 2. Class Generalization in MFU Ontology**

After the completion of class and property definitions, the ontology can be used to create instances which are represented in RDF. The following shows two simplified examples out of the many RDF files used in the MFU prototype. The property values of these instances and the mapping between the instances and the RDF documents are shown in tabular format. Each table represents one RDF document.

<table>
<thead>
<tr>
<th>ranking.rdf</th>
<th>Class</th>
<th>Instance</th>
<th>Property</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RANKING-OVERALL-BU</td>
<td>has-rank</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RANKING-OVERALL-CTU</td>
<td>has-rank</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RANKING-OVERALL-PLU</td>
<td>has-rank</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BU.rdf</th>
<th>Class</th>
<th>Instance</th>
<th>Property</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>University</td>
<td>has-name</td>
<td>Baptess University</td>
<td></td>
</tr>
<tr>
<td></td>
<td>University</td>
<td>has-email-address</td>
<td><a href="mailto:email@bu.edu.hk">email@bu.edu.hk</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>University</td>
<td>has-web-page</td>
<td>BU.htm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>University</td>
<td>has-university-department</td>
<td>BU-CS.rdf#BU-CS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>University</td>
<td>has-university-department</td>
<td>BU-JN.rdf#BU-JN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>University</td>
<td>has-entry-requirement</td>
<td>BU.rdf#genid001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HKALE-Record</td>
<td>genid</td>
<td>001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HKALE-Record</td>
<td>has-HKALE-subject</td>
<td>Chinese</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HKALE-Record</td>
<td>has-HKALE-grade</td>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

7. The MFU Application

The MFU application uses a typical Web-based architecture using JSP. Figure 4 is an overview of the MFU application architecture. It has two special control classes and some API modules from Jena RDF Tool.
Advisor – 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 and the list of recommended degree programs. It takes user inputs as well as query results and then stores them using corresponding entity classes.

Model Builder – used to retrieve Semantic Web documents (.rdf and .daml) and build a RDF model using retrieved RDF documents [23]. Initially, the Model Builder starts crawling from a root URI given by its “Advisor”.

Jena API [24] – a Java library that supports retrieval, parsing and model manipulation for RDF. It implements the RDF model as a set of RDF Triples and provides resource centric methods for manipulating an RDF model as a set of resources with properties.

Jena RDQL Engine [25] – 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 – 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.

MFU firstly allows a student user to enter his/her academic qualification through a simple to use interface shown in Figure 5. After searching, processing and prioritizing with special matching formula, a resulting list of programs that a student might be interested in and is qualified to apply for will be listed as in Figure 6.
8. 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.

The “explore” action by the Model Builder, which is a search process for RDF documents, has two implementations “Explore by depth” & “Explore by class”.

8.1 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 7. Explore by depth up to depth 2](image)

8.2 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 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 treasures. 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 [26-27] 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, in the MFU ontology, a cardinality constraint for property “has-university-department”, that “If University has University-Department, that University must have at least
one University-Department”. Formal expression for this in Description Logic [28-29] is:

- \( \geq 1 \text{ has-university-department} \)

The term “class restriction” refers to one or more conditions that are always true for a class. For example, in the MFU ontology, a class restriction to the class “University” is that “for any University, it must have at least one has-university-department property”. The formal expression for this in Description Logic is:

- University \( \subseteq \geq 1 \text{ has-university-department} \)

The term “range of a property” refers to the object in a subject-predicate-object triple. For example, in the MFU ontology, the range for the property “has-university-department” is that the value of “has-university-department” must be of class “University-Department”. The formal expression for this in Description Logic is:

- has-university-department.University-Department

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

\[ A = A_1 \cup A_2, \]

\[ A_1 \subseteq (\geq N \text{ANY} \cdot C), \]

\[ A_2 \subseteq (\geq N \text{ANY} \cdot C), \]

\[ N \geq 1, \]

\[ \text{ANY} = \text{any properties} \]

\[ C = \text{wanted class (for association level 1 search)} \]

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 association 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 searches 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.
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.

9. Experiments

The MFU prototype allowed us to experiment with the new possibilities brought by our ontology and RDF files under different usage scenarios. For example, in:

(1) Adaptation to newly available resources
Information on web pages are being added, deleted or updated from time to time. In this experiment, new bachelor degree RDF and additional university ranking RDF documents are added and linked for an adaptability test. The result shows that the MFU application can discover and make use of the newly available web documents automatically.

(2) Anyone can say anything about anything anywhere
With Semantic Web, web authors are not bounded to any given set of vocabularies or any ontology. In this experiment, some Bachelor degree RDF documents using multiple ontologies are added and linked. The RDF content describing about a bachelor degree is split into two RDF documents to test the adaptability when arbitrary RDF descriptions about something is distributed in more than one RDF document. The result shows that the MFU application can adapt and make use of information from these RDF documents. This is a significant advantage of RDF over XML as XML vocabularies are bounded by the XML schema and a XML tree must be complete within one XML document.

(3) Intelligent web crawling
Semantic Web crawler is more intelligent than the traditional crawler as searching in Semantic Web is a semantic matching process rather than a syntactic pattern matching. Crawler can even be more intelligent with the “explore by class” algorithm that I developed. By comparing the performance of the Model Builder of MFU using “explore by depth” and “explore by class” algorithms, it is for sure that “explore by class” is a more cost-effective choice as long as the two assumptions of this algorithm are applicable.

(4) Basic inference
A modified version of the MFU ontology is done for a simple inference test with the class definitions shown in the following table:

<table>
<thead>
<tr>
<th>Class</th>
<th>Class property</th>
<th>Class restrictions (informal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associate-Degree</td>
<td>SubclassOf</td>
<td>&quot;Associate-Degree is only offered by a Technical-Institute&quot;</td>
</tr>
</tbody>
</table>
The FaCT description logic classifier (SHIQ reasoner) that integrated with OilEd editor is used as the reasoner for this experiment [30]. The ontology is opened in OilEd and it is classified by the reasoner to decide the class memberships. Initially, all classes are not related to each other in the hierarchy. After running the reasoner on this ontology, superclass-subclass relationships are determined as shown in Figure 11.

### 10. Conclusion

#### 10.1 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.

<table>
<thead>
<tr>
<th>Bachelor-Degree</th>
<th>SameClassAs</th>
<th>&quot;Bachelor-Degree is only offered by a University-Department&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualified-Degree</td>
<td>SameClassAs</td>
<td>&quot;Qualified-Degree is only offered by a University-Department&quot;</td>
</tr>
<tr>
<td>Hons-Degree</td>
<td>SameClassAs</td>
<td>&quot;Hons-Degree has exactly one honor type of Honors&quot;</td>
</tr>
<tr>
<td>HK-Bach-Degree</td>
<td>SubclassOf</td>
<td>&quot;HK-Bach-Degree is only offered by a University-Department&quot;</td>
</tr>
<tr>
<td>Technical-Institute</td>
<td>SubclassOf</td>
<td>--</td>
</tr>
<tr>
<td>University-Department</td>
<td>SubclassOf</td>
<td>--</td>
</tr>
<tr>
<td>Honors</td>
<td>SubclassOf</td>
<td>--</td>
</tr>
</tbody>
</table>

*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→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↔Q.

Fig 11. Class hierarchy by inference
10.2 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.

While RDF means to wide spread the knowledge enabling highly intelligent and automated processing, 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.

10.3 Semantic Web today and tomorrow

Semantic Web is, in no doubt, the future of World Wide Web. In this project, we have shown, using a workable MFU prototype, that Semantic Web enables us to use the Web as a world wide data and knowledge base for machine processing. It is a feasible, beneficial, on-demand technology that enables automation of information processing and intelligent services on web that were not possible before.

With current state-of-the art in Semantic Web, we can integrate and process distributed heterogeneous web data using RDF and define formal semantics of concepts used in the RDF with DAML+OIL so that software agents can reason about these concepts. For the Semantic Web of tomorrow, we expect a web of trust and proof [31] 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 [32].
12. References