DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF AN EXAMPLE SEMANTIC WEB BASED SYSTEM

by

Berna PAKALIN GENÇOĞLU

May, 2007

İZMİR

DESIGN AND IMPLEMENTATION OF AN EXAMPLE SEMANTIC WEB BASED SYSTEM

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program

by
Berna PAKALIN GENÇOĞLU

May, 2007
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN AND IMPLEMENTATION OF AN EXAMPLE SEMANTIC WEB BASED SYSTEM"** completed by **BERNA PAKALIN GENÇOĞLU** under supervision of **PROF. DR. ALP R. KUT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Alp R. KUT

Supervisor

Yrd. Doç. Dr. Reyat YILMAZ

(Jury Member)

Doç. Dr. Yalçın ÇEBİ

(Jury Member)

Prof.Dr. Cahit HELVACI

Director

Graduate School of Natural and Applied Sciences

Fen Bilimleri Enstitüsü

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere thanks to my supervisor Prof. Dr. Alp R. KUT. Without his support, continuous encouragement, valuable guidance and advising it would be impossible for me to complete the work of this thesis.

Special thanks go to my best friend, my husband, Barış, for his positive motivation, endless patience, and support.

Last but not least, I would like to extend my sincere thanks to my parents, Özver and Ergin, for their complete support throughout my life.

Berna PAKALIN GENÇOĞLU

DESIGN AND IMPLEMENTATION OF AN EXAMPLE SEMANTIC WEB BASED SYSTEM

ABSTRACT

The Semantic Web is envisioned as the future of the current Web. The Semantic Web proposes the mark-up of Web content so that machines can automatically process, interpret and integrate information available on the Web. This thesis presents a comprehensive study of the Semantic Web technology. It explains the vision, architecture, theory, fundamental components and applications of the Semantic Web.

In this thesis, an example prototype Semantic Web based system for an academic department of a university has been designed and implemented. An ontology, modelling the knowledge structure of the application domain has been developed and specified in a machine-processable format using Web Ontology Language (OWL). Protege has been used as the ontology editor. The system has been implemented using Java programming language. The architecture of the system contains three major layers, namely user interface layer, internal layer, and knowledge layer. The knowledge base of the system which consists of the ontology and the instance data has been processed, manipulated and queried by Jena API and presented to the user in HTML interface layout through a conventional Web browser.

Keywords : Semantic Web, Ontology.

ANLAMSAL WEB TABANLI ÖRNEK BİR SİSTEM TASARIMI VE GERÇEKLEŞTİRİMİ

ÖZ

Anlamsal Web mevcut Web'in geleceği olarak betimlenmektedir. Anlamsal Web, Web içeriğinin işaretlenmesini ve böylece makinaların Web üzerindeki bilgiyi otomatik işleyebilmesini, yorumlayabilmesini ve birleştirilebilmesini önermektedir. Bu tez, Anlamsal Web teknolojisi ile ilgili kapsamlı bir çalışma sunar. Anlamsal Web'in vizyonunu, mimarisini, teorisini, temel bileşenlerini ve uygulamalarını açıklar.

Bu tez çalışmasında, bir üniversitenin akademik bir departmanı için Anlamsal Web tabanlı bir örnek prototip sistem tasarlanmış ve gerçekleştirilmiştir. Uygulama alanının bilgi yapısını modelleyen bir ontoloji geliştirilmiş ve bu geliştirilen ontoloji OWL Web ontoloji dili kullanılarak makinaların işleyebileceği formatta belirtilmiştir. Ontoloji editörü olarak Protege kullanılmıştır. Sistem Java programlama dili kullanılarak geliştirilmiştir. Geliştirilen sistemin mimarisi, kullanıcı arabirim katmanı, iç katman ve bilgi katmanı olmak üzere üç katmandan oluşmaktadır. Ontoloji ve örnek veriden oluşan bilgi tabanı Jena API ile işlenmekte, sorgulanmakta ve kullanıcıya HTML arayüzü ile geleneksel Web tarayıcısı aracılığıyla sunulmaktadır.

Anahtar sözcükler : Anlamsal Web, Ontoloji.

CONTENTS

Page

CHAPTER ONE

INTRODUCTION

1.1 Progress Towards the Semantic Web Vision

The World Wide Web (WWW) has brought exciting new possibilities for information access and has changed the way people communicate with each other. Currently on the Web, vast amount of information is stored, disseminated, retrieved and also business is conducted. The Semantic Web technology promises to dramatically improve the current Web and its use. For the Semantic Web, the word *semantic* states that the meaning of data on the Web can be understood—not just by people, but also by machines. The Semantic Web is all about making the Web meaningful, understandable, and machine-processable. The Semantic Web is a vision for the future of the Web in which information is organized with semantic mark-up to allow machines to automatically access, process and integrate information available on the Web.

The Semantic Web is still under its elementary development, various introductions and researches are conducted. There has been intensive research activity in this field, and also applications based on Semantic Web technology are already beginning to appear. This thesis is about this emerging, new, challenging and thrilling application area.

1.2 Aim of the Thesis

The next generation of Web technology, which promoted as the “Semantic Web”, aims at moving the current Web from being human understandable to being both human and machine understandable. The idea is to make Web content more accessible to machines by using semantic mark-up.

In this thesis, it is aimed to bring contributions of researchers from academics, industry, and different communities together to study, understand, and explore the theory, tools, languages and applications of the Semantic Web. In addition, it is aimed to apply the Semantic Web technology to build an example Semantic Web based system.

This thesis provides insight into the Semantic Web by first explaining and introducing its function, its technology, its fundamental components, its language and tools and then by projecting to a future vision on its applications with a prototype example system.

1.3 Organization of the Thesis

The content of the thesis is constructed as follows:

In Chapter 2, an overview of the Semantic Web is presented. This chapter indicates problems with the current state of the Web and states why Semantic Web is needed and what Semantic Web technology will enable. It describes fundamental components and layered architecture of the Semantic Web and introduces the key Semantic Web technologies.

In Chapter 3, knowledge representation formalisms for the Semantic Web is presented. This chapter particularly focuses on two important concepts: Ontologies and Description Logics. It describes what an ontology is, why and how an ontology is developed and how it is used to achieve semantic interoperability. It also provides an overview on ontology engineering for the Semantic Web, covering the languages, and some existing tools. This chapter also provides an introduction to Description Logics as a formalism for representing knowledge in Semantic Web. It first addresses the relationship between Description Logics and earlier semantic network and frame systems and then introduces the basic features of Description Logics languages and related reasoning techniques.

In Chapter 4, implementation of an example Semantic Web based system is presented. This chapter includes the design, architecture and characteristics of the system which employs much of the technology described in this thesis.

The thesis completes with a summary of this research and some proposals for future work.

CHAPTER TWO

SEMANTIC WEB TECHNOLOGY

2.1 The Semantic Web Overview

The World Wide Web (WWW) has gained a considerable attention and success due to its distributed nature and its ability which allows users to easily access and share information. Currently on the Web there is a vast repository of information comprising of billions of documents. At the moment, however, the extensive information found on the Web is designed preliminarily for human consumption. The content is marked up in such a way that data is displayed in a meaningful way to human. As such the Web content is not meant for machine consumption. HTML is the predominant language in which web pages are written. It relies on a set of predefined tags which control appearance of a Web page. It carries only information of how the content should be displayed. It provides neither the structure nor the semantics of information.

The current Web has the potential of being a primary source for storing and accessing data of all types, including textual data, audio and video data. However, as the volume of information increases exponentially, it becomes difficult for users to find, access, retrieve and analyze all relevant web pages given a topic of interest. The most common way to find information on the Web is by using search engines such as Google and Yahoo. Although these search engines are indispensable, they have limitations stem from keyword-based searching.

Due to these limitations of the current Web technology, research efforts have been put towards to enhance the current Web in such a way that it will allow the information to be processed more efficiently. Tim Berners-Lee, who invented the WWW in 1989 and is now the director of the World Wide Web Consortium (W3C), introduced the concept of Semantic Web as the future of the current Web. The Semantic Web is not a separate Web but an extension of the current one, in which

information is given well-defined meaning, better enabling computers and people to work in cooperation (Berners-Lee, Hendler, & Lassila, 2001).

The Semantic Web stands for a vision in which machines as well as people can access, find, process, and use data over the Web to accomplish useful goals for users. Semantic Web technology will enable intelligent software agents perform tasks on behalf of users such as making searches, locating meaningful information resources on the Web and combining them in meaningful way, and deciding on the next step to take. Semantic Web aims at giving machines much better access to information resources.

2.2 The Semantic Web Layered Architecture

The development of the Semantic Web is being promoted by the World Wide Web Consortium (W3C). W3C is an international consortium that founded in 1994 for developing guidelines, technologies, specifications, and standards for the Web.

Figure 2.1 shows the layered architecture of the Semantic Web as it is presented by the W3C (World Wide Web Consortium [W3C], 2003).

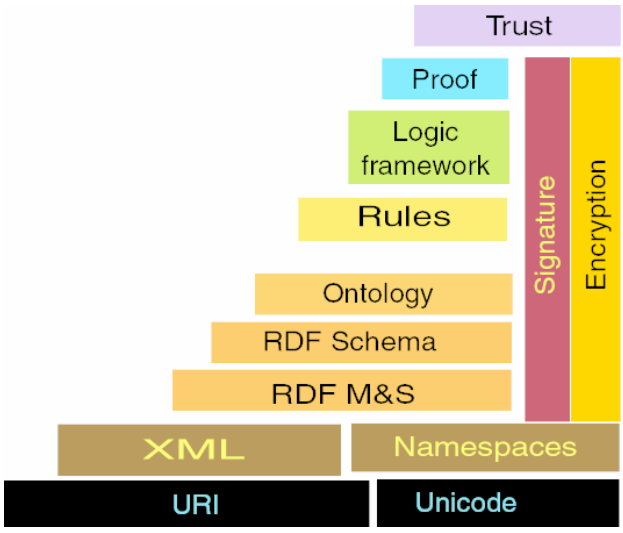

Figure 2.1 The Semantic Web layered architecture.

The base of the architecture is the concept of Uniform Resource Identifier (URI) and a universal character set (Unicode). The Unicode and URI layers make sure that an international character set is used and web resources are identified, respectively.

Above the URI and Unicode, Extensible Mark-up Language (XML) and Namespaces are layered. XML is a simple general purpose mark-up language that used to create semantically rich Web documents. Namespaces are used to allow unambiguous use of several vocabularies.

On top of XML and Namespaces, Resource Description Framework (RDF) is layered. RDF is used to express web resources and relations between them in the form of triples: subject, predicate and object.

Above the RDF layer, RDF Schema is a framework that provides a means to describe vocabularies of RDF resources. Vocabularies are needed for describing types of things and properties.

Ontology layer expands RDF Schema and allow the representation of more complex semantics.

Higher levels of the Semantic Web, which are the layers of Rules, Logic Framework, Proof and Trust, are still subject to ongoing research and have not yet well defined.

Each layer is discussed in more detail in the following subsections.

2.2.1 Uniform Resource Identifier (URI) and Unicode

In order to describe things, a way to reference or identify them is needed first. URI provides a naming scheme and is used to identify or name a web resource on the Web. The named resources can be any abstract or physical thing that has an identity. A URI consists of string of characters and conforms to a particular syntax. URI

became a standart with the invention of the Web, and it serves as the foundation not only for the current Web but also for the Semantic Web.

Unicode is a universal character encoding standart that designed to support the worldwide interchange, processing, and display of the data in any language. It is a 16 bits character set standard that covers all the characters and symbols of the modern written languages. Each character has exactly one encoding and has to be presented by a fixed width in bits. It is universal and unique and provides an unambiguous encoding of the content of plain text, covering all languages in the world.

2.2.2 Extensible Mark-up Language (XML)

XML is a subset of SGML, the Standard Generalized Mark-up Language [ISO8879] and has been designed for ease of implementation and for interoperability with both SGML and HTML (Bray, Paoli, Sperberg-McQueen, Maler & Yergeau, 2006). The XML specification was approved as W3C Recommendation in February 1998.

Like an HTML document, an XML document consists of a collection of tagged elements. However unlike HTML which relies on a set of pre-defined tags for formatting and displaying information on a Web page, XML allows users to write structured Web documents with a user defined vocabulary. Any number of XML tags can be defined and nested to any level of complexity. XML provides a way of encoding both data and meta-data.

An XML document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated by explicit mark-up. Comments begin with <!-- and end with -->. They are ignored by XML parsers. Processing instructions define procedural attachments and provide a mechanism for passing information.

The structure XML uses is a hierarchy or tree structure. An XML document is composed of a properly nested set of tagged elements. Each element contains a start tag (<tag name>), an end tag (</tag name>), and the content between these two tags. For instance, using a tag name “credits”, a tagged element may represent the course credits of 4+2 as follows: <credits>4+2</credits>.

Content may be text, or other elements, or nothing. Name-value pairs (such as <course id=”CSE101”>) attached to an element is called attributes.

Figure 2.2 shows an example of an XML document for a course in an academic department of a university. The id, name, credits, description, homepage and lecturer and assistants’ information of a course are formatted with XML.

```
<?xml version="1.0"?>
<course id="CSE101">
    <courseName>Algorithms and Programming</courseName>
    <courseCredits>4+2</courseCredits>
    <courseDescription>Problem solving, algorithm development</courseDescription>
    <courseHomepage>http://courses.myuniv.edu.tr/CSE101</courseHomepage>
    <lecturer>
        <name>John Black</name><title>Associate Professor</title>
    </lecturer>
    <assistants>
        <assistant>
            <assistantName>Benjamin Ass</assistantName>
            <assistantEmail>benjamin@myuniv.edu.tr</assistantEmail>
        </assistant>
        <assistant>
            <assistantName>Yancy Walt</assistantName>
            <assistantEmail>yancy@myuniv.edu.tr </assistantEmail>
        </assistant>
    </assistants>
</course>
```

Figure 2.2 An example XML document.

The XML specification defines two levels of conformance for XML documents: well formed and valid. An XML document is *well-formed*, if it is syntactically correct, such as having one unique root element and each element has an opening and a corresponding closing tag. An XML document is *valid*, if it is well formed and conforms to a particular set of user-defined rules, or a schema. A schema defines the elements, attributes, and the structure of an XML document. A schema language is used for expressing schemas formally. Two dominant XML schema languages are Document Type Definitions (DTD) and XML Schema.

DTD was the original approach used to validate XML documents. The purpose of DTD is to define the document structure with a list of legal elements. XML Schema provides a powerful method for defining the structure of XML documents.

2.2.2.1 XML Schema

XML schema is used for declaring the names, attributes of elements, and specifying the structure of XML documents. XML Schema became a W3C Recommendation in May 2001.

XML Schema is written in XML syntax. Figure 2.3 shows an example XML Schema document related to the XML document presented in the Figure 2.2.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="course" type="courseType"/>
  <xsd:complexType name="courseType">
     <xsd:sequence>
        <xsd:element name="courseName" type="xsd:string"/>
        <xsd:element name="courseCredits" type="xsd:string"/>
        <xsd:element name="courseDescription" type="xsd:string"/>
        <xsd:element name=" courseHomepage" type="xsd:string"/>
        <xsd:element name="lecturer" type="Lecturer"/>
        <xsd:element name="assistants" type="Assistants"/>
     </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
</xsd:complexType>
  <xsd:complexType name=" Lecturer">
    <xsd:sequence>
       <xsd:element name="name" type="xsd:string"/>
       <xsd:element name="title" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Assistants">
   <xsd:sequence>
     <xsd:element name="assistant" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:sequence>
        <xsd:element name="assistantName" type="xsd:string"/>
        <xsd:element name="assistantEmail" type="xsd:string"/>
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Figure 2.3 An example XML Schema document.

2.2.2.2 XML Namespace

An XML Namespace is a W3C standard for providing unique names for the elements and attributes in an XML document. XML namespace allows the combination of several vocabularies within a single document without ambiguity. Since each vocabulary is given a namespace, duplicate element and attribute names can be distinguished and used without a name conflict.

A namespace is declared using the reserved XML attribute *xmlns*. A namespace declaration has the following syntax:

```
xmlns:namespace-prefix="NamespaceURI"
```

2.2.3 Resource Description Framework (RDF)

RDF provides a standard for describing resources. A resource can be any thing that is identified with a URI. Making statements about resources provide descriptions of the resources and a basis for information representation. RDF has XML based syntax to make statements about resources in a machine-processable way. RDF became a W3C Recommendation in February 1999.

RDF is intended for situations in which information needs to be processed by applications, rather than being only displayed to people and it provides a common framework for expressing information so it can be exchanged between applications without loss of meaning (Manola & Miller, 2004).

RDF is based on the idea of expressing simple statements about resources, where each statement is a triple comprised of a subject, a predicate, and an object. A set of such triples is represented as a node-arc-node link as illustrated in Figure 2.4 (Klyne & Carroll, 2004).

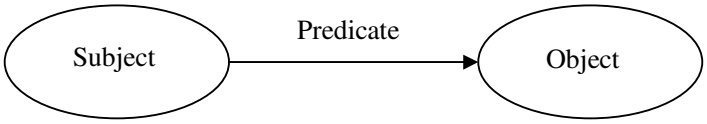

Figure 2.4 The basic RDF data model in RDF graph.

The subject is the resource being described. The predicate is a relation between the subject and the object. The object is the part that identifies the value of that property. RDF uses URI as the basis of its mechanism for identifying the subjects, predicates, and objects in statements. The object can either be a resource or a literal (constant value). In RDF Graph, URIs are shown as ellipses, while nodes that are literals are shown as boxes.

For instance, the following two sentences in English, say, the course “CSE101” is taught by “John Black” and the name of the course “CSE101” is “Algorithms and Programming” can be represented by an RDF graph as shown in Figure 2.5.

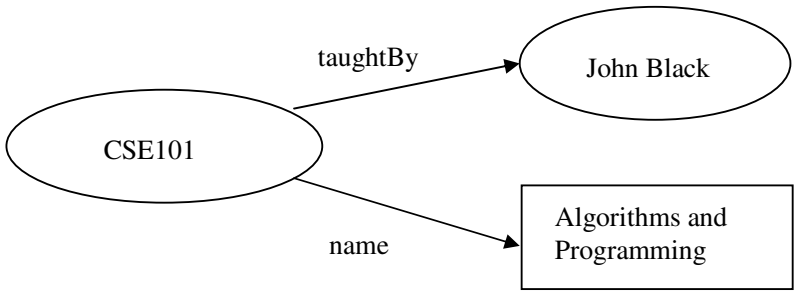

Figure 2.5 A sample RDF graph.

An alternative way of expressing the statements, called triples, is also used. In the triples notation, each statement in the graph is written as a simple triple of subject, predicate, and object, in that order. Each triple corresponds to a single arc in the graph. Sample statements can be written as an RDF triple as follows, with a period at the end:

<#CSE101> <#taughtBy> <#John Black>.

<#CSE101> <#name> Algorithms and Programming.

RDF also provides an XML-based syntax (called RDF/XML) to represent RDF statements in a machine-processable way. For the RDF/XML syntax definitions below and for further details we refer to Beckett (2004). Figure 2.6 is a small chunk of RDF in RDF/XML corresponding to the graph in Figure 2.5.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:course="http://www.myuniv.edu.tr/course#">
<rdf:Description rdf:about="CSE101">
  <course:taughtBy rdf:resource="John Black"></course:taughtBy>
  <course:name>Algorithms and Programming</course:name>
</rdf:Description>
</rdf:RDF>
```

Figure 2.6 A sample RDF document in RDF/XML.

Line 1, *<?xml version="1.0"?>*, is the XML declaration, which indicates that the following content is XML, and what version of XML it is. Line 2 begins with an *rdf:RDF* element. This indicates that the following XML content is intended to represent RDF. In Line 5, the *rdf:Description* start-tag indicates the start of a description of a resource, and goes on to identify the resource the statement is about (the subject of the statement) using the *rdf:about* attribute to specify the URIref of the subject resource.

2.2.4 RDF Schema

Although RDF provides a base for describing resources on the Web, it does not provide means for defining classes and properties. RDF Schema builds on RDF foundation to provide richer representation formalism. RDF Schema allows the definition of classes, subclasses, sub-properties, domain and range restrictions of properties. In other words, RDF Schema introduces basic ontological modelling primitives. RDF Schema became a W3C Recommendation in February 2004.

RDF's vocabulary description language, RDF Schema, is a semantic extension of RDF and it provides mechanisms for describing groups of related resources and the relationships between these resources (Brickley & Guha, 2004).

RDF schema is written in a formal language. The RDF Schema elements are marked by a specific namespace and normally bound to the prefix *rdfs:*. Following list gives an overview of some of the important terms in RDF Schema language. For full list of terms and for further details we refer to Brickley and Guha (2004).

- **rdfs:Resource :** This is the class of everything. All things described by RDF are instances of the class *rdfs:Resource*.

- **rdfs:Class :** It is used to indicate the definition of a resource class. A class corresponds to a general concept of type or category. RDF Schema allows classes to be organized in a hierarchical fashion with use of *rdfs:subClassOf.*

- **rdf:type :** Individual objects that belong to a class are referred to as instances of that class. *rdf:type* states that a resource is an instance of a class. In other words, it relates a resource to a class.

- **rdfs:subClassOf :** It used to state that one class is a a subclass of another. It allows declaring hierarchies of classes. If a class A is a subclass of a class B, then all instances of A will also be instances of B.

- **rdf:Property :** A RDF property defines a characteristic of resource classes. The element *rdfs:domain* defines to which resource classes a given property applies and the *rdfs:range* element describes the allowed range for the property value.

- **rdfs:domain :** It defines all resource classes which an RDF property applies to.

- **rdfs:range :** It defines the allowed set of values for a particular RDF property and is used to indicate that the values of a particular RDF property are instances of a designated resource class.

- **rdfs:subPropertyOf :** It denotes that one property is a subproperty of another. If a property P is a subproperty of property R, then all pairs of resources which are related by P are also related by R.

- **rdfs:Literal :** It is the class of literal values such as strings and integers. Property values such as textual strings are examples of RDF literals.

- **rdfs:label :** It provides a way of defining a human-readable name for the subject.

- **rdfs:comment :** It provides a way for writing comments.

- **rdfs:Container :** It is the super-class of the RDF containers, i.e. *rdf:Bag, rdf:Seq, rdf:Alt*.

- **rdf:Bag :** It is a subclass of *rdfs:Container*. It is used to indicate that the container is intended to be unordered. It is the class of unordered containers.

- **rdf:Seq :** It is a subclass of *rdfs:Container*. It is used to indicate that the numerical ordering of the container is intended to be significant. It is the class of ordered containers.

- **rdf:Alt :** It is a subclass of *rdfs:Container*. It is used to indicate that typical processing will be to select one of the members of the container. It is the class of containers of alternatives.

- **rdfs:seeAlso :** It is an instance of *rdf:Property* that is used to indicate that a resource might provide further information about the subject resource. A

triple of the form: *S rdfs:seeAlso O* states that the resource O may provide additional information about S.

- **rdfs:isDefinedBy :** It is an instance of *rdf:Property* and is used to indicate the definition of the subject resource. A triple of the form: *S rdfs:isDefinedBy O* states that the resource O defines S.

2.2.5 Ontology

RDF Schema is not very expressive since it has only limited modelling primitives. RDF Schema provides means to define vocabulary, structure and constraints for expressing metadata about web resources, however formal semantics for the primitives defined in RDF Schema are not provided, and the expressivity of these primitives is not enough for full-fledged ontological modelling and reasoning (Broekstra, Klein, Decker, Fensel, Horrocks, 2000). The limitations of the RDF Schema specification language has led to the efforts in designing more powerful ontology languages on top of RDF Schema. Many Ontology markup languages have been created in the context of the Semantic Web. Ontologies and related topics are covered in detail in the third chapter of this thesis.

2.2.6 Rules, Logic Framework, Proof and Trust

On top of ontologies, a logic framework and rules are needed to be specified for a fully functional Semantic Web. Logic framework enhances the ontology language further and provides a way to describe and exchange logic assertions over the Web. A rule language allows inference rules to be implemented which allow machines to draw conclusions and make deductions.

The proof layer executes the rules, traces and explains the steps of logical reasoning.

Lastly the trust layer provides evidence of the trustworthiness of data, services and agents. The trust layer clarifies whether and/or how much the information present on the Semantic Web can be trusted? In an open and distributed environment such as Web, there are no restrictions on the information being published. It is liability of human to accurately judge the validity and trust of the information resource. The information in Semantic Web is envisioned not only for human beings but also for machines. This implies machines are responsible to a certain degree for discerning the trust of information resource.

Digital signature and encryption provides a basis for the trust layer and envisioned to be used on all levels of the Semantic Web. Digital signature and encryption across the layers ensures the authentication, integrity, nonrepudiation, and confidentiality of data.

These higher levels of the Semantic Web are still under construction and so far only partly standardized.

CHAPTER THREE

KNOWLEDGE REPRESENTATION FORMALISMS FOR THE SEMANTIC WEB

3.1 Semantic Web and Knowledge Representation

Knowledge is used to achieve intelligent behaviour. To simulate a form of human intelligence in information systems, a way to store and manipulate knowledge in a formal way is needed. Representing knowledge in a manner as to facilitate inferencing i.e. drawing conclusions from knowledge is referred to as Knowledge Representation (KR). Knowledge Representation is one of the central and in some ways the most popular concepts in Artificial Intelligence (AI).

Semantic Web is recognized as a typical application of Knowledge Representation. Two important concepts are involved, namely, ontology and logics/formalism. Ontology allows defining sets of terms as well as relations exists between these terms in a structured and unambiguous way. The formalism makes an ontology machine interpretable so that ontologies can be put into computation and inference by machines.

Following subsections introduces some background material on ontologies and Description Logics and clarifies the key roles they play in the Semantic Web.

3.2 Ontologies

An ontology is a semantic foundation which is used to describe things, such as objects, tasks, events and relations exists among them in a machine-processable way. Ontologies play a key role in Semantic Web, since they give a formal model of the domain by explicitly describing concepts and their relations in some logical

formalism. An ontology abstracts the knowledge of some domain and gives a common understanding of the domain.

3.2.1 What is an Ontology?

The term ontology originates from Philosophy. In that context, ontology is a systematic explanation of existence. It tries to explain what exists by introducing a system of basic categories. Later, in the 1990s, the term ontology became a topic of interest in computer science, especially in Artificial Intelligence community for knowledge sharing and reuse. Many definitions about what ontology have been proposed in the literature.

Neches et al. (1991) defines ontology as the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.

Gruber (1993) defines ontology as an explicit specification of a conceptualisation. This definition became the most quoted in the literature. Based on Gruber's definition many definitions of ontology were proposed. The Gruber's definition was explained by Studer, Benjamins & Fensel (1998) as follows:

An ontology is a formal, explicit specification of a shared conceptualisation. A 'conceptualisation' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group (Studer, Benjamins & Fensel, 1998).

Guarino and Giaretta (1995) collected and classified the following seven definitions:

1. Ontology as a philosophical discipline
2. Ontology as an informal conceptual system
3. Ontology as a formal semantic account
4. Ontology as a specification of a conceptualization
5. Ontology as a representation of a conceptual system via a logical theory
    5.1. characterized by specific formal properties
    5.2. characterized only by its specific purposes
6. Ontology as the vocabulary used by a logical theory
7. Ontology as a (meta-level) specification of a logical theory

On that paper, Guarino and Giaretta (1995) proposed to interpret ontology as a logical theory which gives an explicit, partial account of a conceptualization.

An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base (Swartout, Patil, Knight & Russ, 1996).

3.2.2 Why Develop an Ontology?

Ontologies provide a common vocabulary of some area defining the meaning of basic terms and relations that hold among these terms in a formal way. Many disciplines now develop ontologies that experts can use and share information in their fields. Some of the important reasons for developing ontologies can be summarized as follows:

- to define a vocabulary within a domain.
- to provide a common understanding of information in a domain.
- to share common understanding of the structure of information among people or software agents.
- to enable machines to use the knowledge in some application.
- to enable reuse of domain knowledge.

3.2.3 Ontology Classifications

Classification of ontologies is important for a clear understanding of the various ways the ontologies can be used and the uses of ontologies in certain types of applications. Many classifications have been proposed by some researches taking several criteria into account as the formality of the language or the level of generality. Guarino (1998) classifies ontologies based on their level of generality, as shown in Figure 3.1.

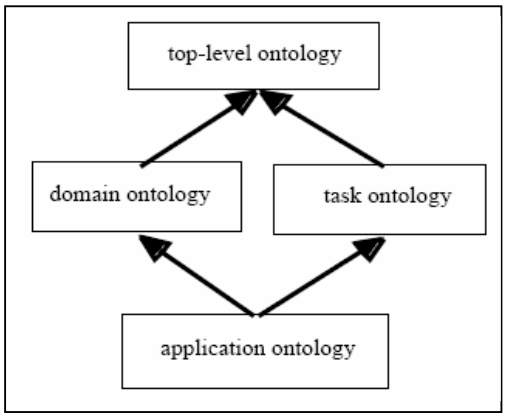

Figure 3.1 Kinds of ontologies, according to their level of generality.

- *Top-level ontologies* describe very general concepts like space, time, event, action, etc., which are independent of any particular domain. It seems therefore reasonable to build unified top-level ontologies serving large communities. A famous example project is Cyc.

- *Domain ontologies* describe the vocabulary related to a generic domain by specializing the concepts introduced in top-level ontologies. Some example domains are medicine, engineering, and chemistry.

- *Task ontologies* describe the vocabulary related to a generic task or activity, such as diagnosing, selling by specializing the concepts introduced in top-level ontologies.

- *Application ontologies* are specific ontologies that describe concepts depending on both the particular domain and the task.

Studer, Benjamins & Fensel (1998) makes a distinction between static knowledge and problem-solving knowledge and gives a slightly different classification for static knowledge ontologies as follows: (1) *Generic ontologies* (are similar to top-level ontologies), (2) *Domain ontologies* (3) *Application ontologies* and (4) *Representational ontologies* which are domain independent and describe representational entities without describing what they represent.

3.2.4 Ontology Engineering

During the last decade, increasing attention has been directed to ontologies in the fields of Knowledge Engineering, Artificial Intelligence and Computer Science. Ontologies are now widely used in many applications related to these areas such as knowledge management, education, natural language processing, e-Commerce, Bio-Informatics, Semantic Web, etc. There is much literature on what they are, how they can be implemented, what the tools are, and where they can be used inside applications. All these topics and activities are covered under the term "Ontological Engineering". Ontology engineering refers to the set of activities that concern the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suits and languages that support them (Gomez-Perez, Fernandez-Lopez & Corcho, 2004).

Ontologies have become a basic component of Semantic Web applications since they describe a domain and enable processing and sharing of knowledge. In general, ontologies contain descriptions of important concepts in a domain, crucial properties of concepts and restrictions on the value or cardinality of the property, as well as a set of axioms with each defined by itself or in relation to another property. The building of a Semantic Web application is essentially relies on ontologies.

Following sections provide an overview on ontology engineering for the Semantic Web, covering ontology development process, the status of the languages, and some existing tools.

3.2.5 Ontology Development

Ontologies are used to represent the background knowledge that is needed by the application. Currently there is no standard ontology building methodology. Different methodologies are developed based on project characteristics. Noy & McGuinness (2001) states that there is no one "correct" way or methodology for developing ontologies and offers a very simple one possible process based on iterative design that can be used in developing ontologies. The main sequence of the steps is listed as follows:

- Determine the domain and scope of the ontology
- Consider re-using existing ontologies
- Enumerate important terms in the ontology
- Define the classes and class hierarchy
- Define the properties of classes – slots
- Define the facets of the slots
- Create instances

Determine scope

Scope can be determined based on the use to which the ontology will be put, and on the future extensions that are already anticipated. A useful technique for determining the scope of an ontology is by devising some questions. Basic questions to be answered at this stage are: What is the domain that the ontology will cover? For what we are going to use the ontology? For what types of questions should the ontology provide answers? Who will use and maintain the ontology?

Consider re-using

With the spreading deployment of the Semantic Web, many ontologies are already available and can be used instead of starting from scratch. So it is always worth checking if an existing ontology can be used or refined to suit the needs of the ontology being developed.

Enumerate terms

A list of all the relevant terms that are expected to appear in the ontology is written down without worrying about the overlap between concepts or relations among the terms, or whether the terms should be classes or properties in their own right.

Define taxonomy

Once the relevant terms are identified, terms are organized in a taxonomic hierarchy. Opinions differ on whether it is more efficient to do this in a top-down or a bottom-up approach. *Top-down approach* defines first the broadest concepts in the domain and then defines in turn more and more specific concepts. A *bottom-up approach* goes the other way around, starting with the most specific concepts and the subsequent super concepts. A *combination* of these two approaches can also be applied by defining the most obvious concepts and then both specifying and generalizing other classes around them.

Define properties

Once classes and class hierarchy are defined, properties are attached to the classes. A property should be attached to the highest class in the hierarchy to which it applies since each subclass of the class inherits this property. While attaching properties to the classes, it makes sense to immediately define the domain and range of these properties.

Define facets

Previously defined properties are enriched with facets by describing the value type, allowed values, the number of values (cardinality) and other features of the values a property (slot) can take.

Create instances

Finally the ontology is populated with instances. Instances represent the objects (individuals) in the domain of interest.

3.2.6 Ontology Languages

Ontology languages are quite essential and needed for describing a model into concrete, computer serializable files. Ontology languages provide some forms of logics and formal modelling means considering the strictness of expressing and ease of computation. An ontology language must fulfil three important requirements (Fensel, van Harmelen, Horrocks, McGuinness & Patel-Schneider, 2001):

- It must be highly intuitive to the human user. Given the current success of the frame-based and object-oriented modelling paradigm, an ontology should have a frame-like look and feel.
- It must have a well-defined formal semantics with established reasoning properties to ensure completeness, correctness, and efficiency.
- It must have a proper link with existing Web languages like XML and RDF to ensure interoperability.

Many Ontology markup languages have been created in the context of the Semantic Web. Figure 3.2 shows the stack of these ontology markup languages: Simple HTML Ontology Extension (SHOE), Ontology Exchange Language (XOL), Ontology Markup Language (OML), RDF Schema, Ontology Inference Layer (OIL), DARPA Agent Markup Language+OIL (DAML+OIL), and Web Ontology Language (OWL).

All are based on the syntax of existing markup languages such as HTML and XML. Comprehensive explanation and comparison of these languages, except the latest OWL we refer to Gomez-Perez & Corcho (2002).

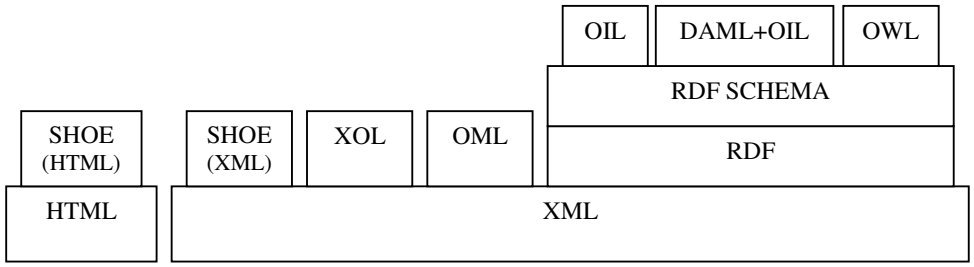

Figure 3.2 Language stack for the Semantic Web.

SHOE, which developed at the University of Maryland, appeared as the first ontology markup language. It was previously designed as an extension of HTML. Later its syntax was adapted to XML.

XOL was designed by the US bioinformatics community to fulfill the representational needs of experts in bioinformatics.

OML, which developed at the University of Washington, is partially based on SHOE. In fact, it was first considered as an XML serialization of SHOE. Therefore, OML and SHOE share many similar features.

OIL, DAML+OIL, and OWL have been developed as extentions to RDF Shema. RDF Schema and OWL are W3C proposed standart languages for constructing Semantic Web applications.

The following subsections describes the languages OIL, DAML+OIL and OWL which are based on RDF Schema.

3.2.6.1 Ontology Inference Layer (OIL)

OIL is a Web-based representation and inference layer for ontologies, which combines the widely used modelling primitives from frame-based languages with the formal semantics and reasoning services provided by Description Logics (Fensel et al., 2000). OIL has been proposed as an extension of RDF Schema. OIL has RDF Schema based syntax. OIL was developed using a layered approach. The layers are illastrated in Figure 3.3 (Fensel, van Harmelen, Horrocks, McGuinness & Patel-Schneider, 2001).

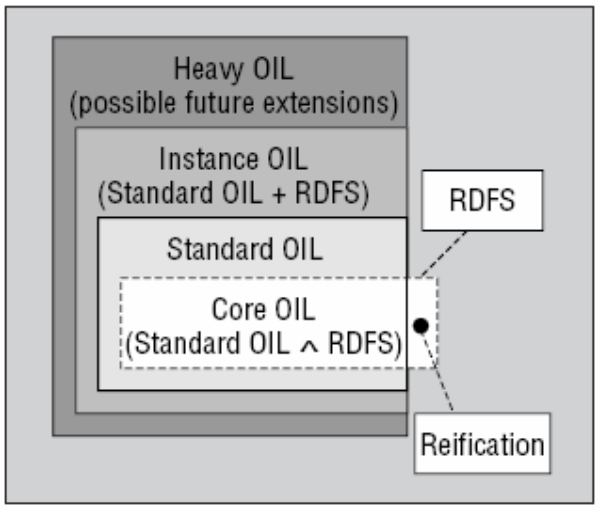

Figure 3.3 OIL's layered language model.

Each layer is built on top of the existing one by adding new features and complexity to the layer below.

The lowest layer, known as Core OIL, is compatible with RDF Schema, though RDF Schema's reification feature not supported. Ontologies defined by the Core OIL is interpretable by an RDF Schema aware application. The next layer, Standart OIL is designed so that it provides adequate expressive power as well as reasoning support. Instance OIL includes full integration of individuals into the language. On top, Heavy OIL includes more representational and reasoning capabilities.

3.2.6.2 DAML+OIL

DAML+OIL is an ontology language specifically designed for the use on the Web; it exploits existing Web standards (XML and RDF), adding the familiar ontological primitives of the object oriented and frame based systems, and the formal rigor of a very expressive description logic (Horrocks, 2002). DAML+OIL is the result of merging Ontology Inference Layer (OIL) and DARPA Agent Markup Language (DAML). DAML+OIL builds on RDF by supporting richer modelling primitives. Later DAML+OIL submitted to W3C as a protype of Web Ontology Language (OWL) which is now the W3C standart ontology language.

3.2.6.3 Web Ontology Language (OWL)

OWL is a revision of the DAML+OIL web ontology language incorporating lessons learned from the design and application of DAML+OIL (McGuinness & van Harmelen, 2004). OWL aims to provide sufficient expressive support to describe machine-processable content needed for Semantic Web applications. OWL is the proposed standard for building ontologies on the Web. OWL became W3C Recommendation in February 2004, taking its place alongside other Web standards such as URI, XML, etc. OWL provides three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

OWL Lite allows to express everything that RDF Schema allows and so much more besides. It extends RDF Schema and supports the users that are mainly interested in creating ontologies which involve only simple constraints and class taxonomies. An example of the simple constraints that OWL Lite provides is the cardinality feature: OWL Lite limits the cardinality values to only zero and one.

OWL DL provides increased expressiveness also ensuring computational properties (completeness, decidability) of reasoning systems. It is based on "Description Logics" and hence it is named with the suffex DL.

OWL Full provides more expressiveness and more syntactic flexibility to represent ontologies than OWL DL does. For example, in OWL Full a class can be defined as a collection of individuals and as an individual at the same time. A reasoning software could not totally support for every feature of OWL Full since OWL Full does not guarantee computational properties of reasoning systems.

3.2.7 Ontology Development Tools

Tools are important and needed for the purposes of ontology development and manipulation. There are numerous tools developed to support and ease the task of ontology development. Most of the tools include an ontology editor with a user-friendly graphical interface that allows users to create a new ontology from scratch, view and modify an existing ontology. Some tools have support on ontology documentation, ontology export and import to/from different formats and ontology languages. Also some tools allow adoption and merging of ontologies to make them fit for new tasks and domains. Gomez-Perez (2002) collects and introduces a wide range of ontology tools by categories as follow:

Ontology development tools: This group includes tools that can be used to build a new ontology from scratch. In addition to common edition and browsing functionality, these tools usually provide ontology documentation, ontology export and import to/from different formats and ontology languages, ontology graphical edition, ontology library management, etc.

Ontology merge and integration tools: These tools are used to solve the problem of merging and aligning different ontologies in the same domain.

Ontology evaluation tools: They are used to evaluate the content of ontologies and their related technologies. Ontology content evaluation tries to reduce problems that can be encountered on integrating and using ontologies and ontology-based technology in other information systems.

Ontology-based annotation tools: These tools allow users to insert instances of concepts and of relations in ontologies and maintain (semi)automatically ontology-based markups in Web pages.

Ontology storage and querying tools: These tools allow using and querying ontologies easily and performing inferences with them.

Ontology learning tools: They are used to (semi)automatically derive ontologies from natural language texts, and semi-structured sources and databases, by means of machine learning and natural language analysis techniques.

Among the most widely used ontology editors that support RDF, RDF Schema and/or OWL are Protégé, InferEd, and WebODE.

***Protégé*:** Protégé is a free, open source, graphical ontology editor that is built in Java. It is being developed by Stanford University and is supported by a strong community of developers. Protégé allows ontologies be exported into a variety of formats including RDF Schema, OWL, and XML Schema. The Protégé-OWL plug-in extends the Protégé platform into an ontology editor for the OWL.

***InferEd*:** InferEd supports the RDF Schema and OWL specifications. It provides integration with RDF Gateway database tables. It also provides simplified editing of RDF documents by allowing to view the content of an RDF document in any of the common formats including RDF/XML, N-Triples, N3/Turtle.

WebODE: WebODE has been developed by the Ontological Engineering Group (OEG) from the Artificial Intelligence Department of Technical University of Madrid (UPM). It supports most of the activities involved in the ontology development process. It has export and import services from and into XML, RDF Schema, OIL, DAML+OIL, and OWL.

Text-To-Onto: Text-to-Onto has been developed by the Knowledge Management Group of the Institute AIFB at the University of Karlsruhe. It supports developers in

the ontology construction process by applying text mining techniques. It provides an integrated environment for the task of learning ontologies from text.

Chimaera: Chimæra is a browser-based editing, merging, and diagnosis tool developed by the Stanford University Knowledge Systems Laboratory (KSL) (McGuinness, Fikes, Rice, & Wilder, 2000). It supports users in creating, merging and maintaining distributed ontologies on the web.

3.3 Description Logics

Description Logics (DL) form an important and powerful class of logic-based Knowledge Representation languages which represents the knowledge of an application domain in a structured and formally well-defined way.

3.3.1 Description Logics Overview

Knowledge Representation as a branch of Artificial Intelligence focuses on the design and implementation of languages and systems that represent knowledge. The main effort of research in Knowledge Representation is directed at providing theories and systems for expressing structured knowledge and for accessing and reasoning with it in a principled way (Fensel, 2000). Logics are formal languages for representing information such that conclusions can be drawn. There are varieties of logics/formalisms utilized in Knowledge Representation systems. The well-known ones are semantic networks, first-order logic, conceptual graphs, frames, and Description Logics.

In the non-logic-based Knowledge Representation systems, knowledge is represented by means of some ad hoc data structures, and reasoning is accomplished in an ad hoc manner by manipulating these structures. The well known examples of this approach are s*emantic networks* and *frames.*

The semantic networks was first introduced by Quillian (Quillian, 1968). Semantic networks represents knowledge constructs (concepts) and also their associations in a node-and-link network structure. Later the notion of frames was first introduced by Minsky (Minsky, 1975). A frame consists of slots. Attached to each slot can be descriptions or procedures. Collections of frames are organized and interconnected in frame systems.

One problem with semantic networks and frames is that, since they suffer from semantic formalism, the reasoning methods are strongly based on the particular implementation strategies, rather than on a formal language. For solving this problem, an important recognition is that, frames can be given a formal semantics based on first-order logic (Hayes, 1979), and furthermore, frames and semantic networks can be regarded as fragments of first-order logic (Brachman & Levesque, 1985).

Description Logics was designed as an extension to frames and semantic networks which were not equipped with a formal logic-based semantics. Previously Description Logics was called with various names such as terminological knowledge representation languages, concept languages, and term subsumption languages. Today Description Logics has become a cornerstone of the Semantic Web for its use in the design of ontologies. The W3C Recommendation language, Web Ontology Language (OWL), is based on Description Logics.

KL-ONE (Brachman & Schmolze, 1985) was recognised as the first DL-based system. In addition to supporting frame representation, KL-ONE included formalization with description-forming structures. It introduced most of the key notions: concepts and roles, number and value restrictions, subsumption and classification, the distinction between TBox and ABox.

A wide range of languages that based on Description Logics are developed with different level of expressivity. They are LOOM, BACK, KRIS, CLASSIC, FaCT, RACER, OIL, DAML+OIL and OWL.

Description Logics provides formalism for representing knowledge. Description Logics is a family of KR formalisms that represent the knowledge of an application domain by first defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects and individuals occurring in the domain (Baader, Calvanese, McGuinness, Nardi, Patel-Schneider, 2003).

Description Logics places an emphasis in concept definitions, roles and taxonomy building constructs. Concepts describe the common properties of a collection of individuals and roles are interpreted as binary relations. Description Logics allow specifying a terminological hierarchy in a subsumption framework. Subsumption is a special relation, such as the *subclassof* or *isa* relation that organizes the entire model into a hierarchy. Description Logics has formal semantics which can be used to automatically reason about Knowledge Base.

3.3.2 Description Logics Basic Formalism

A Knowledge Representation system based on Description Logics provides facilities to set up knowledge bases, to reason about their content, and to manipulate them. Figure 3.4 sketches the architecture of such a system (Baader, Calvanese, McGuinness, Nardi, Patel-Schneider, 2003).

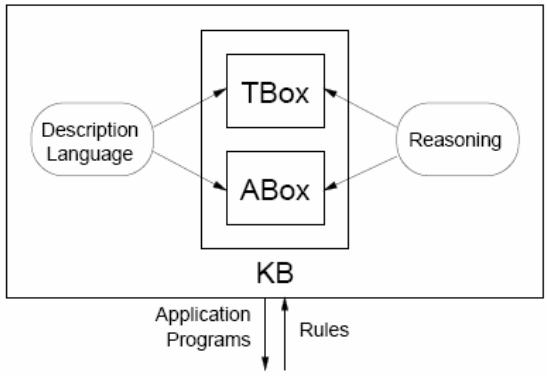

Figure 3.4 Architecture of a knowledge representation system based on Description Logics.

A knowledge base contains set of sentences in a formal language. A knowledge base based on Description Logics consists of two components, namely the TBox and the ABox.

- ***The TBox*** contains the *terminology* and is built through declarations that describe general properties of concepts. It is the intentional knowledge. The terminology is composed of concepts, which denote sets of individuals, and roles, which denote binary relationships between individuals. In addition, complex descriptions of concepts and roles can also be defined.

- ***The ABox*** contains the *assertions*, which is specific to individuals of the domain of discourse. It is the extensional knowledge.

In addition to storing terminologies and assertions, a Description Logics system also offers reasoning services about them. Considering both the structure of a Description Logics knowledge base and its associated reasoning services, Description Logics can be viewed as the most important set of formalism in the field of Knowledge Representation.

3.3.3 Description Logics and the Semantic Web

Ontologies play a crucial role in describing the meaning of resources in a machine-processable form. For this reason, the languages that are used to express ontologies have an extreme importance for the Semantic Web. Description Logics provides not only clear and well defined semantics but also provides an efficient reasoning support for ontology languages.

Ontologies that are expressed in logic-based Knowledge Representation languages such as OIL, DAML+OIL and OWL have Description Logics. This is the reason that ontologies modelled in those languages can be machine-processable. The machine knows exactly what the model means and how the model works logically, and can make inferences.

Description Logics is adopted as the formalism for building Semantic Web. It is used as the formal foundations on ontology languages, such as OIL, DAML+OIL, and OWL. Providing the formal basis for these important ontology languages, Description Logics are of great importance for the machine-processable of information. The well-defined semantics and efficient inference support of the Description Logics is essential to the success of the Semantic Web.

CHAPTER FOUR

A SEMANTIC WEB BASED SYSTEM

This chapter presents an example Semantic Web based system that developed to demonstrate a possible Semantic Web application. It introduces the approach for the construction of the system and describes the architecture, the ontology and the characteristics of the system.

4.1 Architecture of the System

An academic department of a university is selected as the domain since it contains a large number of entities and has quite many relations that connect each entity. Also the university domain is a popular domain for searching among students and academics that look for admission, publications, and so on.

By the system users can get substantial knowledge about staff, courses, students, research topics, and publications of the academic department. The ontology defines elements included and the activities occurred at the academic department. The system extracts information from ontology and makes it accessible to the end-user. Users can access the system through a conventional Web browser interface.

The overall architecture of the system consists of three main layers: User Interface Layer, Internal Layer, and Knowledge Layer. This architecture is illustrated in Figure 4.1.

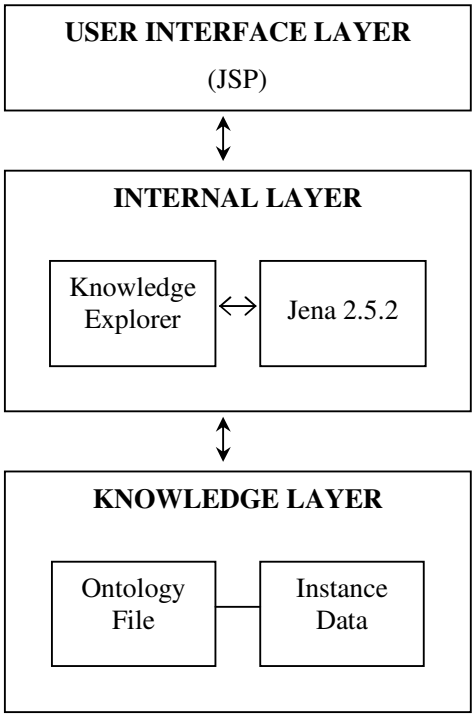

Figure 4.1 Overall architecture of the system.

- **User Interface Layer**

This layer interacts with the user and handles all incoming requests submitted by the users. Users can get the results and explore the ontology information using a conventional Web browser.

- **Internal Layer**

This layer handles the interaction with knowledge base and runs the business logic. Knowledge Explorer located at that layer gets the ontology and instance data from Knowledge Layer and transforms them to triples using Jena 2.5.2 API as the parser. Content is dynamically generated and presented to the user in HTML interface layout.

- **Knowledge Layer**

Knowledge Layer includes either a local or a remote file system that hosts the knowledge base file which contains the ontology and instances data in it.

4.2 Development of the Ontology

Ontologies are developed to build the fundamental knowledge structure of the application domain. They are used to describe a specification of basic categories by the class, property and relationship definitions, and also used to provide a structure to hold the instances data. With ontologies, the information is denoted in a semantically machine-processable format which enables machines to deduce additional information, a task difficult in a traditional database system.

Development of the ontology constitutes a core task of implementing a Semantic Web based system. The following subsections describe the ontology development procedure, characteristics of the ontology together with the ontology development language and tool used in the implementation for the example Semantic Web based system.

4.2.1 Ontology Development Procedure

Ontology is developed by considering and following the steps described in the section 3.2.5 of this thesis. An iterative process is used in the development of the ontology.

In the first step, the domain and the scope of the ontology is determined by posing basic questions like, what is the domain that the ontology will cover? For what the ontology will be used?

Next all the concepts which expected to appear in the ontology are figured out. A listing of key terms in the ontology, such as staff, professor, course, graduate student, under-graduate student, publication, etc. is created.

From the list of these key terms, classes are defined and organized in a class hierarchy. Classes describe concepts in the domain. Classes are organized into a superclass-subclass hierarchy, which is also known as taxonomy. Subclasses

represent the concepts that are more specific than the superclass. If every instance of class B is also an instance of class A, then Class A is defined as superclass of class B. For instance, the class *Staff* has two subclasses: *AcademicStaff* and *NonAcademicStaff*. The class *AcademicStaff* is sub-classed extensively to *Assistant*, *Instructor*, *and Professor*.

There are twelve classes defined in the ontology of the system. The hierarchy of classes are listed below, where a subclass is indented to show the subClassOf.

Owl:Thing
- Courses
- Publication
- ResearchTopics
- Staff
  - AcademicStaff
    - Asistant
    - Instructor
    - Professor
  - NonAcademicStaff
- Students
  - Graduate
  - UnderGraduate

The class *owl:Thing* is the class that represents the set of all classes.

After defining the classes that will exist in the ontology, in the next step, properties of the classes are defined. Properties are relationships between one or more arguments where each argument is either a class or a data type like integer, string, etc. There are two main types of properties: datatype properties and object properties.

Datatype properties are used for defining a value type for a property. Value type describes what types of values the property can be. For instance, *hasFirstName* is a datatype property attached to the class *Staff* with value type s*tring*. The domain of the property *hasFirstName* is the class *Staff* and the range of that property is the value type s*tring*.

Object properties, on the other hand, are used for defining a relationship between classes. The class or classes to which the property is attached constitute the domain of that property. The designated class or classes for that property is referred as the range of that property. For instance, *isTaughtBy* is an object property attached to the class *Courses* with value of the classes *Professor* and *Instructor.* The domain of the property *isTaughtBy* is the class *Courses* and the range of that property is the classes *Professor* and *Instructor.*

An object property may have a corresponding inverse property. If a property links the class A to the class B, then its inverse property links the class B to the class A. For instance, the inverse of the object property *isTaughtBy* is the object property *teaches.* The domain of the property *teaches* is the classes *Professor* and *Instructor* and the range of that property is the class *Courses.*

The ontology of the system includes thirty-two properties. Table 4.1 summarizes the properties defined in the ontology of the system.

Table 4.1 List of properties defined in the ontology.

RELATION	TYPE	DOMAIN	RANGE
assists	Object property	Assistant	Courses
hasAssistant	Object property	Courses	Assistant
hasClassNumber	Datatype property	UnderGraduate	String
hasCourseDescription	Datatype property	Courses	String
hasCourseHomePage	Datatype property	Courses	String
hasCourseID	Datatype property	Courses	String
hasCourseName	Datatype property	Courses	String
hasCredits	Datatype property	Courses	String
hasDegree	Datatype property	Graduate	String
hasEmail	Datatype property	Staff	String
hasFirstName	Datatype property	Staff	String
hasHomepage	Datatype property	AcademicStaff	String
hasInterest	Object property	Academic Staff	ResearchTopics
hasJobTitle	Datatype property	NonAcademicStaff	String
hasLastName	Datatype property	Staff	String
hasPhoneNumber	Datatype property	Staff	String
hasPublicationName	Datatype property	Publication	String
hasPublicationType	Datatype property	Publication	String
hasPublictionYear	Datatype property	Publication	String
hasStudentID	Datatype property	Students	String
hasStudentName	Datatype property	Students	String
hasStudentSurname	Datatype property	Students	String
hasTitle	Datatype property	Professor	String
hasTopic	Object property	Publication	ResearchTopics
hasTopicName	Datatype property	ResearchTopics	String
interestedBy	Object property	ResearchTopics	AcademicStaff
isPublishedBy	Object property	Publication	AcademicStaff
isTakenBy	Object property	Courses	Students
isTaughtBy	Object property	Courses	Professor, Instructor
publishes	Object property	AcademicStaff	Publication
takes	Object property	Students	Courses
teaches	Object property	Professor, Instructor	Courses

The last step in ontology development is creating instances. In this step, academic department of the university is fulfilled with usable data by creating instances. Instances are the individuals of the classes. An instance of a class is created by filling both datatype and object property values of the class.

For instance, *Prof_Dr_Barbara_Moon* is an instance of the class *Professor*. The instance has the following property values:

hasFirstName: Barbara
hasLastName: Moon
hasTitle: Full Professor
hasPhoneNumber: +90 232 9801777
hasEmail: barbara@myuniv.edu.tr
hasHomepage: http://www.myuniv.edu.tr/Barbara
publishes: Elements of Reusable Object Oriented Software
Programming from Specifications
hasInterest: Software Engineering
Information Systems
teaches: CSE304
CSE517

Figure 4.2 shows the screenshot of the definition of that instance in Protégé.

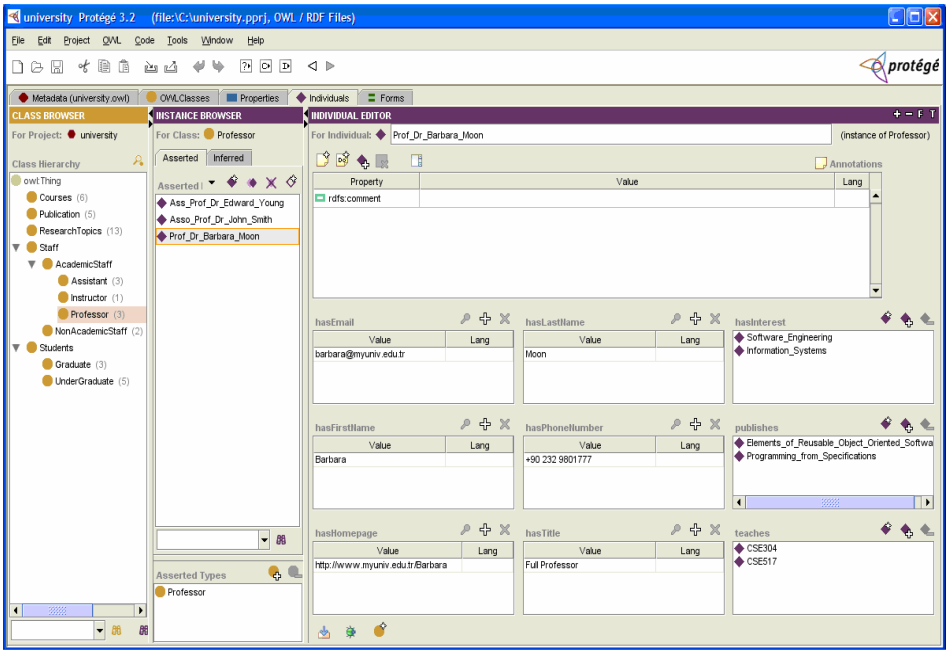

Figure 4.2 Screenshot of the definition of an instance in Protégé.

Ontologies that developed must be consistent and formally correct. Protégé has a feature to test the created ontology with “Run ontology test” option. The ontology developed has been tested and validated by the use of Protégé ontology test feature.

Figure 4.3 depicts the graphical representation of the ontology implemented. It displays all the classes, properties of the classes and the relationships between the classes. The OWL ontology file along with several instance data is attached to the Appendix A.

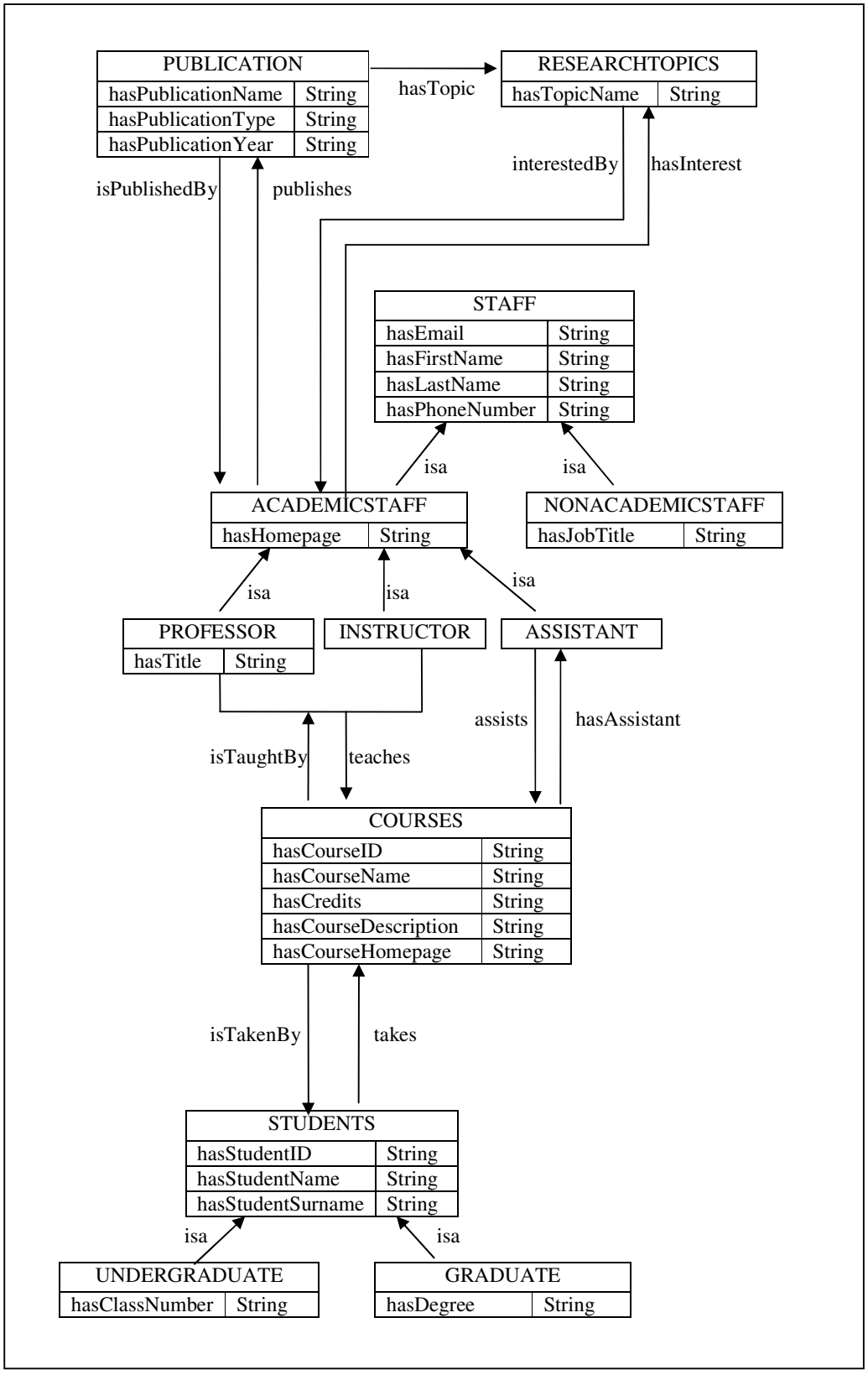

Figure 4.3 Graphical representation of the ontology.

4.2.2 Ontology Development Tool

In the development of the ontology, a popular ontology editor, Protégé 3.2 with OWL plug-in, is used. Protégé is a powerful tool for creating and editing ontologies and knowledge bases. Protégé has an easy-to-use graphical user interface and an extensible plug-in architecture.

In the implementation, Protégé is used basically to construct the ontology in OWL and also for the purpose of knowledge-acquisition, i.e. instantiating the classes defined in the ontology.

4.2.3 Ontology Development Language

The W3C Recommendation, OWL is selected as the ontology development language due to its expressiveness, standard, widely recognition, and tool support. OWL as an ontology language allows describing the semantics of knowledge in a machine-processable way.

The ontology of the system is represented in OWL format so that the knowledge encoded can be interpreted and processed by machines. Classes, properties and instances are formally represented with OWL language in the ontology file.

Figure 4.4 shows the base of an ontology in OWL. The first line in an OWL file indicates the version of XML. The beginning of OWL ontology is the *rdf:RDF* tag in which namespaces and the other constructs of the ontology are defined. Ontology header is defined in *owl:Ontology* tag. It includes housekeeping assertions like comment or explanation of the ontology, the version information and label of the ontology. After the end indicator of </owl:Ontology>, the main definitions of concepts, relations, restrictions, and instances are included.

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.myontology.com/university.owl#"
   xml:base="http://www.myontology.com/university.owl">
  <owl:Ontology rdf:about="">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >This is OWL file of my Semantic Web University</rdfs:comment>
   <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >My version 1.0.0</owl:versionInfo>
   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >University Ontology</rdfs:label>
  </owl:Ontology>
   <!--The rest of the ontology goes here >
   ……
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 3.2, Build 355)  http://protege.stanford.edu -->
```

Figure 4.4 The base part of the ontology.

Concepts in the ontology are expressed as classes. Figure 4.5 shows the definition of the class *Graduate* in OWL.

```
<owl:Class rdf:ID="Graduate">
   <rdfs:subClassOf rdf:resource="#Students"/>
   <owl:disjointWith>
     <owl:Class rdf:ID="UnderGraduate"/>
   </owl:disjointWith>
  </owl:Class>
```

Figure 4.5 A class definition in the ontology file.

The class *Graduate* is defined as a subclass of the more general class *Students*. It is also defined that the class *Graduate* is disjoint with the class *UnderGraduate.* Disjointness ensures that any instance which is defined as a member of one of these classes can not be a member of the other at the same time. Completeness of an ontology demands subclasses to be disjoint.

Datatype properties of a class are specified with *owl:DatatypeProperty* tag. Figure 4.6 shows the definition of the datatype property *hasCourseName* in OWL. It defines that the class *Courses* has *hasCourseName* as one of its properties.

```
<owl:DatatypeProperty rdf:ID="hasCourseName">
   <rdfs:domain rdf:resource="#Courses"/>
   <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
```

Figure 4.6 A datatype property definition in the ontology file.

Object properties specify relations between classes. Object properties of a class are specified with *owl:ObjectProperty* tag. Figure 4.17 shows the definition of the object property *takes* which links the class S*tudents* to the class C*ourses.* In this case, the domain of the object property *takes* is the class *Students* and the range is *Courses*.

```
<owl:ObjectProperty rdf:ID="takes">
   <rdfs:domain rdf:resource="#Students"/>
   <rdfs:range rdf:resource="#Courses"/>
   <owl:inverseOf>
     <owl:ObjectProperty rdf:ID="isTakenBy"/>
   </owl:inverseOf>
 </owl:ObjectProperty>
```

Figure 4.7 An object property definition in the ontology file.

The inverse of the object property *takes* is the object property *isTakenBy* which is defined in OWL as shown in the Figure 4.8.

```
<owl:ObjectProperty rdf:about="#isTakenBy">
   <rdfs:domain rdf:resource="#Courses"/>
   <owl:inverseOf rdf:resource="#takes"/>
   <rdfs:range rdf:resource="#Students"/>
 </owl:ObjectProperty>
```

Figure 4.8 An object property definition in the ontology file.

4.3 Programming Environment

4.3.1 Oracle JDeveloper

The system is implemented using Java programming language. Much of the application logic is implemented in a conventional object-oriented language Java. Java Server Pages (JSP) technology which enables to mix regular, static HTML with dynamically generated content is used to handle the presentation of the user interface.

Oracle JDeveloper 10.1.3 is selected as the development environment for the system. Oracle JDeveloper provides a single, visual, developer-friendly Integrated Development Environment (IDE) for Java programming. It is Java-based and runs on multiple platforms including Windows, Linux, Mac, and various Unix-based systems, enabling the developers to choose their preferred development platform. It offers a rich set of features for designing, modelling, developing, testing, debugging, and deploying Java 2 Platform, Enterprise Edition (J2EE) applications.

4.3.2 Jena 2.5.2 API

Jena is a Java-based open source API that is widely used in the development of Semantic Web applications. Since Jena is based on Java, it is easy to integrate it with any Java application. Jena has being developed by the Hewlett-Packard (HP) Company. It can be downloaded from the SourgeForge Web site (http://jena.sourceforge.net/).

Jena provides a Java library for ontology languages and reasoning over data described in them. The ontology and instance data stored in files can be processed, manipulated and queried by Jena API. Jena supports RDF, RDF Schema, DAML+OIL, and OWL, including a rule based engine and support for querying ontologies. The advantage of Jena Ontology API is that it aims to provide a consistent programming interface to the Semantic Web application developer, independent of which ontology language is being processed (Anonymous, n.d.).

Jena has classes to represent graphs, resources, properties, and instances that exist in the ontology file. Graph which holds the RDF statements is called a model and is represented by the *Model* interface. So the *Model* allows access to RDF statements held in base graph.

Figure 4.9 shows the Jena's ontology architecture base (Anonymous, n.d.). Above the simple base graph, Jena's architecture provides a mechanism for reasoners, if any is being used. Jena is quite flexible in that it is designed to allow inference engines or reasoners to be plugged into Jena. At the top ontology model which is represented by *OntModel* adds support for the kinds of objects that exists in an ontology like classes, properties and instances. In other words, in Jena an ontology is represented as a model which contains resources like classes, properties and instances.

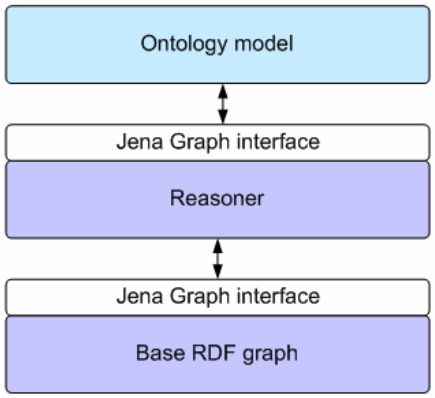

Figure 4.9 Jena's ontology architecture base.

Ontology model which can be considered as an extension of the Jena RDF base model provides extra capabilities for handling ontology files. Ontology models are created using the Jena's *ModelFactory*. Jena's *ModelFactory* class is the preferred means of creating different types of models.

Figure 4.10 shows Jena code that creates an ontology model and then loads an ontology file to the model created. The ontology file is accessed and opened using the read method. This method takes the URL of the file as its parameter.

```
import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntProperty;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.RDFNode;
import java.util.Iterator;
public static OntModel loadOntology(BaseURI) {
        OntModel model = ModelFactory.createOntologyModel();
        model.read(BaseURI);
        return model;
    }
```

Figure 4.10 Creating an ontology model and loading an ontology file to the model in Jena.

Classes which are the basic building blocks of an ontology is represented in Jena by *OntClass*. Figure 4.11 shows Jena code which lists the classes in the ontology file.

```
public static void listClasses(OntModel model) {
   for (Iterator i = model.listClasses(); i.hasNext(); )
  {
      OntClass Class = (OntClass)i.next();
      System.out.println(Class.getURI());
   }
}
```

Figure 4.11 Listing classes in Jena.

Figure 4.12 shows Jena code which lists the instances of a class.

```
public static void listInstances(OntModel model, String className) {
   OntClass cls;
   cls=model.getOntClass(BaseURI+className);
   for(Iterator i= cls.listInstances();i.hasNext();)
   {
       System.out.println(((Individual)i.next()).toString());
   }
}
```

Figure 4.12 Listing instances of a class in Jena.

Figure 4.13 shows Jena code which lists the instances of all classes in the ontology file.

```
public static void listAllInstances(OntModel model) {
    for (Iterator i = model.listClasses(); i.hasNext(); ) {
        OntClass cls = (OntClass)i.next();
        System.out.println("Instances for class:" + cls.getURI());
        for (Iterator j = cls.listInstances(); j.hasNext(); ) {
            Individual ind = (Individual)j.next();
            System.out.println("\t" + ind.getURI());
        }
    }
}
```

Figure 4.13 Listing the instances of all classes in Jena.

Figure 4.14 shows Jena code which lists the properties of all the classes in the ontology file.

```
public static void listPropertiesofClasses(OntModel model) {
    for (Iterator i = model.listClasses(); i.hasNext(); ) {
        OntClass cls = (OntClass)i.next();
        System.out.println("Properties for class:" + cls.getURI());
        for (Iterator j = cls.listDeclaredProperties(); j.hasNext(); ) {
            OntProperty prop = (OntProperty)j.next();
            System.out.println("\t" + prop);
        }
    }
}
```

Figure 4.14 Listing the properties of all classes in Jena.

Figure 4.15 shows Jena code which lists the statements as subject, property, and object form in the ontology file.

```
    public static void listStatements(OntModel model, String className) {
        String subject, pred, object;
        subject="";
        pred="";
        object="";
        OntClass oc = model.getOntClass(BaseURI + className);
        for (Iterator j = oc.listInstances(); j.hasNext(); ) {
            Individual ind = (Individual)j.next();
            subject = ind.toString();
            for (Iterator t = oc.listDeclaredProperties(); t.hasNext(); ) {
                OntProperty prop = (OntProperty)t.next();
                pred = prop.getURI() + ": ";
                for (Iterator ss = ind.listPropertyValues(prop); ss.hasNext();
                ) {
                    RDFNode value = (RDFNode)ss.next();
                    object = value.toString();
                    System.out.println(subject + pred + object);
                }
            }
        }
}
```

Figure 4.15 Listing the all statements as subject, property, and object form in Jena.

4.3.3 SPARQL

Query languages are used to get information from a knowledge base or from a conventional data store. In the case of the Semantic Web, the most promising RDF query language is probably SPARQL. It is undergoing standardization by the RDF Data Access Working Group (DAWG) of the W3C and it is currently a W3C Last Call Working Draft (Prud'hommeaux & Seaborne, 2007).

SPARQL stands for **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage. As it is implied by its name, SPARQL is a general term for both a protocol and a query language. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions and also supports extensible value testing and constraining queries by source RDF graph (Prud'hommeaux & Seaborne, 2007).

There are several tools and APIs that already provide SPARQL functionality, such as ARQ (a SPARQL processor for Jena), AllegroGraph RDFStore, Protégé, RDF API for PHP, Top Braid Composer.

SPARQL is a syntactically-SQL-like language for querying RDF graphs via pattern matching. Following is the SPARQL query that models the competency question “Which students take the course “CSE100?”.

```
PREFIX table:<http://www.myontology.com/university.owl#>
SELECT  ?y
FROM <http://www.myontology.com/university.owl#>
WHERE {
?x table:hasCourseID 'CSE100'.
?x table:isTakenBy ?y.
}
```

Starting from the top, we encounter the PREFIX keyword. PREFIX is essentially the SPARQL equivalent of declaring an XML namespace. It associates a label with a specific URI. The start of the query proper is the SELECT keyword. Like in a SQL query, the SELECT clause is used to define the data items that will be returned by a query. The FROM keyword identifies the data against which the query will be run. In this instance, the query references the URI of the ontology. We have the WHERE clause which is a query pattern as a list of triple patterns. A graph pattern is a collection of triple patterns that identify the shape of the graph that we want to match against.

SPARQL support in Jena is currently available via a module called ARQ. It provides creating and executing SPARQL queries from Java.

Figure 4.16 shows Jena code which implements the above example SPARQL query. SPARQL queries are created and executed with Jena using classes in the *com.hp.hpl.jena.query* package.

```
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.query.*;

public static void deneme() {
  String cID="CSE517";
  String qStr =  "PREFIX table:<http://www.myontology.com/university.owl#> " +
  "SELECT ?y " +
  "WHERE { " +
  "?x table:hasCourseID '" + cID + "'. " +
  "?x table:isTakenBy ?y. " +
  "}";
  Query query = QueryFactory.create(qStr);
  QueryExecution qe = QueryExecutionFactory.create(query, model);
  ResultSet results = qe.execSelect();
  ResultSetFormatter.out(System.out, results, query);
  qe.close();
}
```

Figure 4.16 Executing a SPARQL query in Jena.

4.4 Characteristics of the System

The system provides a web browser interface to access and view the ontology information. At the top part and the left hand side of the interface, there are links provided to access the semantic information of an academic department in the

university community. The user can access, browse, and view ontology information by clicking these links provided. The links include: Staff, Students, Courses, Research Topics, and Publications.

Figure 4.17 shows the homepage of the interface.

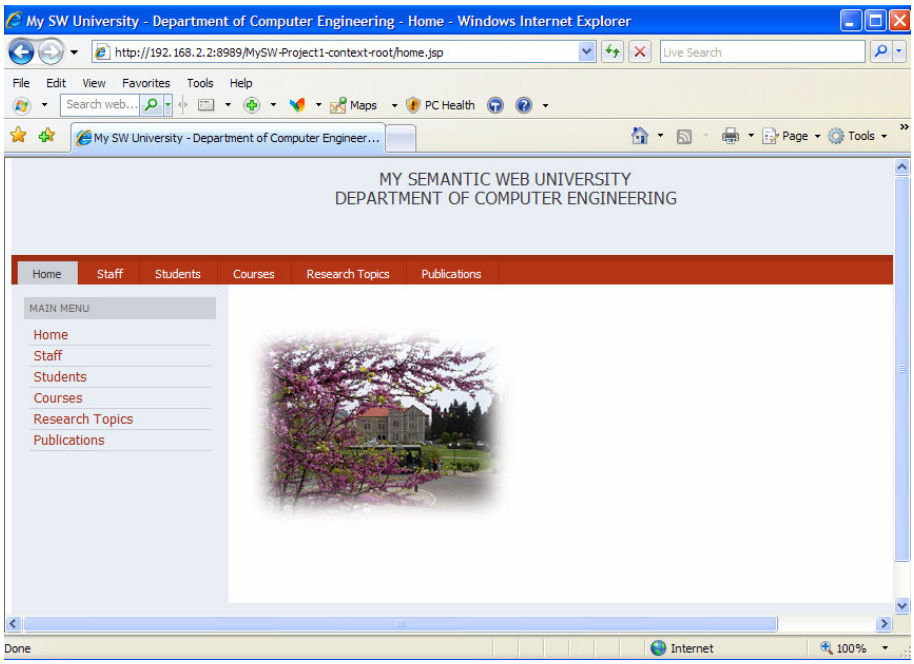

Figure 4.17 User homepage interface.

Figure 4.18 shows the snapshot of the interface when the user presses “Staff” link. The user can then make a selection by clicking an associate link to access the Professors, Instructors, Assistants, and Non-academic Staff.

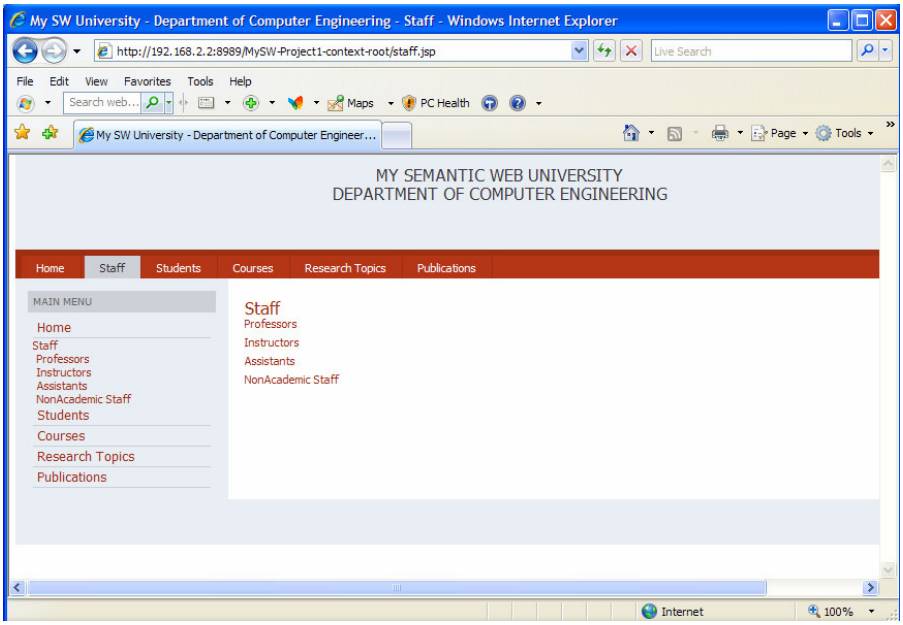

Figure 4.18 User interface of staff.

The Professors interface is shown in Figure 4.19. It displays detailed information about each professor, including the title, first name, last name, e-mail, homepage, phone number, publications and the research topics interested by the professor.

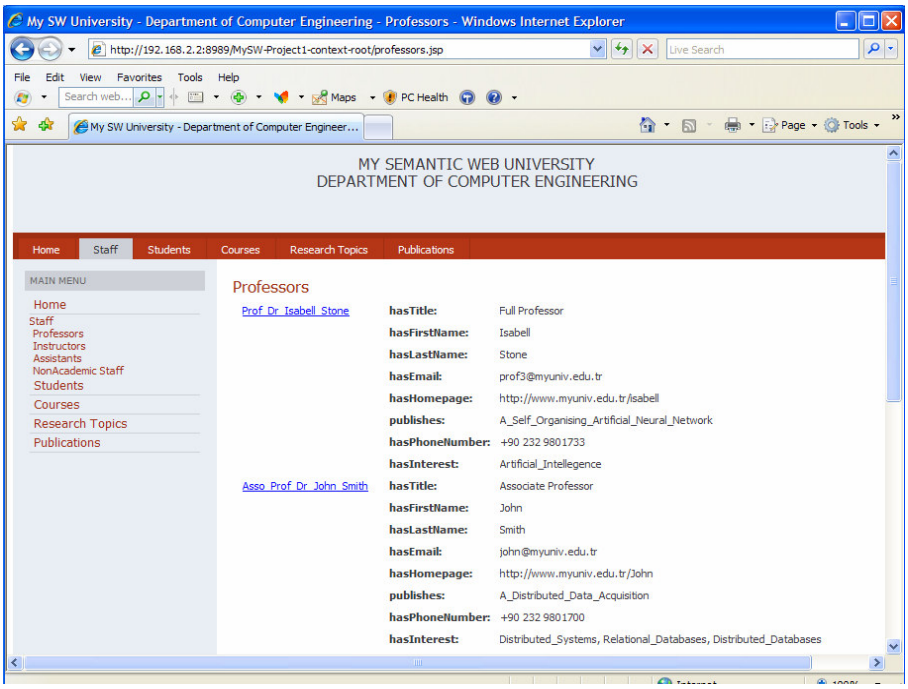

Figure 4.19 User interface of professors.

The Courses interface is similar to the Professors interface. The Courses interface is shown in Figure 4.20.

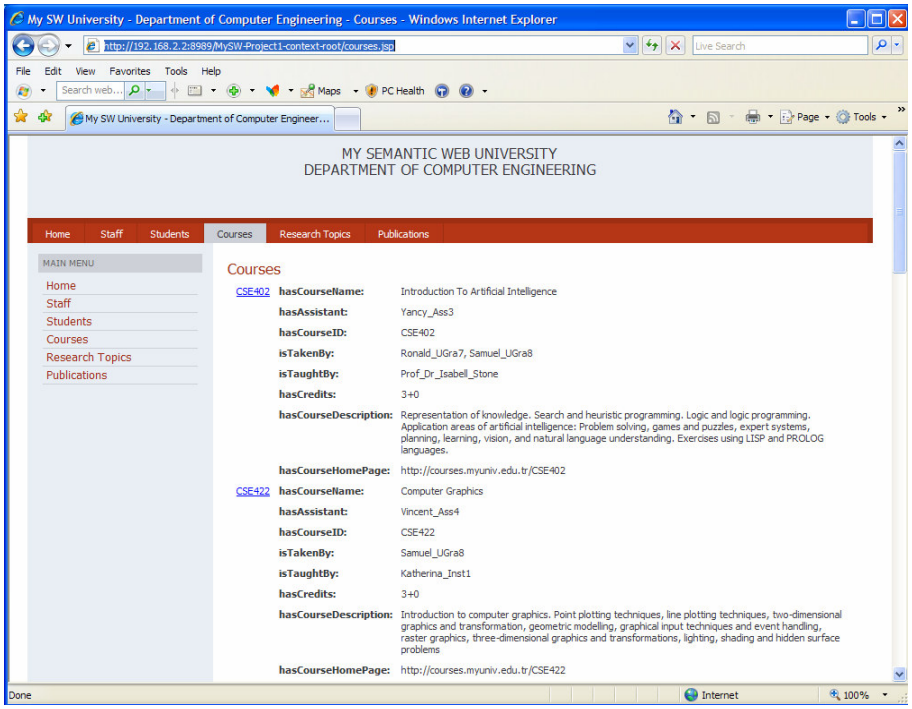

Figure 4.20 User interface of courses.

It displays detailed information about each course, including the course ID, course name, course credits, course description, course homepage, the academic staff who teaches the course, the assistants who assist the course and the students who take the course.

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

5.1 Summary

The Semantic Web has been promoted as the future of the current Web. In this thesis, key theories and technologies surrounding the Semantic Web have been introduced and explained. A comprehensive overview of the Semantic Web technology covering its vision, its architecture, its theory, its tools and its languages has been presented.

The Semantic Web proposes content on the Web be organized with semantics for machines consumption. Ontologies play a key role in the Semantic Web applications since they represent significant concepts and relationships for the application domain in a machine-processable format. Ontologies abstract the knowledge of application domain and enable processing and sharing of this knowledge.

In this thesis, an example prototype Semantic Web based system for an academic department of a university has been designed and implemented. The system implemented in this thesis uses state of the art principles of the Semantic Web. An ontology has been developed to model the knowledge structure of the application domain in a machine-processable format. Since the information is denoted semantically in a formal way with an ontology, the system provides enhanced capabilities compared to the classical web applications: First, it enables to separate the content from presentation. Second, semantically marked-up information in a machine-processable form makes it possible for end-user applications to make intelligent decisions and also to deduce additional information which is a task difficult in a traditional system. The system can be easily put into computation and inference by machines.

The ontology of the system has been specified in a machine-processable format using the W3C Recommendation language, OWL. OWL allows the construction of sophisticated representations of knowledge providing a rich set of primitives and elements for describing the application domain. Protégé has been used as the ontology editor. The system has been implemented using Java programming language. The architecture of the system consists of three major layers, namely user interface layer, internal layer, and knowledge layer. The knowledge base of the system which consists of the ontology and the instance data has been processed, manipulated and queried by Jena API and presented to the user in HTML interface layout through a conventional Web browser. The system allows the user to access, browse, and query the information contained in the knowledge base.

In conclusion, this thesis explores the Semantic Web from different points of view, including its theoretical root, its technology, and its implementation. It explains the vision of the Semantic Web, presents the language, tools and general framework for achieving the Semantic Web, demonstrates the steps and techniques for building a Semantic Web application, and introduces an example Semantic Web based prototype system for projecting to a future vision.

5.2 Future Works

The Semantic Web is still under its development. The W3C is continuing its work on developing specifications and standards. At this stage it is hard to judge to what degree this vision will eventually be implemented or when its use and deployment will become widespread. A huge volume of research activity is going on in this area. Some applications based on Semantic Web technology are already beginning to appear.

It is convinced that in the future the technologies being developed will make more efficient use of the Web and will offer more active, automated and intelligent services in all kinds of application areas, such as e-commerce, web services, and e-learning, etc. Based on more machine-processable content, agents can browse,

search, match the information on the Web and can act on behalf of human. Furthermore, these agents can also be implemented on other types of computing devices, such as mobile phones, PDAs, or household appliances, etc. There remains significant opportunity in the area of the Semantic Web. This thesis is groundwork for further research and development in this area.

The example system implemented in this thesis utilizes Semantic Web technology as its principle architecture. The system and/or the ontology developed in this thesis can be taken as a model and can be easily adapted to other applications of the Semantic Web. The enhancement of the system includes further implementation. The system is an example research prototype. The scale of the system and amount of data being processed is small. Further optimization might be needed to enable the system to handle large number of data and concurrent users. In addition, some applications, such as agents, web services, expert systems, decision support systems can be implemented to benefit from the system developed.

REFERENCES

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2003). *The description logic handbook: Theory, implementation, and applications.* Cambridge University Press

Beckett, D. (2004). *RDF/XML syntax specification (revised).* Retrieved April 20, 2007, from http://www.w3.org/TR/rdf-syntax-grammar/.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American,* May 2001.

Brachman, R. J., & Levesque, H. J. (Eds.). (1985). *Readings in knowledge representation*, Morgan Kaufman Pub.

Brachman, R. J., & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science, 9* (2), 171–216.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2006). *Extensible markup language (XML) 1.0 (fourth edition)*. Retrieved April 20, 2007, from http://www.w3.org/TR/xml/.

Brickley, D., & Guha, R.V. (2004). *RDF vocabulary description language 1.0: RDF Schema.* Retrieved April 20, 2007, from http://www.w3.org/TR/rdf-schema/.

Broekstra, J., Klein, M., Decker, S., Fensel, D., & Horrocks, I. (2000). Adding formal semantics to the Web: Building on top of RDF Schema. *Proceedings of the ECDL 2000 Workshop on the Semantic Web*, Lisbon, Portugal.

Fensel D. (2000). Relating ontology languages and web standards. In J. Ebert et al. (Eds.). *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik*, Modellierung 2000, St. Goar, April 5-7, 2000, Foelbach-Verlag, Koblenz.

Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M., & Klein, M. (2000). OIL in a nutshell. *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW 2000)*, Lecture Notes In Computer Science, Vol. 1937, Springer-Verlag, 1-16.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D., & Patel-Schneider, P.F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems, 16* (2), 38-45.

Gomez-Perez A. (2002). *A survey on ontology tools*, OntoWeb deliverable D1.3. Retrieved April 20, 2007, from http://citeseer.ist.psu.edu/525623.html.

Gomez-Perez, A., & Corcho O. (2002). Ontology languages for the semantic web. *IEEE Intellegent Systems, 17* (1), 54-60.

Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering: With examples from the areas of knowledge management, e-commerce and the semantic web*, Springer.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5* (2), 199-220.

Guarino, N. (1998). Formal ontology and information systems. In N. Guarino (Ed.). *Formal ontology in information systems*. Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98, Trento, Italy, (3-15). Amsterdam: IOS Press.

Guarino, N., & Giaretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars, (Ed.). *Towards very large knowledge bases: Knowledge building and knowledge sharing* (25-32). Amsterdam: IOS Press.

Hayes, P. (1979). The logic of frames. In D. Metzing (Ed.). *Frame conceptions and text understanding* (46-61). Berlin: Walter de Gruyter and Co.

Horrocks, I. (2002). DAML+OIL: A description logic for the semantic web. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 25* (1), 4-9.

Jena 2 Ontology API. (n.d.). Retrieved April 20, 2007, from http://jena.sourceforge.net/ontology/index.html.

Klyne, G., & Carroll, J.J. (2004). *Resource description framework (RDF): Concepts and abstract syntax.* Retrieved April 20, 2007, from http://www.w3.org/TR/rdf-concepts/.

Manola, F., & Miller, E. (2004). *RDF primer*. Retrieved April 20, 2007, from http://www.w3.org/TR/rdf-primer/.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.). *The Psychology of Computer Vision* (211-277). New York: McGraw-Hill.

McGuinness, D. L., Fikes, R., Rice, J., & Wilder, S. (2000). An environment for merging and testing large ontologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000).* Breckenridge, Colorado, USA.

McGuinness, D. L., & van Harmelen, F. (2004). *OWL web ontology language overview.* Retrieved April 20, 2007, from http://www.w3.org/TR/owl-features/.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. (1991). Enabling technology for knowledge sharing, *AI Magazine*, *12* (3), 36-56.

Noy N. F., & McGuinness, D. L. (2001). *Ontology development 101: A guide to creating your first ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

Prud'hommeaux, E. & Seaborne, A. (2007). *SPARQL Query Language for RDF.* Retrieved April 20, 2007, from http://www.w3.org/TR/rdf-sparql-query/.

Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.). *Semantic information processing* (216-270). Cambridge: MIT Press.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *IEEE Transactions on Data and Knowledge Engineering, 25* (1-2), 161-197.

Swartout, B., Patil, R., Knight, K., & Russ, T. (1996). Toward distributed use of large-scale ontologies. *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

The World Wide Web Consortium (W3C) (2003). *Semantic web status and direction ISWC2003 keynote.* Retrieved April 20, 2007, from http://www.w3.org/2003/Talks/1023-iswc-tbl/slide26-0.html.

APPENDIX A

ONTOLOGY FILE: UNIVERSITY.OWL

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.myontology.com/university.owl#"
  xml:base="http://www.myontology.com/university.owl">
  <owl:Ontology rdf:about="">
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >This is OWL file of my Semantic Web University</rdfs:comment>
   <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >My version 1.0.0</owl:versionInfo>
   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >University Ontology</rdfs:label>
  </owl:Ontology>
  <owl:Class rdf:ID="Publication"/>
  <owl:Class rdf:ID="NonAcademicStaff">
   <owl:disjointWith>
    <owl:Class rdf:ID="AcademicStaff"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Staff"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Students">
   <owl:disjointWith>
    <owl:Class rdf:about="#Staff"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="Courses"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="ResearchTopics"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Graduate">
   <rdfs:subClassOf rdf:resource="#Students"/>
   <owl:disjointWith>
    <owl:Class rdf:ID="UnderGraduate"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Courses">
   <owl:disjointWith rdf:resource="#Students"/>
   <owl:disjointWith>
    <owl:Class rdf:about="#Staff"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ResearchTopics"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Staff">
```

```
  <owl:disjointWith rdf:resource="#Students"/>
  <owl:disjointWith rdf:resource="#Courses"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ResearchTopics"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#ResearchTopics">
  <owl:disjointWith rdf:resource="#Courses"/>
  <owl:disjointWith rdf:resource="#Staff"/>
  <owl:disjointWith rdf:resource="#Students"/>
</owl:Class>
<owl:Class rdf:about="#UnderGraduate">
  <owl:disjointWith rdf:resource="#Graduate"/>
  <rdfs:subClassOf rdf:resource="#Students"/>
</owl:Class>
<owl:Class rdf:about="#AcademicStaff">
  <owl:disjointWith rdf:resource="#NonAcademicStaff"/>
  <rdfs:subClassOf rdf:resource="#Staff"/>
</owl:Class>
<owl:Class rdf:ID="Instructor">
  <owl:disjointWith>
    <owl:Class rdf:ID="Assistant"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Professor"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#AcademicStaff"/>
</owl:Class>
<owl:Class rdf:about="#Assistant">
  <rdfs:subClassOf rdf:resource="#AcademicStaff"/>
  <owl:disjointWith rdf:resource="#Instructor"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Professor"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Professor">
  <rdfs:subClassOf rdf:resource="#AcademicStaff"/>
  <owl:disjointWith rdf:resource="#Assistant"/>
  <owl:disjointWith rdf:resource="#Instructor"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="publishes">
  <rdfs:domain rdf:resource="#AcademicStaff"/>
  <rdfs:range rdf:resource="#Publication"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isPublishedBy"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="assists">
  <rdfs:domain rdf:resource="#Assistant"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasAssistant"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Courses"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isPublishedBy">
  <rdfs:domain rdf:resource="#Publication"/>
  <rdfs:range rdf:resource="#AcademicStaff"/>
  <owl:inverseOf rdf:resource="#publishes"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTopic">
  <rdfs:domain rdf:resource="#Publication"/>
  <rdfs:range rdf:resource="#ResearchTopics"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="interestedBy">
  <rdfs:domain rdf:resource="#ResearchTopics"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasInterest"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#AcademicStaff"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isTaughtBy">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="teaches"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Courses"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Professor"/>
        <owl:Class rdf:about="#Instructor"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="takes">
  <rdfs:domain rdf:resource="#Students"/>
  <rdfs:range rdf:resource="#Courses"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isTakenBy"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isTakenBy">
  <rdfs:domain rdf:resource="#Courses"/>
  <owl:inverseOf rdf:resource="#takes"/>
  <rdfs:range rdf:resource="#Students"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasAssistant">
  <rdfs:domain rdf:resource="#Courses"/>
  <owl:inverseOf rdf:resource="#assists"/>
  <rdfs:range rdf:resource="#Assistant"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#teaches">
  <owl:inverseOf rdf:resource="#isTaughtBy"/>
  <rdfs:range rdf:resource="#Courses"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Professor"/>
        <owl:Class rdf:about="#Instructor"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasInterest">
  <rdfs:range rdf:resource="#ResearchTopics"/>
  <owl:inverseOf rdf:resource="#interestedBy"/>
```

```
  <rdfs:domain rdf:resource="#AcademicStaff"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasClassNumber">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#UnderGraduate"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPublicationName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Publication"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasTitle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Professor"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCourseID">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Courses"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasLastName">
  <rdfs:domain rdf:resource="#Staff"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasEmail">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasJobTitle">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#NonAcademicStaff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasStudentID">
  <rdfs:domain rdf:resource="#Students"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCourseName">
  <rdfs:domain rdf:resource="#Courses"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCredits">
  <rdfs:domain rdf:resource="#Courses"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasHomepage">
  <rdfs:domain rdf:resource="#AcademicStaff"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPublicationYear">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Publication"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCourseDescription">
  <rdfs:domain rdf:resource="#Courses"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasTopicName">
  <rdfs:domain rdf:resource="#ResearchTopics"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="hasDegree">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Graduate"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPublicationType">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Publication"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasPhoneNumber">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasFirstName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCourseHomePage">
  <rdfs:domain rdf:resource="#Courses"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasStudentName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Students"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasStudentSurname">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Students"/>
</owl:DatatypeProperty>
<UnderGraduate rdf:ID="Jason_UGra4">
  <hasStudentName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Jason</hasStudentName>
  <hasStudentSurname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >UGra4</hasStudentSurname>
  <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ug4@stu.myuniv.edu.tr</hasEmail>
  <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >UGLastname4</hasLastName>
  <takes>
    <Courses rdf:ID="CSE100">
      <hasCredits rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >2+2</hasCredits>
      <hasAssistant>
        <Assistant rdf:ID="Benjamin_Ass2">
          <assists rdf:resource="#CSE100"/>
          <hasHomepage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.myuniv.edu.tr/assistant2</hasHomepage>
          <assists>
            <Courses rdf:ID="CSE304">
              <isTaughtBy>
                <Professor rdf:ID="Prof_Dr_Barbara_Moon">
                  <hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >Barbara</hasFirstName>
                  <publishes>
                    <Publication rdf:ID="Elements_of_Reusable_Object_Oriented_Software">
                      <isPublishedBy>
                        <Assistant rdf:ID="Dominic_Ass1">
                          <assists>
                            <Courses rdf:ID="CSE517">
```

```
                        <hasAssistant rdf:resource="#Dominic_Ass1"/>
                        <hasCourseDescription rdf:datatype=
                        "http://www.w3.org/2001/XMLSchema#string"
                        >advanced course in information Systems.</hasCourseDescription>
                        <hasCourseName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                        >Advanced Information Systems</hasCourseName>
                        <isTaughtBy rdf:resource="#Prof_Dr_Barbara_Moon"/>
                        <isTakenBy>
                          <Graduate rdf:ID="Simon_Gra3">
                            <hasStudentSurname rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >Gra3</hasStudentSurname>
                            <hasDegree rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                            >Phd</hasDegree>
                            <hasLastName rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >GLastname2</hasLastName>
                            <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                            >g2@stu.myuniv.edu.tr</hasEmail>
                            <hasStudentID rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >666</hasStudentID>
                            <hasStudentName rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >Simon</hasStudentName>
                            <takes rdf:resource="#CSE517"/>
                          </Graduate>
                        </isTakenBy>
                        <isTakenBy>
                          <Graduate rdf:ID="Eric_Gra2">
                            <hasDegree rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                            >Master</hasDegree>
                            <hasLastName rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >GLastname3</hasLastName>
                            <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                            >g3@stu.myuniv.edu.tr</hasEmail>
                            <hasStudentID rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >502</hasStudentID>
                            <hasStudentName rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >Eric</hasStudentName>
                            <hasStudentSurname rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >Gra2</hasStudentSurname>
                            <takes rdf:resource="#CSE517"/>
                            <takes>
                              <Courses rdf:ID="CSE450">
                                <hasCourseDescription
                                 rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                                >Centralization versus decentralization in
databases.</hasCourseDescription>
                                <isTaughtBy>
                                  <Professor rdf:ID="Asso_Prof_Dr_John_Smith">
                                    <publishes>
                                      <Publication rdf:ID="A_Distributed_Data_Acquisition">
                                        <isPublishedBy rdf:resource="#Asso_Prof_Dr_John_Smith"/>
```

```
    <hasTopic>
      <ResearchTopics rdf:ID="Distributed_Systems">
        <interestedBy rdf:resource="#Asso_Prof_Dr_John_Smith"/>
        <hasTopicName
         rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#string"
        >Distributed Systems</hasTopicName>
      </ResearchTopics>
    </hasTopic>
    <hasTopic>
      <ResearchTopics rdf:ID="Distributed_Databases">
        <interestedBy rdf:resource="#Asso_Prof_Dr_John_Smith"/>
        <hasTopicName
         rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#string"
        >Distributed Databases</hasTopicName>
      </ResearchTopics>
    </hasTopic>
    <hasPublicationYear
     rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >April 2006</hasPublicationYear>
    <hasPublicationName
     rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >A Distributed Data Acquisition</hasPublicationName>
    <hasPublicationType
     rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >Article</hasPublicationType>
   </Publication>
  </publishes>
  <hasInterest rdf:resource="#Distributed_Databases"/>
  <hasEmail rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string"
  >john@myuniv.edu.tr</hasEmail>
  <hasFirstName rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string"
  >John</hasFirstName>
  <teaches rdf:resource="#CSE450"/>
  <hasTitle rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string"
  >Associate Professor</hasTitle>
  <hasInterest>
   <ResearchTopics rdf:ID="Relational_Databases">
    <interestedBy>
     <Assistant rdf:ID="Yancy_Ass3">
      <assists rdf:resource="#CSE100"/>
      <hasInterest>
       <ResearchTopics rdf:ID="Software_Engineering">
        <hasTopicName
         rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#string"
        >Software Engineering</hasTopicName>
        <interestedBy rdf:resource="#Yancy_Ass3"/>
        <interestedBy rdf:resource="#Prof_Dr_Barbara_Moon"/>
        <interestedBy rdf:resource="#Dominic_Ass1"/>
       </ResearchTopics>
```

```
                                    </hasInterest>
                                    <hasHomepage
                                     rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#string"
                                    >http://www.myuniv.edu.tr/assistant3</hasHomepage>
                                    <hasLastName
                                     rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#string"
                                    >Ass3</hasLastName>
                                    <hasFirstName
                                     rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#string"
                                    >Yancy</hasFirstName>
                                    <hasPhoneNumber
                                     rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#string"
                                    >+90 232 9801818</hasPhoneNumber>
                                    <publishes>
                                      <Publication
rdf:ID="A_Self_Organising_Artificial_Neural_Network">
                                        <hasPublicationName
                                         rdf:datatype=
                                        "http://www.w3.org/2001/XMLSchema#string"
                                        >A Self Organising Artificial Neural
Network</hasPublicationName>
                                        <isPublishedBy rdf:resource="#Yancy_Ass3"/>
                                        <hasPublicationType
                                         rdf:datatype=
                                        "http://www.w3.org/2001/XMLSchema#string"
                                        >Article</hasPublicationType>
                                        <hasPublicationYear
                                         rdf:datatype=
                                        "http://www.w3.org/2001/XMLSchema#string"
                                        >Mach 2000</hasPublicationYear>
                                      </Publication>
                                    </publishes>
                                    <hasEmail
                                     rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#string"
                                    >yancy@myuniv.edu.tr</hasEmail>
                                    <hasInterest rdf:resource="#Relational_Databases"/>
                                  </Assistant>
                                </interestedBy>
                                <interestedBy rdf:resource="#Benjamin_Ass2"/>
                                <interestedBy rdf:resource="#Asso_Prof_Dr_John_Smith"/>
                                <hasTopicName
                                 rdf:datatype=
                                "http://www.w3.org/2001/XMLSchema#string"
                                >Relational Databases</hasTopicName>
                              </ResearchTopics>
                            </hasInterest>
                            <hasPhoneNumber rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >+90 232 9801700</hasPhoneNumber>
                            <hasInterest rdf:resource="#Distributed_Systems"/>
                            <hasLastName rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#string"
                            >Smith</hasLastName>
```

```
            <hasHomepage rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >http://www.myuniv.edu.tr/John</hasHomepage>
          </Professor>
        </isTaughtBy>
        <hasCourseID rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#string"
        >CSE450</hasCourseID>
        <isTakenBy rdf:resource="#Eric_Gra2"/>
        <hasCourseName rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#string"
        >Distributed Databases</hasCourseName>
        <hasCourseHomePage rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#string"
        >http://courses.myuniv.edu.tr/CSE450</hasCourseHomePage>
        <hasAssistant rdf:resource="#Dominic_Ass1"/>
        <hasCredits rdf:datatype=
        "http://www.w3.org/2001/XMLSchema#string"
        >3+0</hasCredits>
      </Courses>
    </takes>
  </Graduate>
</isTakenBy>
<hasCourseID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>CSE517</hasCourseID>
<hasCourseHomePage rdf:datatype=
"http://www.w3.org/2001/XMLSchema#string"
>http://courses.myuniv.edu.tr/CSE517</hasCourseHomePage>
<hasCredits rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>3+0</hasCredits>
<isTakenBy>
  <Graduate rdf:ID="Judy_Gra1">
    <takes rdf:resource="#CSE517"/>
    <hasDegree rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Master</hasDegree>
    <hasStudentID rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >501</hasStudentID>
    <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >g1@stu.myuniv.edu.tr</hasEmail>
    <hasStudentName rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >Judy</hasStudentName>
    <hasLastName rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >GLastname1</hasLastName>
    <hasStudentSurname rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string"
    >Gra1</hasStudentSurname>
  </Graduate>
</isTakenBy>
</Courses>
</assists>
<assists rdf:resource="#CSE304"/>
<hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>+90 232 9801212</hasPhoneNumber>
<hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Dominic</hasFirstName>
```

```
            <hasInterest rdf:resource="#Software_Engineering"/>
            <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >dominic@myuniv.edu.tr</hasEmail>
            <publishes rdf:resource="#Elements_of_Reusable_Object_Oriented_Software"/>
            <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Ass1</hasLastName>
            <hasHomepage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >http://www.myuniv.edu.tr/assistant1</hasHomepage>
            <assists rdf:resource="#CSE450"/>
          </Assistant>
        </isPublishedBy>
        <isPublishedBy rdf:resource="#Prof_Dr_Barbara_Moon"/>
        <hasPublicationName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Elements of Reusable Object Oriented Software</hasPublicationName>
        <hasPublicationType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Article</hasPublicationType>
        <hasPublicationYear rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >January 2007</hasPublicationYear>
        <hasTopic rdf:resource="#Software_Engineering"/>
      </Publication>
    </publishes>
    <hasHomepage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.myuniv.edu.tr/Barbara</hasHomepage>
    <teaches rdf:resource="#CSE304"/>
    <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Moon</hasLastName>
    <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >+90 232 9801777</hasPhoneNumber>
    <teaches rdf:resource="#CSE517"/>
    <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Full Professor</hasTitle>
    <hasInterest rdf:resource="#Software_Engineering"/>
    <publishes>
      <Publication rdf:ID="Programming_from_Specifications">
        <hasTopic rdf:resource="#Software_Engineering"/>
        <hasPublicationType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Conference Paper</hasPublicationType>
        <hasPublicationYear rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >June 2006</hasPublicationYear>
        <hasPublicationName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Programming from Specifications</hasPublicationName>
        <isPublishedBy rdf:resource="#Prof_Dr_Barbara_Moon"/>
      </Publication>
    </publishes>
    <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >barbara@myuniv.edu.tr</hasEmail>
    <hasInterest>
      <ResearchTopics rdf:ID="Information_Systems">
        <interestedBy rdf:resource="#Prof_Dr_Barbara_Moon"/>
        <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Information Systems</hasTopicName>
      </ResearchTopics>
    </hasInterest>
  </Professor>
</isTaughtBy>
<hasCredits rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>3+2</hasCredits>
<isTakenBy>
```

```
      <UnderGraduate rdf:ID="Neil_UGra2">
        <hasStudentID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >324</hasStudentID>
        <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >UGLastname2</hasLastName>
        <hasStudentSurname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >UGra2</hasStudentSurname>
        <takes rdf:resource="#CSE304"/>
        <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >ug2@stu.myuniv.edu.tr</hasEmail>
        <hasClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >3</hasClassNumber>
        <hasStudentName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Neil</hasStudentName>
      </UnderGraduate>
    </isTakenBy>
    <hasCourseName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Software Engineering</hasCourseName>
    <hasAssistant rdf:resource="#Dominic_Ass1"/>
    <hasCourseDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Software life cycle analysis.</hasCourseDescription>
    <hasCourseHomePage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://courses.myuniv.edu.tr/CSE304</hasCourseHomePage>
    <hasCourseID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >CSE304</hasCourseID>
    <hasAssistant rdf:resource="#Benjamin_Ass2"/>
  </Courses>
</assists>
<hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Ass2</hasLastName>
<hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>+90 232 9802222</hasPhoneNumber>
<hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Benjamin</hasFirstName>
<hasInterest>
  <ResearchTopics rdf:ID="Data_Structures">
    <interestedBy rdf:resource="#Benjamin_Ass2"/>
    <interestedBy>
      <Instructor rdf:ID="Dr_Katherina_Inst1">
        <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >+90 232 9801212</hasPhoneNumber>
        <teaches>
          <Courses rdf:ID="CSE422">
            <hasCourseHomePage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >http://courses.myuniv.edu.tr/CSE422</hasCourseHomePage>
            <isTaughtBy rdf:resource="#Dr_Katherina_Inst1"/>
            <hasCourseName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Computer Graphics</hasCourseName>
            <hasCourseID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >CSE422</hasCourseID>
            <hasCourseDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Introduction to computer graphics.</hasCourseDescription>
            <hasCredits rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >3+0</hasCredits>
          </Courses>
        </teaches>
        <hasHomepage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.myuniv.edu.tr/instructor1</hasHomepage>
```

```
            <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >katherina@myuniv.edu.tr</hasEmail>
            <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Inst1</hasLastName>
            <hasInterest>
             <ResearchTopics rdf:ID="Computer_Graphics">
              <interestedBy rdf:resource="#Dr_Katherina_Inst1"/>
              <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Computer Graphics</hasTopicName>
             </ResearchTopics>
            </hasInterest>
            <hasInterest rdf:resource="#Data_Structures"/>
            <hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Katherina</hasFirstName>
           </Instructor>
          </interestedBy>
          <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Data Structures</hasTopicName>
         </ResearchTopics>
        </hasInterest>
        <hasInterest rdf:resource="#Relational_Databases"/>
        <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >benjamin@myuniv.edu.tr</hasEmail>
       </Assistant>
      </hasAssistant>
      <isTaughtBy>
       <Professor rdf:ID="Ass_Prof_Dr_Edward_Young">
        <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >edward5@myuniv.edu.tr</hasEmail>
        <teaches>
         <Courses rdf:ID="CSE403">
          <hasCourseID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >CSE403</hasCourseID>
          <hasCourseHomePage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://courses.myuniv.edu.tr/CSE403</hasCourseHomePage>
          <hasCourseDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Overview of networks.</hasCourseDescription>
          <hasCourseName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Computer Networks</hasCourseName>
          <hasCredits rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >3+0</hasCredits>
          <isTaughtBy rdf:resource="#Ass_Prof_Dr_Edward_Young"/>
         </Courses>
        </teaches>
        <hasInterest>
         <ResearchTopics rdf:ID="Cryptography">
          <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Cryptography</hasTopicName>
          <interestedBy rdf:resource="#Ass_Prof_Dr_Edward_Young"/>
         </ResearchTopics>
        </hasInterest>
        <hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Edward</hasFirstName>
        <hasTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Assistant Professor</hasTitle>
        <teaches rdf:resource="#CSE100"/>
        <publishes>
         <Publication rdf:ID="Introduction_to_Distributed_Communications_Networks">
```

```
        <hasTopic>
          <ResearchTopics rdf:ID="Computer_Networks">
            <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Computer Networks</hasTopicName>
            <interestedBy rdf:resource="#Ass_Prof_Dr_Edward_Young"/>
          </ResearchTopics>
        </hasTopic>
        <hasTopic rdf:resource="#Distributed_Systems"/>
        <hasPublicationType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        ></hasPublicationType>
        <hasPublicationType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Book</hasPublicationType>
        <isPublishedBy rdf:resource="#Ass_Prof_Dr_Edward_Young"/>
        <hasPublicationYear rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >January 2006</hasPublicationYear>
        <hasPublicationName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Introduction to Distributed Communications Networks</hasPublicationName>
      </Publication>
    </publishes>
    <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >+90 232 9801990</hasPhoneNumber>
    <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Young</hasLastName>
    <hasInterest rdf:resource="#Computer_Networks"/>
    <hasInterest>
      <ResearchTopics rdf:ID="Network_Security">
        <interestedBy rdf:resource="#Ass_Prof_Dr_Edward_Young"/>
        <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Network Security</hasTopicName>
      </ResearchTopics>
    </hasInterest>
    <hasHomepage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.myuniv.edu.tr/Edward</hasHomepage>
  </Professor>
</isTaughtBy>
<isTakenBy rdf:resource="#Jason_UGra4"/>
<isTakenBy>
  <UnderGraduate rdf:ID="Bob_UGra1">
    <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >UGLastname1</hasLastName>
    <hasStudentName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Bob</hasStudentName>
    <hasStudentSurname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >UGra1</hasStudentSurname>
    <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ug1@stu.myuniv.edu.tr</hasEmail>
    <hasStudentID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >123</hasStudentID>
    <hasClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1</hasClassNumber>
    <takes rdf:resource="#CSE100"/>
  </UnderGraduate>
</isTakenBy>
<hasAssistant rdf:resource="#Yancy_Ass3"/>
<hasCourseDescription rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Use of windows, spreadsheet programs, and graphics.</hasCourseDescription>
<hasCourseName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Introduction To Computers</hasCourseName>
```

```
      <hasCourseID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >CSE100</hasCourseID>
      <hasCourseHomePage rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://courses.myuniv.edu.tr/CSE100</hasCourseHomePage>
    </Courses>
  </takes>
  <hasStudentID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >124</hasStudentID>
  <hasClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >1</hasClassNumber>
</UnderGraduate>
<NonAcademicStaff rdf:ID="Calvin_Non2">
  <hasJobTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Technical Staff</hasJobTitle>
  <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >calvin@myuniv.edu.tr</hasEmail>
  <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Non2</hasLastName>
  <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >+90 232 9801010</hasPhoneNumber>
  <hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Calvin</hasFirstName>
</NonAcademicStaff>
<UnderGraduate rdf:ID="Leon_UGra5">
  <hasStudentID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >257</hasStudentID>
  <hasClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >2</hasClassNumber>
  <hasStudentName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Leon</hasStudentName>
  <hasStudentSurname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >UGra5</hasStudentSurname>
</UnderGraduate>
<UnderGraduate rdf:ID="John_UGra3">
  <hasStudentID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >256</hasStudentID>
  <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ug3@stu.myuniv.edu.tr</hasEmail>
  <hasStudentSurname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >UGra3</hasStudentSurname>
  <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >UGLastname3</hasLastName>
  <hasClassNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >2</hasClassNumber>
  <hasStudentName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >John</hasStudentName>
</UnderGraduate>
<NonAcademicStaff rdf:ID="Julia_Non1">
  <hasFirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Julia</hasFirstName>
  <hasEmail rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >julia@myuniv.edu.tr</hasEmail>
  <hasLastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Non1</hasLastName>
  <hasJobTitle rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Secretary</hasJobTitle>
  <hasPhoneNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >+90 232 9800000</hasPhoneNumber>
```

```
  </NonAcademicStaff>
  <ResearchTopics rdf:ID="Wireless_Networks">
    <hasTopicName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Wireless Networks</hasTopicName>
  </ResearchTopics>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355)  http://protege.stanford.edu -->
```