# ONTOLOGY LEARNING AND QUESTION ANSWERING (QA) SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELTEM BAŞKURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

APRIL 2010

Approval of the thesis:

**ONTOLOGY LEARNING AND QUESTION ANSWERING (QA) SYSTEMS**

submitted by **MELTEM BAŞKURT** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Ferda Nur Alpaslan
Supervisor, **Computer Engineering Dept., METU** _____

**Examining Committee Members:**

Prof. Dr. Mehmet R. Tolun
Computer Engineering Dept., Çankaya University _____

Assoc. Prof. Dr. Ferda Nur Alpaslan
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering Dept., METU _____

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU _____

Asst. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU _____

**Date:** _____

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name: MELTEM BAŞKURT

Signature :

## ABSTRACT

ONTOLOGY LEARNING AND QUESTION ANSWERING (QA) SYSTEMS

Başkurt, Meltem
M.Sc., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ferda Nur Alpaslan

April 2010, 81 pages

Ontology Learning requires a deep specialization on Semantic Web, Knowledge Representation, Search Engines, Inductive Learning, Natural Language Processing, Information Storage, Extraction and Retrieval. Huge amount of domain specific, unstructured on-line data needs to be expressed in machine understandable and semantically searchable format. Currently users are often forced to search manually in the results returned by the keyword-based search services. They also want to use their native languages to express what they search. In this thesis we developed an ontology based question answering system that satisfies these needs by the research outputs of the areas stated above. The system allows users to enter a question about a restricted domain by means of natural language and returns exact answer of the questions. A set of questions are collected from the users in the domain. In addition to questions, their corresponding question templates were generated on the basis of the domain ontology. When the user asks a question and hits the search button, system chooses the suitable question template and builds a SPARQL query according to this template. System is also capable of answering questions required inference by using generic inference rules defined at a rule file.

Our evaluation with ten users shows that the sytem is extremely simple to use without any training resulting in very good query performance.

Keywords: Description Logics, Ontology, Question Answering, Reasoning.

# ÖZ

ONTOLOJİ ÖĞRENME VE SORU CEVAPLAMA SİSTEMLERİ

Başkurt, Meltem
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ferda Nur Alpaslan

Nisan 2010, 81 sayfa

Ontoloji Öğrenme; Anlamsal Web, Bilgi Gösterimi, Arama Motorları, Tümevarımlı Öğrenme, Doğal Dil İşleme, Bilgi Depolama, Bilgi Çekme konularında derin bir uzmanlaşma gerektirir. Çok büyük miktardaki alana özel, yapısal olmayan bağlantılı verinin makinaların anlayabileceği ve anlamsal olarak ulaşabileceği bir formatta ifade edilmesi gerekmektedir. Şu anki kullanıcılar anahtar sözcüğe dayalı arama servislerinden dönen sonuçlar içinde elle aramaya zorlanılıyor. Ayrıca aradıkları şeyi kendi anadillerini kullanarak ifade etmek istiyorlar. Bu tezde, belirtilen ihtiyaçları yukarda bahsedilen alanların araştırma sonuçları dogrultusunda karşılayacak bir sistem geliştirdik. Sistem kullanıcıların kısıtlı bir domain hakkında doğal dilde soru sormalarını sağlıyor ve soruların tam cevabını dönüyor. Domain de bulunan kullanıcılardan soru seti toplandı. Bu sorulara ek olarak, domain ontolojisi baz alınarak ilgili soru şablonları oluşturuldu. Kullanıcı soruyu yazıp ara tuşuna basınca, sistem uygun şablonu seçip doğal dildeki soruyu bu şablona uygun olarak SPARQL sorguya çevirir. Sistem çıkarım gerektiren sorulara da kural dosyasında tanımlı genel kuralları kullanarak cevap verebilmektedir. On kullanıcı ile yaptığımız değerlendirme gösteriyor ki, sistem herhangi bir eğitime gerek kalmadan kullanımı oldukça basit ve çok iyi sorgulama performansı ile sonuçlanan bir sistemdir.

Anahtar Kelimeler: Tanımlama Mantığı, Ontoloji, Soru Cevaplama, Çıkarım.

*To My Family*

## ACKNOWLEDGMENTS

Firstly, I would like to thank to my thesis supervisor Assoc. Prof. Dr. Ferda Nur Alpaslan for her guidance, support and motivation she provided throughout my research.

I also like to thank to Özgür Alan for his guidance about the Ontology Reasoning and Experimentation parts in my study.

I would also like to thank all the jury members for their valuable suggestions and comments on this thesis.

Finally, I would like to thank my mom, my dad and my brother for everything. Thanks to my husband for his understanding during my thesis study.

## TABLE OF CONTENTS

## LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| NLP | Natural Language Processing |
| NL | Natural Language |
| QA | Question Answering |
| SPARQL | SPARQL Protocol and RDF Query Language |
| KMS | Knowledge Management System |
| KB | Knowledge Base |

# CHAPTER 1

# INTRODUCTION

## 1.1 Statement of the problem

Question Answering is an information retrieval form where its purpose is to find an exact answer to a natural language question rather than finding documents or text passages, which match the question. In order to find an acceptable answer to the natural language question, question answering systems combine different techniques like document retrieval, answer analysis and extraction together. The ultimate aim of Question Answering is extracting the exact answers by processing large volumes of open domain texts (documents extracted from the World Wide Web) without any restriction on both question and the documents to be searched for. However; this is inherently a hard task because without a restriction imposed either on the question type or on the user's vocabulary, the question answering process gets a big hit even at the question interpretation phase. This is why; most of the efforts are focused on answering "factoid style" questions, since it is much more efficient to use thorough text processing algorithms based on pattern extraction or information retrieval techniques.

Speaking in Description Logics terms [3], the ontology has provided the basic classes, their properties and their relations between themselves as TBoxes. Moreover, the individuals, which are initializations of the classes aforementioned, are represented as ABoxes. With this approach, the question answering mechanism simply becomes as a set of reasoning tasks over the ontology. The aim of this work is constructing a question answering system using document ontology. The ontology to be used contains both the information semantics in the shape of TBoxes

and the answers to the questions as ABoxes. The question answering system provides an authoring environment which facilitates content sharing by automatically tagging content with semantic metadata and by using open standards to store it in networked repositories supporting symbolic and similarity-based indexing and search capabilities for all content types.

## 1.2 Objective of the study

The aim of this study is to present an automated question answering system for the software test document domain. The system is intended to provide exact answer to the questions entered by the members of the software test team at Vodafone.

The thesis proposes an algorithm which converts a natural language question into an ontology based search engine query. Moreover the system enhances querying capabilities by using domain ontology knowledge. A document ontology is designed for this purpose. A small number of software test document has been used for populating the ontology. These documents were related with the software test prosedures performed at Vodafone. For the query conversion, the idea of the proposed algorithm is to firstly determine the question structure. After performing the steps in this phase, the algorithm determines the SPARQL triples (class, property, value) and performed the related steps of this phase. As a last phase, the query building steps performed and related SPARQL queries for each question template are generated.

The contribution of this study lies in using the proposed algorithm as a basis for the ontology based question answering system for document domain.

## 1.3 Organization of the thesis

The next chapter gives the background information used in the thesis. In Chapter 3 the previous and related work is presented. Chapter 4 presents the system architecture and gives information about the modules of the system. Chapter 5 defines the implementation and experimentation details of an ontology based

question answering system on a case problem from a real-life application in test document domain. Eventually, Chapter 6 summarizes and concludes the thesis.

# CHAPTER 2

# BACKGROUND

In this chapter, the background information is given for the topics relevant to thesis study. The major related topics are listed as *Description Logics, Ontology, OWL Ontology Language, A Knowledge Engineering Methodology for Ontology Development, Ontology Reasoning, Ontology-based Question Answering Systems and Wordnet*.

## 2.1 Description Logics

Baader et al. [3] defines the DLs as a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way.

DLs are based on the notions of concepts (i.e. unary predicates) and properties (i.e. binary relations). Using different constructors, complex concepts can be built up from simple ones. With its unambiguous semantics, DLs lend themselves to powerful reasoning algorithms which can automatically classify and memoise the domain knowledge to provide a cache for further inderences, according to Hu et al. [11].

In Description Logic, the knowledge base is represented in a tuple (T-Box, A-Box). A-Box contains assertions about individuals and T-Box defines the concepts (classes or types of instances), roles (predicates), and features (attributes/properties) of these instances. Fig 2.1 shows the architecture of a knowledge representation system based on Description Logics.

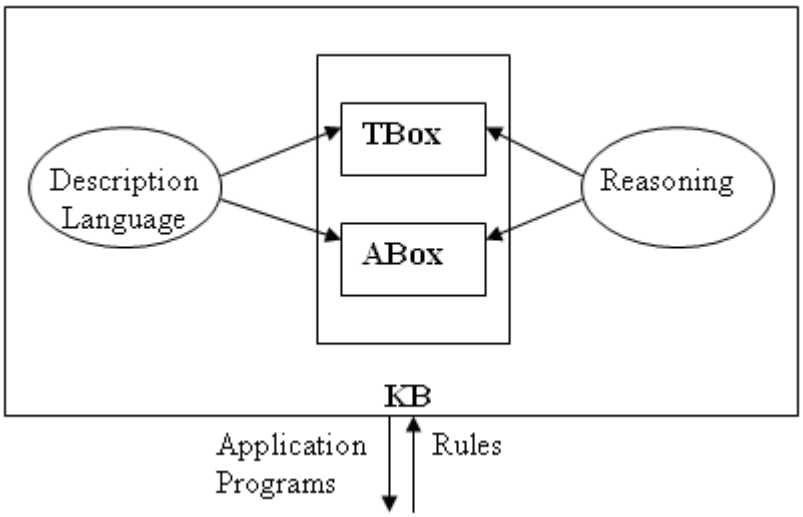


Figure 2.1: Architecture of a knowledge representation system based on Description Logics.

## 2.2 Ontology

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them according to Noy et al. [8]. Apart from subclass relationships, ontologies may include information such as

- properties (X teaches Y)
- value restrictions (only faculty members can teach courses)
- disjointness statements (faculty and general staff are disjoint)
- specification of logical relationships between objects (every department must include at least ten faculty members)

Ontologies are heavily used by semantic web for the purpose of comprehensive and transportable machine understanding. It is mainly related with the definition of metadata. Metada annotations are very important for the agents to reach the

resources effectively. The effect of ontology appears here as a construction that consists of the descriptions of concepts and their properties in the specified domain.

### 2.2.1 OWL Ontology Language

The hardest part of an Automated Question Answering application dealing with World Wide Web can be identified as comprehending the question that was asked. Without a reasonable idea over the question, the system is deemed to be unsuccessful from the very beginning. Mere syntactic aspect of the question and answer domain is insufficient at best. To get both what has been asked and where to be searched, a common ground must be set for representing the semantic layer of the domain. OWL[1] exactly tries to fill that gap.

An OWL ontology, simply, includes the classes, the properties that may have and the relations between the classes. The entailed consequences can be derived by applying the OWL formal semantics.

OWL has three increasingly-expressive sublanguages:

- OWL-Lite
- OWL-DL
- OWL-Full

Among them, OWL Lite is a subset of OWL DL, and OWL DL is a subset of OWL Full. OWL-DL is much more expressive than OWL-Lite and is based on Description Logics. Because Description Logics are a decidable fragment of First Order Logic, they are amenable to automated reasoning. It is therefore possible to automatically compute the subsumption reasoning and check for inconsistencies in an ontology that conforms to OWL-DL.

[1] W3C OWL, http://www.w3.org/TR/owl-guide/

### 2.2.2 A Knowledge Engineering Metholody for Ontology Development

Noy and McGuinness et al. [8] states that there is no one correct way to model a domain. There are always valid alternatives. They also state that Ontology development is necessarily an iterative process that is there will be almost certainly need to revise the initial ontology. Process of iterative design will likely continue through the entire lifecycle of the ontology.

Noy and McGuinness et al. [8] offers a simple Knowledge-Engineering Methodology for developing an ontology. The steps of this methodoly are listed as below:

- *Determining the domain and scope of the ontology*: In this step, several basic questions about the domain should be answered to define the scope of the model.
- *Considering reusing existing ontologies*: This step states that it's worth considering to check what someone else done and if existing sources can be refined and extended.
- *Enumerating important terms in the ontology*: In this step, an extensive list of terms should be prepaid without worrying about overlap between concepts they represent, relations among the terms, or any properties that the concepts may have, or whether the concepts are classes or slots.
- *Defining the classes and the class hierarchy*: In this step, classes in the domain are defined firstly. After that, a class hierarchy is developed using one of the three approaches (top-down, bottom-up,combination)
- *Defining the properties of classes—slots*: In this step, properties of classes are defined once some of the classes is defined. These properties become slots attached to classes.
- *Defining the facets of the slots*: In this step, facets of the slots are defined. Facets describe the value type, allowed values, the number of the values (cardinality) and so on.

- *Creating instances*: The last step is creating individual instances of classes in the hierarchy.

### 2.2.3 Ontology Reasoning

Given an ontology, an additional statement can be deduced from the ontology if it is true in all the models of the ontology. So, ontology reasoning is the process of deriving valid deductions from an ontology. If the typical statement of ontology language is considered, the following deductions can be introduced:

- *Class membership* deduces whether an object is an instance of a class. For example, if in the ontology it is stated that Mehmet Öz is an instance of the class specialists, and that specialists is a subclass of the doctors class, then it can be inferred that Mehmet Öz is an instance of staff, because this latter statement is true in all the models of the ontology.
- *Classification* deduces all subclass relationships between the existing classes in the ontology. For example, if in the ontology it is stated that the class specialists is a subclass of the class doctors, and that class doctors is a subclass of the class staff, then it can be inferred that specialists class is a subclass of staff class.
- *Equivalence of classes* deduces whether two classes are equivalent. For example, if specialists class is equivalent to doctors class and doctors class is equivalent to staff class, then specialists class is equivalent to staff class.
- *Consistency of a class* checks that some class does not have necessarily an empty extension. For example, given an ontology in which the class doctor-patients is defined to be a subclass of two disjoint classes doctors and patients, it can be inferred that the class doctor-patients is inconsistent, since in every model of the ontology its extension is empty.
- *Consistency of the ontology* checks that the ontology admits at least a model. That is, there is at least a possibility to have an instantiation of the domain

compatible with the ontology. For example, when it is declared in the ontology:

1. Ahmet is an instance of both doctors and patient classes
2. Doctors and patients are two disjoint classes.

Then, there exists an inconsistency because the two constraints can not be satisfied simultaneously. This clearly indicates that there is an error in the ontology, since it does not represent any possible situation.

## 2.3 Ontology-based Question Answering Systems

Guo et al. [9] states that Semantic web and ontology are the key technologies of Question Answering system. Ontology is becoming the pivotal methodology to represent domain-specific conceptual knowledge in order to promote the semantic capability of a QA system.

Question Answering (QA) systems are able both to understand questions in natural language and to produce answers in the form of selected paragraphs extracted from very large collections of text. Generally, they are opendomain systems, and do not rely on specialised conceptual knowledge as they use a mixture of statistical techniques and shallow linguistic analysis. Ontological Question Answering systems propose to attack the problem by means of an internal unambiguous knowledge representation. As any knowledge intensive application, ontological QA systems have as intrinsic limitation related to the small scale of the underlying syntactic-semantic models of natural language.

Although its limitations, Lee et al. [10] states that as a new approach for QA, ontology-based QA system has the following advantages other than improved performance.

- offering additional information about an answer,
- providing measures of reliability,

- explaining how the answer was derived

## 2.4 WordNet

Miller et al. [5] states that WordNet is an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept.

The meanings can be thought as synonym sets. In other words, each synonymy word is treated as one set. Each synonym set is related to other synonym sets in different ways. What WordNet does is representing these sets and relations. The relations are the pointers between the synonym sets. Some of these relations are symmetric while some others are not. There may be many relations to be discussed; yet the scope of this thesis limited us to use the only one explained below.

**Synonymy** To construct the synonymy sets, we need the basic relation, synonymy, to gather the words having the same meaning as one set. One of the definitions of synonymy states that, *two expressions are synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made.*

# CHAPTER 3

# RELATED WORK

Throughout this chapter, relevant researches and works that helped us in this work will be presented.

## 3.1 Question Answering Systems

Question Answering (QA) System is a type of information retrieval system. These systems are able both to understand questions in natural language and to produce answers in the form of selected paragraphs extracted from very large collections of text. Generally, they are open-domain systems, and do not rely on specialised conceptual knowledge as they use a mixture of statistical techniques and shallow linguistic analysis. On the other hand, these systems usually have much more data available from which to extract the answer. Alternatively, closed-domain question answering systems deals with questions under a specific domain where only a limited type of questions are accepted.

In order to find an acceptable answer to the natural language question, questions answering systems tend to use several approaches like worldwide web, ontology or shallow parsing as the information source.

In the following two sub chapters, the general architecture of question answering systems and ontology based questions answering systems will be described.

### 3.1.1 General Architecture

In order to find an answer to a question in a data source, question answering systems run some common processes, which constitute the general structure of

question answering systems. The four common processes of question answering are *Question Processing, Document Processing, Answer Extraction and Answer Formulation* as seen in Figure 3.1.

The Question Processing is for determining the expected answer type of the given question. After the natural language question is entered into question answering system, the question is analyzed and categorized into its type. For example, suppose the question is “Who is the prime minister of Turkey?” The expected answer type is a PERSON. So, while searching through the document collection, the system indeed looks for PERSONs.

During the Document Processing, candidate sections or paragraphs from the document collection is prepared. This is accomplished using search engines to identify the documents or paragraphs in the document set that are likely to contain the answer. Immediately after, a filter preselects small text fragments that contain strings of the same type as the expected answer. For example, if the question is "Where is the capital city of Turkey" the filter returns text that contains names of cities.

Answer Extraction is the phase where the question answering system tries to extract candidate answers from the documents, which are retrieved in the document phase. In order to prepare the list of candidate answers, question answering system uses various methods.

Answer Formulation provides to present the answer in a way as natural as possible. In some cases, simple extraction is sufficient. For example, when the question classification indicates that the answer type is a name, a quantity (monetary value, length, size, distance, etc.) or a date (e.g. the answer to the question "On what day did establish the Republic of Turkey?") the extraction of a single data is adequate. For other states, the presentation of the answer requires the use of various methods.

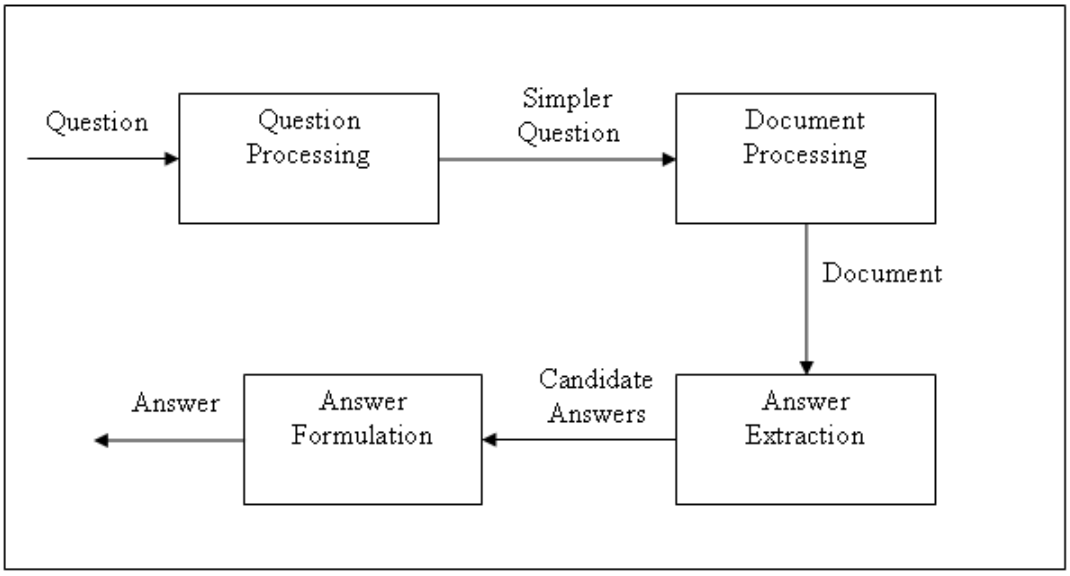


Figure 3.1: General Architecture of Question Answering Systems.

## 3.2 Ontology-based Question Answering Systems

Due to their powerful knowledge representation formalism and associated inference mechanisms, ontology-based systems are emerging as a natural choice for the next generation of Kowledge Management Systems (KMSs).

One of the most crucial parts of an ontology-based QA system is the conversion of the natural language question into an ontology based query. A number of studies have been presented to define precise methods for generating a semantic query for ontology based question answering systems.

### 3.2.1 AquaLog

Lopez et al. [1] introduced an ontology portable QA system which takes queries expressed in natural language and ontology as input and returns answers drawn from the available semantic markup as shown in Figure 3.2. The system makes use of linguistic tools (e.g. GATE) and resources (e.g. WordNet) to annotate the terms

and relations in a natural language question and then translates the question into set of <subject, predicate, object> triples.


Figure 3.2: A screenshot from AquaLog.

These intermediate triples are further processed to produce ontology-compliant logical queries for drawing the answers from a knowledge base with ontology-compliant semantic markup.

### 3.2.2 CHESt

A new e-Learning tool named CHESt (Computer History Expert System) has been presented by Linckels et al. [2]. The semantic search engine of the system gets a NL question from the user and maps it to a general assertion. To make it possible, it is firstly looked for semantically important words and translated them into RDF. In this study [2], a specific domain dictionary is used to retrieve the semantics for every word in the sentence. Semantically unnecessary words like {what, did} or too general words like {operating, system} are ignored. After that, this transformed

question is mapped to a general assertion. And finally based on that interpretation, an RDQL query is generated and started against the knowledge base.

### 3.2.3 ONTOSEARCH2

A Semantic Web engine, called ONTOSEARCH2, has been presented by Pan et al. [7]. The engine searches and queries web ontologies by creating and storing a copy of ontologies in tractable description logic. ONTOSEARCH2 allows formal querying of its repository, including both the structures and instances of ontologies, using the SPARQL query language. An example scene from the engine is given in Figure 3.3.

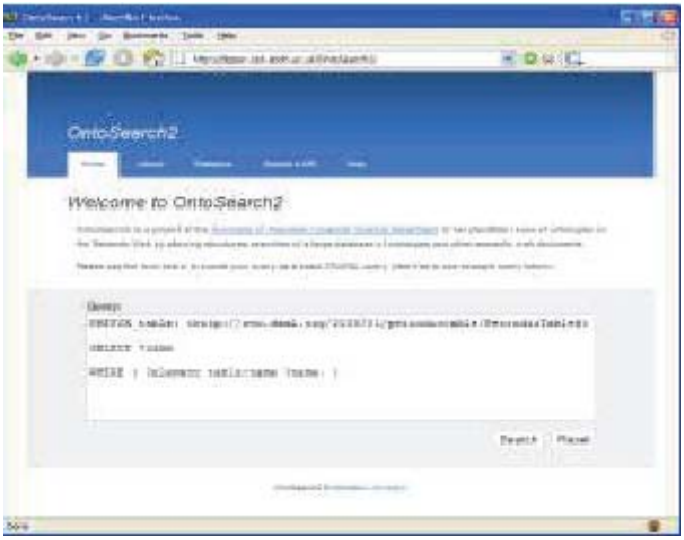


Figure 3.3: A screenshot from ONTOSEARCH2.

### 3.2.4 Querix

Kaufmann et al. [4] introduced a natural language interface to semantic web querying where the interface allows formulating queries based on Clarification Dialogs if ambiguities occur. The system allows users to enter question in NL and

choose the ontology to be queried. When the user enters a question, a SPARQL query is generated. An example scene from the interface is given in Figure 3.4.

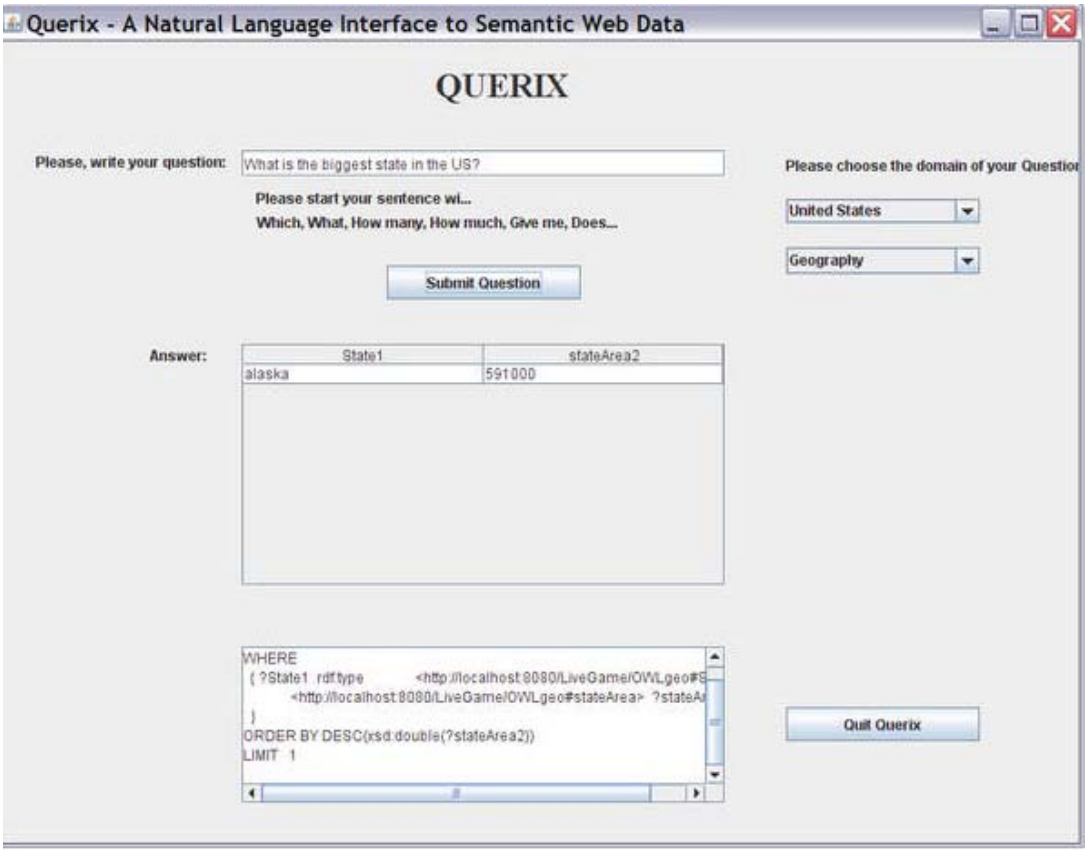


Figure 3.4: A screenshot from Querix.

Kaufmann et al. [4] states that Querix does not claim to be ”intelligent” by interpreting and understanding the input queries; it employs a reduced set of NLP tools and consults the user when hitting its limitations.

### 3.2.5 Ginseng

Another study has been introduced by Bernstein et al. [12]. Ginseng is a Guided Input Natural Language Search Engine. It allows users to query any Semantic Web

knowledge base using a guided input. When the user starts to enter question, it guess the possible completions of what the user enters and presents the user with a choice popup box as shown in Figure 3.5.

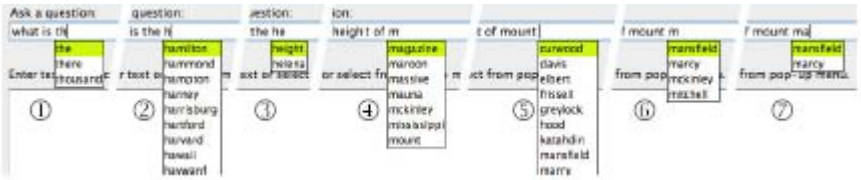

Figure 3.5: Ginseng query-completion popup window.

When a question is completed, Ginseng translates it into a RDQL query and displays the result to the user as shown in Figure 3.6.

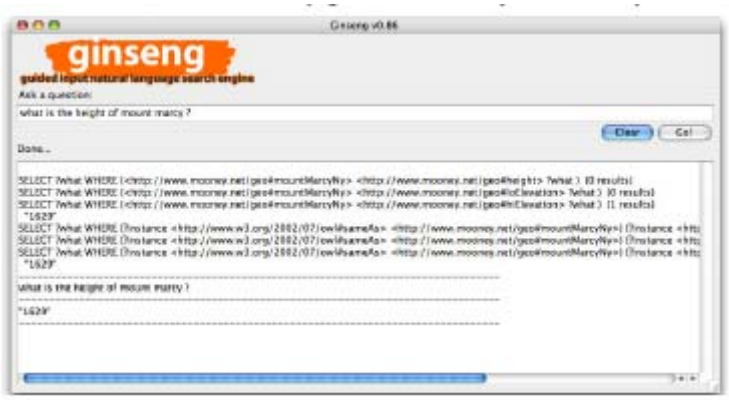


Figure 3.6: A screenshot from Ginseng.

# CHAPTER 4

# SYSTEM OVERVIEW

In this chapter, the general architecture of the Ontology based Question Answering System for Software Test Documents Domain will be described.

## 4.1 The Architecture of the System

Figure 4.1 shows the architecture of the system. It consists of three major parts: User Interface Module, Storage Module, and Inference Module.

The User Interface Module allows the user to enter full NL queries. After executing a query, it displays the results to the user. The Storage Module is a repository that stores the ontology. The Inference Module based on Jena API provides an ontology-based search or inference mechanism for the most appropriate answer.

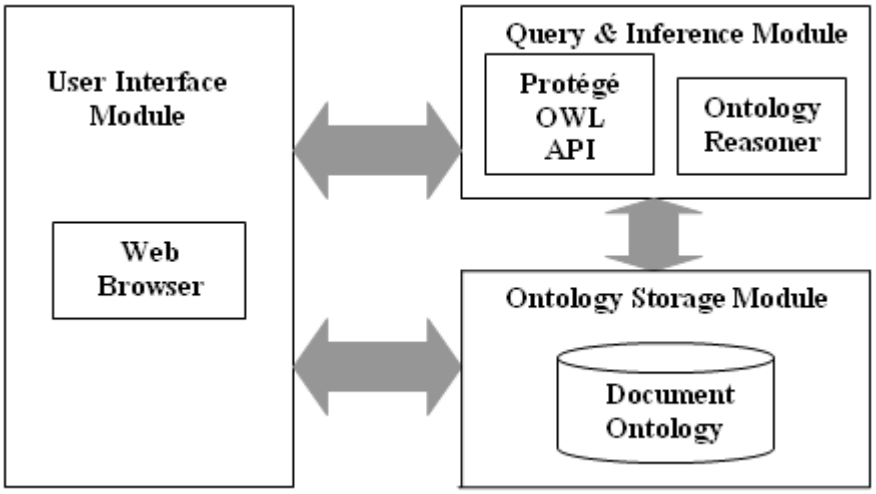


Figure 4.1: System Architecture.

The main flow of action in the system is initiated by the user request, which is posted from the StartSearch.html as seen in Fig. 4.2.

After the initiation, question is sent to StartSearch () method of the Searcher Servlet class. This method firstly calls for loadOntology () method of the Ontomapper.java class for loading the domain ontology.

Immediately after the ontology is loaded, Searcher.StartSearch method calls for getSearchQuery () method of QueryProcessor.java class for building of the query. While building the query, getShortQuestion () method of the Sentence.java class is called. This class analyzes the structure of the question sentence and tries to distinguish the class, property, value and criteria components in the sentence. It also simplifies the sentence by keeping out the words of the sentence except question sentence types and auxiliary verbs.

As a next step, the application decides if an inference is required or not to be able to find the answer of the question. If an inference is required, runEngine () method of the Inference class is called. For inferencing, rule file and owl ontology file is given into the inference engine. After making inferencing, the result is sent to the StartSearch.html for users to be able to see the answer of the questions. If an inferencing is not required, simplified question sentence is sent back to QueryProcessor class. QueryProcessor converts the question into a SPARQL query and pass it to Searcher class. As soon as the question is arrived to Searcher class, StartSearch method of the class calls for the executeQuery () method of the OntoMapper class. After the query is executed here, the answer is sent to the StartSearch.html for users to be able to see the answer of the questions.

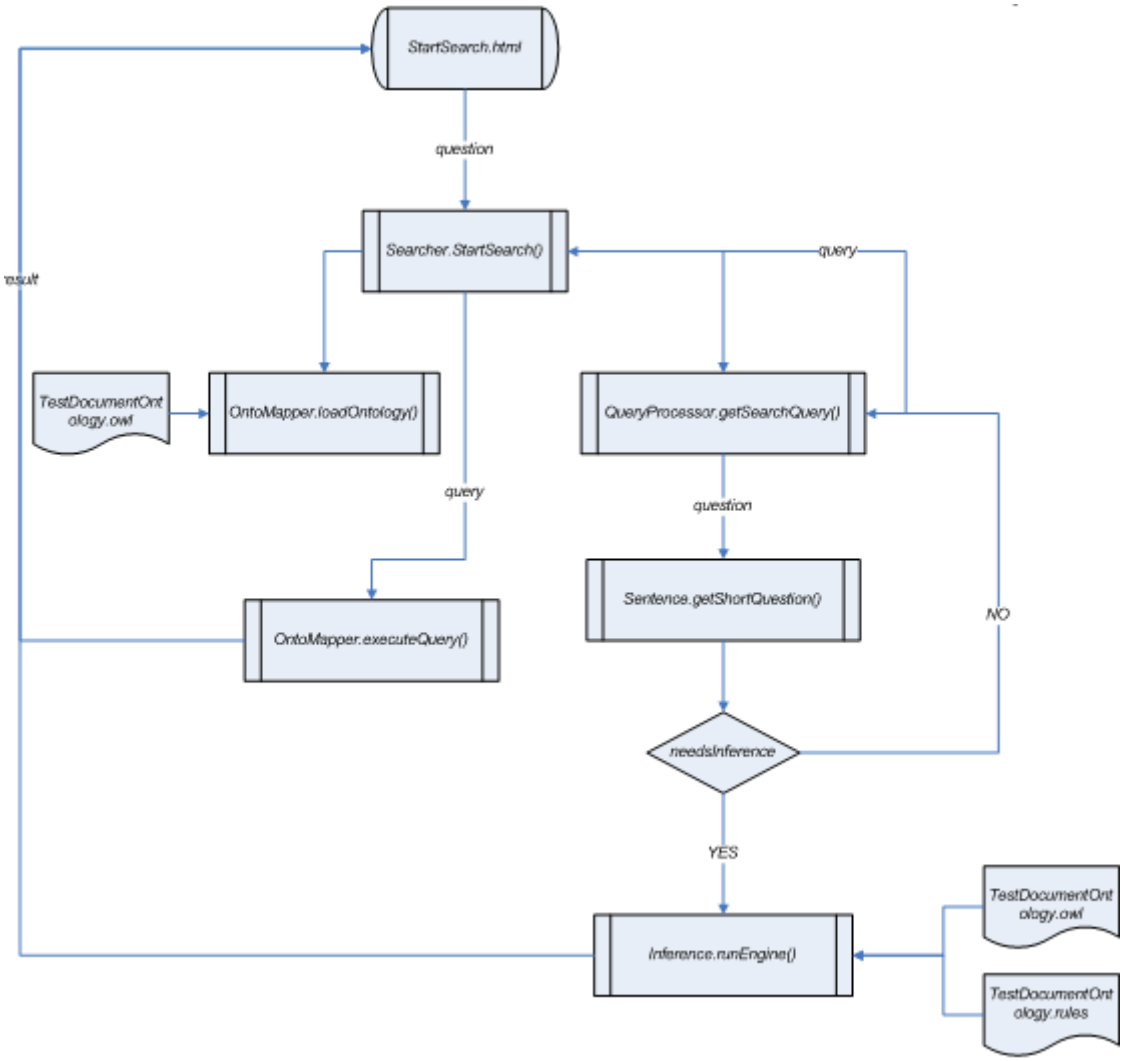


Figure 4.2: Flowchart of the system.

## 4.2 The Algorithm

The algorithm at first analyzes the type of the question sentence by searching for the keywords like which, who, what etc. in the question sentence. In case of not defining the type of the question, it's decided to run the Jena's Inference engine. At this phase, words in the sentence like "MAX" and "MIN" are also controlled and these informations are kept in the Java code.

As a next step, question is splitted word by word and it's started to examine each word in order. At first, it's checked whether the word is among the defined rules or not. For example if *nominate* word is encountered in the sentence, it's decided to run the Inference engine since there is a rule definition with this name.

While the words are processing particularly, for each of the word the class list is controlled to define the SPARQL triple. If the word does not exist in the class list, alternative words from Wordnet are found. If the class is found, the next word is taken consideration. In case of not finding the class, the word is looked for in the property list. If the word is not found in the property list, alternative words are found from Wordnet. If the word is still not found, it's kept as value and the next word is taken consideration.

At the end of this algorithm if the property is not found, it is decided that the wrong class is selected and the search process is again started but with different method. Except the word which was previously selected as class, each word firstly is looked for in the Class list. If the alternative class is found, searching process is started again and at this time each word is looked for in the Property list. When it's found in the Property list, it's passed to the third phase which is query building phase.

In query building firstly it's looked to the question type and builded queries using related question template components (class, property, value) according to question type. At this phase, correct query is tried to be built making controls like hasSubClasses for classes and isIndividual for values. If the value contains more than one word, alternative queries also are built to increase the possibility of returning the correct result. For example for a value containing three words, below possibilities are used and seperate queries are built for each one.

```
Value = words.get(2) +" "+ words.get(1) +" "+
words.get(0);//Replace word locations

      Value = words.get(1) +" "+ words.get(2);//Remove first word

      Value = words.get(0) +" "+ words.get(1);//Remove last word
```

```
Value = words.get(2);//Keep last word
Value = words.get(1);//Keep second word
Value = words.get(0);//Keep first word
```

Each of the built alternative queries are run untill the answer is found and the founded result is written to the screen.

In the application, there are five java classes which are Searcher.java, OntoMapper.java, Sentence.java, QueryProcessor.java and Inference.java classes.

After the question sentence is simplified by Sentence class; firstly class component, secondly property component and lastly value component is tried to find by seperating the simplified question sentence word by word. Remaining words in the simplified sentence is respectively looked for in the class list which is formed at Ontomapper class. First of all an exact matching is performed. In case of not finding the word in the class list, classes containing that word are looked for. In Figure 4.3 *isClass* method of QueryProcessor class is shown.

```
    private static String isClass(String varWord)
    {
        for(int j = 0; j < OntoMapper.classList.size(); j++){
           String cls = (String) OntoMapper.classList.get(j);
            if (cls.toUpperCase().substring(cls.indexOf("#") +
1).compareTo(varWord.toUpperCase()) == 0 ) {
                  return cls.substring(cls.indexOf("#") + 1);
            }
        }
      //If could not find with exact match
        for(int j = 0; j < OntoMapper.classList.size(); j++){
            String cls = (String) OntoMapper.classList.get(j);
             if (cls.toUpperCase().substring(cls.indexOf("#") +
1).indexOf(varWord.toUpperCase()) > -1 ) {
                   return cls.substring(cls.indexOf("#") + 1);
             }
         }
       return "";
    }
```

Figure 4.3: Method for searching class component in class list.

If still there is no matching, the application connects to WordNet 2.1[2] to map wanted word to a lexical item in a WordNet synset. After that, this lexical item is looked for in the class list. In Figure 4.4, *getAlterWord* method of QueryProcessor class is shown.

```
private static String[] getAlterWord(String varWord)
    {
      String[] wordForms = null;

         Synset[] synsets = wordDatabase.getSynsets(varWord.toLowerCase());
         //  Display the word forms and definitions for synsets retrieved
         if (synsets.length > 0)
         {
                 //System.out.println("The following synsets contain '"
+varWord + "' or a possible base form of that text:");
                 for (int i = 0; i < synsets.length; i++)
                 {
                     wordForms = synsets[i].getWordForms();
                 }
         }

         return wordForms;
          }
```

Figure 4.4: Method for getting synset words from WordNet.

If a matching class is found then it's returned and called *isProperty* method of QueryProcessor class to look for property by taking the next word in the question sentence. In case of not finding a matched class, it's started to look for in property list continuing searching with same word. Even if the word is not found in property list, it's accepted as a value. When there exists more than one value in the question sentence, they are appended repeatedly and looked for all together as a value. If a matching record cannot be found, then it's looked for separately from each other by dividing these values.

[2] WordNet 2.1 : http://wordnet.princeton.edu/wordnet/download/

```
    private static String isProperty(String varWord)
    {
        for(int j = 0; j < OntoMapper.propertyList.size(); j++){
           String property = (String) OntoMapper.propertyList.get(j);
            if (property.toUpperCase().compareTo(varWord.toUpperCase()) ==
0 ) {
                 return property;
            }
        }
        //If could not find with exact match
        for(int j = 0; j < OntoMapper.propertyList.size(); j++){
            String property = (String) OntoMapper.propertyList.get(j);
             if (property.toUpperCase().indexOf(varWord.toUpperCase()) > -1
) {
                  return property;
             }
         }
       return "";
          }
```

Figure 4.5: Method for searching property component in property list.

When the user asks a question that requires inference, the application creates reasoner from Inference class. After that, it runs the Inference Engine giving rule file and owl ontology into the inference engine.

For example for the question "*Show the qualified persons that has at least three years of experience*", a new ontology model is created immediately after the reaosoner runs over the ontology. Then SPARQL query is run over the new ontology model by using a generic rule defined at TestDocumentOntology.rules file.

# CHAPTER 5

# IMPLEMENTATION AND EXPERIMENTATION

This chapter describes the implementation and experimentation details of the Ontology-based Question Answering System for Software Test Documents Domain.

## 5.1 Tools and Technologies used

Before delving into the details of the implementation, the introduction of the tools and technologies being used will be appropriate.

### 5.1.1 Jena2 Inference Engine - Generic Rule Reasoner

Jena2 Inference Engine[3] allows a range of inferences engines or reasoners to be plugged into Jena. Generic Rule Reasoner is one of the predefined reasoners included in the Jena distribution. It is a rule based reasoner that supports user defined rules. In the thesis, a list of rule is defined at a rule file. OWL ontology file and the rule file are given to the inference engine to make a new model from the ontology. In the thesis, Jena's Generic Rule Reasoner is used to infer knowledge about models from the document ontology.

### 5.1.2 Jena Ontology API

Using the guidelines in Jena Ontology API [4] for developing functions for accessing the reasoner server can be a tedious and unnecessary effort for this work since its focus is not that. Moreover, there are readily available tools that suit for exactly this

[3] Jena2 Inference Engine, http://jena.sourceforge.net/inference/
[4] Jena Ontology API, http://jena.sourceforge.net/ontology/

purpose. Jena Ontology API is one of them.It is a free, open source application that gives the opportunity for editing and developing ontologies via OWL or Frame versions. In the thesis study, Jena Ontoloy API is used.

### 5.1.3 Servlet Container – Apache Tomcat

Nowadays, the popular trend in application development is designing web-based software. Because of the ease of client access and installation, it has also adopted the same strategy and implemented the system web-based. For that, a server that handles the requests coming via the web browsers and hosting the servlets is essential. In the thesis study, it has been chosen Apache Tomcat[5] to do that job. It is freely available.

### 5.1.4 Integrated Development Environment

The programming language choice for the development of the system in the thesis is Java. The whole system is implemented by means of Java language. Using all these tools in a single development environment simplified the effort to be dedicated. Eclipse[6] provides this easy to use integrated development environment.

## 5.2 Functionality

The functionality of our system becomes available through its interface. The interface is shown at Fig. 5.1. It is quite simple and user friendly. It consists of a text area and button. In the system, the action is initiated by entering the question in natural language form and hitting the “Search” button. At Fig. 5.1, a screenshot shows an example of a question and the returned answers.

[5] Apache Tomcat Web Server, http://tomcat.apache.org/
[6] Eclipse, http://www.eclipse.org/

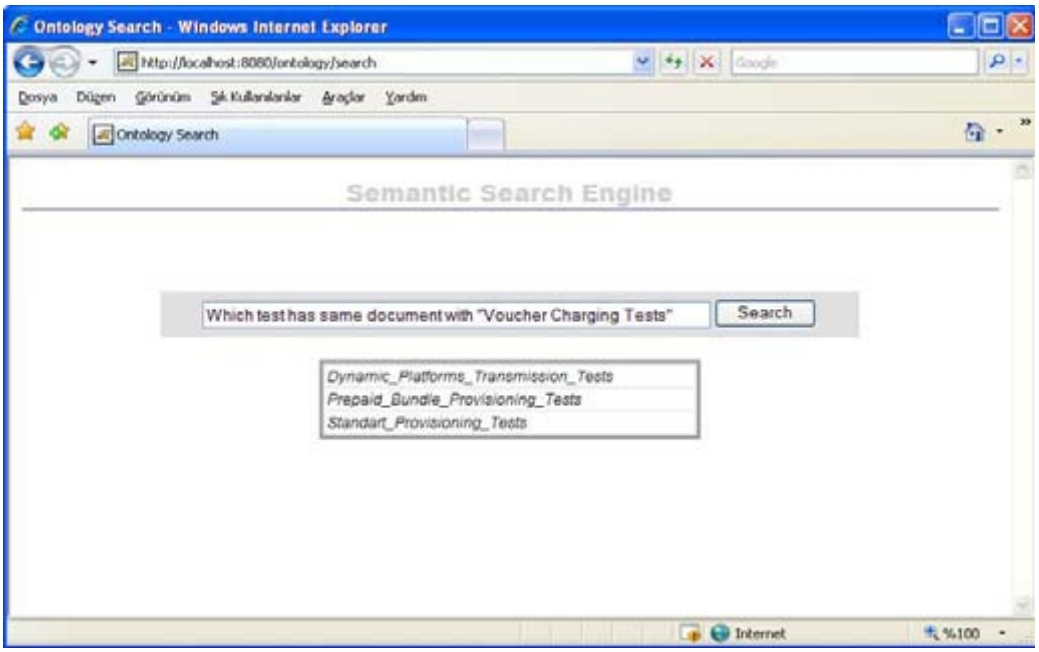


Figure 5.1: The Application's Interface.

## 5.3 Implementation

The implementation part of the system can be separated into two parts. The beginning part talks about building a document ontology which will be used as both the knowledge representation and knowledge base. The other part mentions how to interpret a user question which is entered in natural language. To be able to achieve this, it is developed an algorithm which understands the NL question and converts it into a suitable query.

### 5.3.1 Developing the document ontology

In the thesis study, OWL-DL web ontology language is used for building of the document ontology.

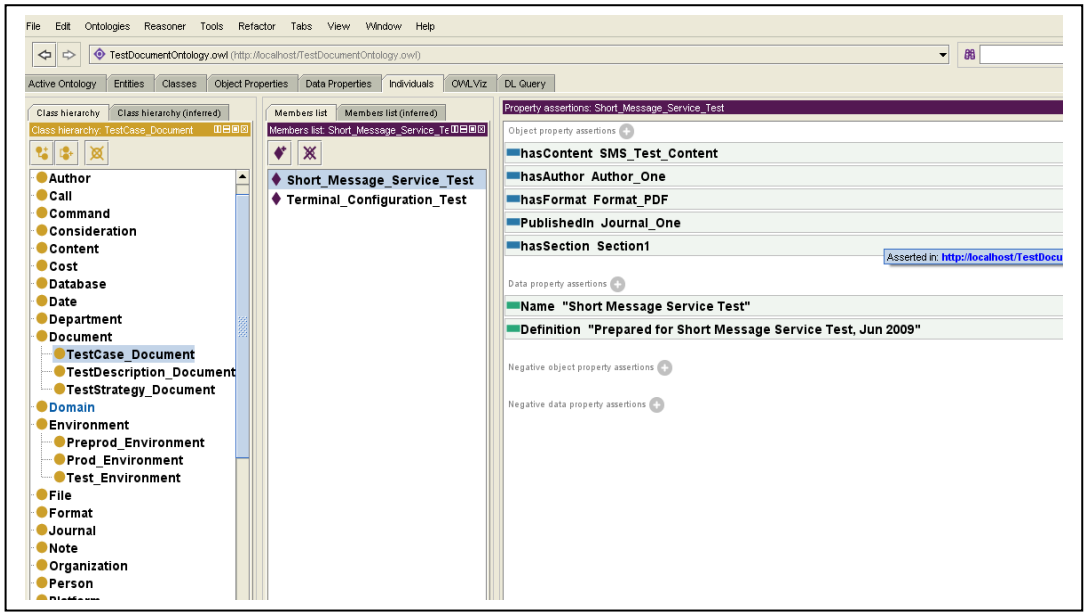


Figure 5.2: A snapshot in developing the document ontology using Protégé.

Building ontology is a large task of system engineering. As a first step of this thesis, it has been opted to construct a domain ontology following the advised methodology in [8]. The domain ontology is builded in the light of the information exist in software test documents. The test documents were related with the software test procedures performed at Vodafone. This ontology has then been used as a basis for the question answering system. As an ontology modeling and knowledge acquisition tool, Protégé [6] is used. Figure 5.2 is a snapshot in developing the document ontology using Protégé 4.1.0.

The document ontology consists of thirtyfour classes with 12 subclasses. The classes are listed below:

*author, call, command, consideration, content, cost, database, date, department, document, domain, environment, file, format, journal, note, organization, person, platform, prerequisite, procedure, processstep, project, record, section, software, subscriber, table, tariff, test, testprocess, testresult, testcase, testtool.* A part of the

software test document ontology is shown in Figure 5.3 where the ellipse boxes represent classes and the rectangular represent literals; superclasses are highlighted in bold. In addition the solid lines represent object properties and datatype properties, and the dotted lines represent subClassOf.

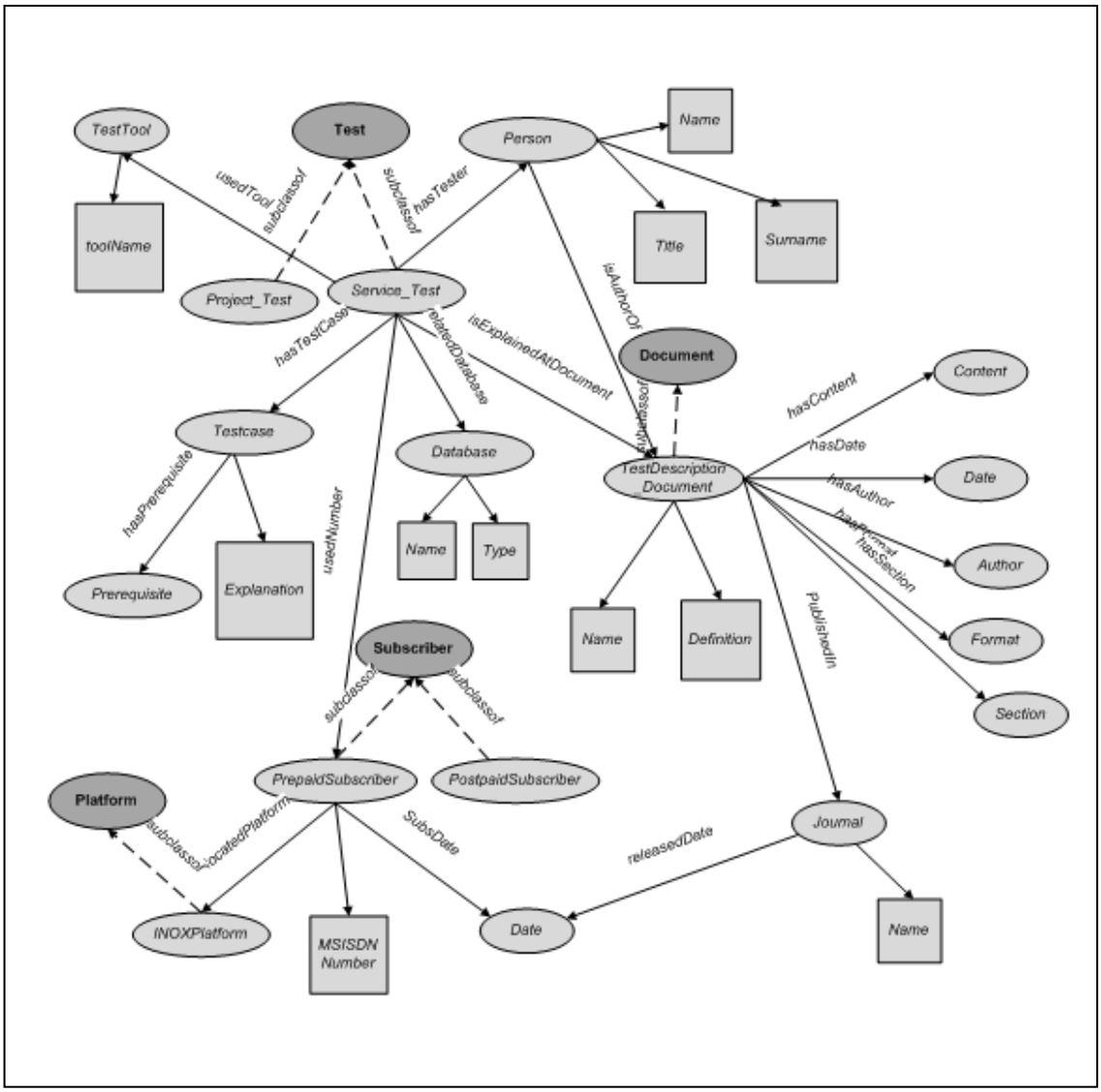


Figure 5.3: A part of the “Software Test Documents” ontology.

### 5.3.2 Interpreting a user question

In this section we describe in more detail how our application interpretes and process a question asked by the user. When the “Search” button is pressed, *doGet(HttpServletRequest request, HttpServletResponse response)* method of the SearcherServlet which appears in the aforementioned design philosophy is being called. Following the *doGet method, public ArrayList StartSearch (String question)* is called in SearcherServlet. In the *StartSearch* method, *loadOntology ()* method of the *OntoMapper class* is called. This method loads the ontology for preparing the model. By using *loadOntology* method, a small data structure keeping classes, properties and instances is created. In this way, the information belongs to the model can be used for analyzing the query sentence.

After these steps, the process of parsing sentence is started. The query sentence is formed by the object derived from *Sentence (question, true)* class. By using this object, type of the sentence is being investigated and suitable question type (i.e.: who, what, which, how many) for the question sentence is found. After that, the words of the sentence except question sentence types and auxiliary verbs ("the", "a", "an", "of", "is", "was", "has", "who", "what", "when", "which", "how", "many ", "much", "why") are being kept for parsing. While parsing the sentence, the below strategies are taken into consideration:

- Matching of the word with the rules defined at rule file is being controlled
- If the word is not a rule, then matching of the word with the classes is being controlled.
- If the class cannot be found, then alternative words are looked from WordNet.
- If class is found next word is taken into consideration, otherwise matching of the word with properties is being controlled.
- If the property cannot be found, then alternative words are looked from WordNet. If the word is still not found, then the word is accepted as a value.

- At the end of the algorithm, if the property is not found search process is again started. Except the word which was previously selected as class, each word firstly is looked for in the Class list.
- If the alternative class is found, searching process is started again. Each word is looked for in the Property list. When it's found in the Property list, it's passed to the third phase which is query building phase.

Immediately after finishing the parsing process, the query is built. Lastly, the query is sent to the reasoner by means of JENA API. After getting and processing the query by the reasoner, it is sent to the web page over the SearcherServlet. Finally, the answer is displayed to the users.

In this study a set of questions are collected from the members of the software test team at Vodafone[7]. The questions were related with the information exists in the software test documents that would be of interest to them. The main types of the questions in the system are who, what, when, which and how many.

From the collected questions, corresponding question templates were generated on the basis of the document ontology. The questions to be asked can suit to one of the eight question templates listed below:

**T1**: Which <class> has the <property> with <value>

**T2**: Which <class> has the <property>

**T3**: Who is the <property> of <value>

**T4**: How many <class> are/has <property> by <value>

**T5**: When did <value> <property>

**T6**: What is the <property> of <value> <class>

**T7**: What is the <property> of <class> which has the <value>

[7] Vodafone Telecommunication A.S.: http://www.vodafone.com.tr

**T8**: What is the <value> of <class>

When the user is entered a question and pressed the search button, the application firstly decides to choose which question template is suitable for the entered question. After choosing the suitable question template, the query generation process is started. This study has used SPARQL RDF query language to get information from the document ontology. Below a number of questions for each question template with their SPARQL query can bee seen as example:

**Q1-1:** Which persons has same surname with Acar?

```
   SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT ?Result ?Value WHERE { ?Result  rdf:type  f:Person .
?Result f:Surname  ?Value .
FILTER regex(?Value, "acar", "i" ) }
```

**Q1-2**: Which documents are written by Ahmet?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT DISTINCT ?Result ?Value WHERE {  ?z rdfs:subClassOf
f:Document .
      ?Result  rdf:type  ?z .
      ?Result f:WrittenBy  ?Value .
      ?Result ?y  ?t .?t ?v  ?b
       FILTER regex(?b, "ahmet", "i" ) }
```

**Q2-1**: Which testcases has Prerequisite?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT ?Result ?Value WHERE { ?Result  rdf:type  f:Testcase .
?Result f:hasPrerequisite  ?Value .}
```

**Q3-1:** Who has surname Aydin?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT ?Result ?Value WHERE { ?Result f:Surname  ?Value .
 FILTER regex(?Value, "aydin", "i" ) }
```

**Q4-1**: How many documents has word format?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
            SELECT ?Result WHERE {  ?z rdfs:subClassOf f:Document .
      ?Result  rdf:type  ?z .
      ?Result f:hasFormat  f:Format_Word .}
```

**Q4-2**: How many documents are written by Ahmet?

```
SPARQL query :
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT ?Result WHERE {  ?z rdfs:subClassOf f:Document .
      ?Result  rdf:type  ?z .
      ?Result f:WrittenBy  ?Value .?Value ?y ?t .
             FILTER regex(?t, "ahmet", "i" ) }
```

**Q5-1**: When was the journal Telecomm Journal released?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT DISTINCT ?Value ?c WHERE { ?Result  rdf:type
f:Journal .
 ?Result f:Released  ?Value .
 ?Result ?y  ?t .  ?Value f:hasDate  ?c .
       FILTER regex(?t, "telecomm journal", "i" ) }
```

**Q6-1**: What is the test result of 7076 sms service test ?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?Result ?Value
WHERE {  ?z rdfs:subClassOf f:Test .
?Result  rdf:type  ?z .
?Result f:hasTestResult  ?Value .
?Result ?v  ?b
FILTER regex(?b, "7076 sms service test", "i" ) }
```

**Q7-1**: What is the documents which are written by Serap?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?Result ?Value
WHERE {  ?z rdfs:subClassOf f:Document .
?Result  rdf:type  ?z .
?Result f:WrittenBy  ?Value .
?Value ?v  ?b
FILTER regex(?b, "serap", "i" ) }
```

**Q8-1**: What is the caption of Gizli Numara SMS Service Test?

```
SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?Result ?Value WHERE { ?Result f:Caption  ?Value .
       FILTER regex(?Value, "gizli numara sms servis testi", "i" )
}
```

Below examples also show the SPARQL quries of the questions where an inference is required to find the answer of the question.

**Rule Question1**: Show the qualified persons that he has at least four years of experience.

```
Rule Definition:
@prefix tst: http://localhost/TestDocumentOntology.owl#
[qualified: (?s rdf:type tst:QualifiedPerson)
           <-
          (?s rdf:type tst:Person)
          (?s tst:Experience ?c)
```

```
          greaterThan(?c,3)
]

SPARQL query:
PREFIX rdf: <http://localhost/TestDocumentOntology.owl#>
          SELECT ?Result
          WHERE {  ?Result ?y rdf:QualifiedPerson };
```

**Rule Question2**: List the persons that could be nominated to be a team leader.

```
Rule Definition:
@prefix tst: http://localhost/TestDocumentOntology.owl#
[nominate: (?s rdf:type tst:Nominated)
           <-
          (?s rdf:type tst:Person)
          (?s tst:Title "Senior")
]


SPARQL query:
PREFIX f: <http://localhost/TestDocumentOntology.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      SELECT ?Result
      WHERE { ?Result rdf:type f:Person .
  ?Result ?y f:Nominated}
```

## 5.4 Experimentation

In order to validate the proposed approach, the approach has been tested with a real-life project in software test document domain. For the evaluation it was necessary to assess whether or not the answers retrieved by the QA system are correct. As the first step of the experimentation phase, a comparison is made between the expected answer of the questions and the found answer of the questions for the dataset used in the testing session. For this aim a list of questions is determined. We collected in total 78 questions by asking 10 members of the software test team at Vodafone to generate questions for the system. In the system, users query the information and these NL queries are formalized into predicates which correspond to classses and relations in the ontology. That's why; an answer is accepted as correct with respect to a query over a specific ontology. Our QA system fails to give an answer if the knowledge is in the ontology but it cannot find it for several reasons like data model failures and linguistic failures.

Our ontology based QA system for test document domain was able to handle %69 of the total number of questions. All the results are presented in Table 5.1

**Table 5.1 – Results of QA System**

| | Amount | Percentage |
|---|---|---|
| **Number of questions answered correctly** | 54 | 69% |
| **Number of questions not answered correctly** | 0 | 0% |
| **Number of linguistic failures** | 24 | 31% |
| **Number of data model failures** | 0 | 0% |

In the evaluation of our system, data model failure accounted for 0% (0 of 78). This type of failure never occurred. All questions can be formulated as triples by our QA system. Linguistic failures accounted for 31 % (24 of 78). The reason for linguistic failure is the value variables other than class and property variables in the question sentence. Our QA system cannot build the correct query when the number of the words in the question sentence increases. For example for the question "*What is the content of Short Message Service Test document*" as shown in Table 5.2, the system finds *Test* as class and *isExplainedAtDocument* as property. So, it finds the value variable as "*content short message service*". On the other hand, class should be *Document,* property should be *hasContent* and value variable should be *Short Message Service Test document.* Since the system cannot find the class, property and value variables correctly it builds a wrong query. Consequently, the system cannot find the answer of the question.

**Table 5.2 – Example questions not correctly answered by the QA System**

| |
|---|
| **What is the content of Short Message Service Test document?** |
| **What is the author of Short Message Service Test document?** |
| **Which test has max number of test cases?** |
| **Which document has min number of test?** |
| **Who is the writer of Short Message Service Test?** |
| **Who is the author of Short Message Service Test?** |
| **What is the used tool at 7076 SMS Service Test?** |
| **Who is the tester of 7076 SMS Service Test?** |
| **What is the caption of Gizli Numara Service Test Case Document?** |
| **Who is the writer of Gizli Numara Service Test Case Document?** |

In Table 5.3, a selection of questions which are correctly answered by the system is shown. The type of the questions are who, what, when, which and how many.

**Table 5.3 – Example questions correctly answered by the QA System**

| |
|---|
| **Which persons has same surname with Acar?** |
| **Which documents are written by Ahmet?** |
| **Which testcases has Prerequisite?** |
| **Who has surname Aydin?** |
| **Who work for Telecomm Organization?** |
| **Who is the writer of SMS CDR Test document?** |
| **How many documents has word format?** |
| **How many test has type cdr?** |
| **How many documents are written by Ahmet?** |
| **When was the journal Telecomm Journal released?** |
| **When was Ali born?** |
| **What is the test result of 7076 SMS Service Test?** |
| **What are the documents written by Serap?** |
| **What is the caption of Gizli Numara SMS Service Test ?** |
| **What is the result of testcase1?** |
| **What is the testcases of Interactive Voice Response Service Test?** |

Our evaluation with ten users shows that the sytem is extremely simple to use without any training resulting in adequate query performance. Our system can answer %69 of the total number of questions correctly. As a future work, improvements on the interpreting NL question part of the algorithm can be performed in such a way that although the question sentence contains lots of words, the sytem can be cabaple of finding class, property and value variables correctly. So, it can be capable of building correct query to find the answer of the question correctly.

# CHAPTER 6

# CONCLUSION

## 6.1 Summary

This thesis presents an automated QA system in which the software test documents domain knowledge is represented by means of an ontology. The system allows users to enter a question about the domain in NL and returns exact answer of the questions. Conversion of the natural language question into the ontology based query is the challenging part of the system. To be able to achieve this, a new algorithm is proposed. The algorithm is based on investigation of suitable question type and parsing the words of the question sentence.

At the beginning, the background information has been given about Description Logics, Ontology, OWL Ontology Language, A Knowledge Engineering Methodology for Ontology Development, Ontology Reasoning, Ontology-based QA Systems and WordNet. Then, the system architecture was explained in detail. Following the system architecture, tools and implementation details were described one by one. Finally, after the implementation details, experimentation section has taken place.

## 6.2 Conclusions

The aim of this work is to describe an algorithm regarding conversion of natural language question into an ontology based search engine query for an automated question answering system which is mainly based on software test document domain. This algorithm is appropriate for ontology-based question answering in

restricted domains because the query templates which in turn can be used to retrieve answers, are able to be generated on the basis of a domain ontology.

In question answering, it is particularly important to achieve a high accuracy in parsing the questions. So, experimentation phase has been done for the evaluation of the QA system. Main goal of this thesis was getting successful results with correct answers to the questions. Evaluation phase ended up with successful results, which validates the idea of this approach with enough success rates.

## 6.3 Future work

Although the results obtained are accurate, the work presented in this study can be extended in several directions. For the evaluation part of the system, the number of the questions to be asked can be increased in such a way that same questions can be asked with different ways or different types of new questions can be added into the dataset. So the performance of the system can be evaluated more truly. In addition; although the algorithm interpretes and understands the input queries intelligently, it can be gained more intelligence to the algorithm for answering much complicated questions.

Another extension of the study might be enhancing performance issues. For each new question, the reasoner is loading the ontology into the memory. This must be avoided at all costs. There might be some improvements to the term expansion and query relaxation strategies for getting more precise answers. Moreover, much trickier questions that require complex automated reasoning processes can also be handled by the proposed ontology based question answering system.

## REFERENCES


[1] Lopez V., Pasin M., Motta E., "AquaLog: An Ontology-portable Question Answering System for the Semantic Web", In Proceedings European Semantic Web Conference, Crete, 2005.

[2] Linckels S., Meinel C., "A Simple Application of Description Logics for a Semantic Search Engine", In Proceedings of the IADIS International Conference of Applied Computing (IADIS AC2005), Vol. II, pp. 306-311, Lisbon, Portugal, 2005.

[3] Baader F., Calvanese D., McGuinness D., Nardi D., Schneider P., "The Description Logic Handbook", Cambridge University Press, 2003.

[4] Kaufmann E., Bernstein A., Zumstein R., "Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs", The Semantic Web – ISWC 2006, Athens, GA, 2006, pp. 980-981.

[5] Miller, George A., "WordNet: A lexical database for English", Communications of the ACM, 38(11):39-41, 1995.

[6] Gennari J., Musen M., Fergerson R., Grosso W., Crubézy M., Eriksson H., Noy N., Tu S., "The evolution of Protégé: An environment for knowledge-based systems development", International Journal of Human-Computer Studies, 1, 89–123, 2003.

[7] Pan J. Z., Thomas E., Sleeman D., "ONTOSEARCH2: Searching and Querying Web Ontologies", In Proceedings IADIS International Conference WWW/Internet 2006, pages pp. 211-219.

[8] Noy N.F., McGuinness D.L., "Ontology Development101: A Guide to Creating Your First Ontology", 2001.

[9] Guo Q., Zhang M., "Question Answering System Based on Ontology and Semantic Web", RSKT 2008, LNAI 5009, pp. 652–659, 2008.

[10] Lee SM., Ryu P., Choi KS. "Ontology-based Question Answering System".

[11] Hu B., Dasmahapatra S., Lewis P., Shadbolt N., "Ontology-based Medical Image Annotation with Description Logics", ICTAI'03, 2003.

[12] Bernstein A., Kaufmann E., Kaiser C., "Querying the Semantic Web with Ginseng: A Guided Input Natural Language Search Engine", In Proceedings of

the 15th Workshop on Information Technology and Systems (WITS 2005), Las Vegas, NV, 2005, pp. 45-50.

# APPENDIX A

# DOMAIN ONTOLOGY

In this section, document ontology used in the question answering system is given.

## A.1 OWL-DL Document Ontology

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY TestDocumentOntology
"http://localhost/TestDocumentOntology.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY TestDocumentOntology2
"http://localhost/TestDocumentOntology.owl#7075" >
    <!ENTITY TestDocumentOntology4
"http://localhost/TestDocumentOntology.owl#7076" >
    <!ENTITY TestDocumentOntology5
"http://localhost/TestDocumentOntology.owl#5427773432" >
    <!ENTITY TestDocumentOntology3
"http://localhost/TestDocumentOntology.owl#5423129893" >
]>


<rdf:RDF xmlns="http://localhost/TestDocumentOntology.owl#"
     xml:base="http://localhost/TestDocumentOntology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:TestDocumentOntology3="&TestDocumentOntology;5423129893"
     xmlns:TestDocumentOntology2="&TestDocumentOntology;7075"

xmlns:TestDocumentOntology="http://localhost/TestDocumentOntology.o
wl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:TestDocumentOntology5="&TestDocumentOntology;5427773432"
     xmlns:TestDocumentOntology4="&TestDocumentOntology;7076">
    <owl:Ontology
rdf:about="http://localhost/TestDocumentOntology.owl"/>
```

```
    <!--

///////////////////////////////////////////////////////////////////////
//////////////////
    //
    // Datatypes
    //

///////////////////////////////////////////////////////////////////////
//////////////////
     -->



    <!-- http://www.w3.org/2001/XMLSchema#date -->

    <rdfs:Datatype rdf:about="&xsd;date"/>



    <!--

///////////////////////////////////////////////////////////////////////
//////////////////
    //
    // Object Properties
    //

///////////////////////////////////////////////////////////////////////
//////////////////
     -->



    <!-- http://localhost/TestDocumentOntology.owl#Age -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Age">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Caption -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Caption">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Definition -->
```

```
    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;Definition">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#MSISDN_Number --
>

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;MSISDN_Number">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Name -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Name">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#PublishedIn -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;PublishedIn">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Released -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Released">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Result -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Result">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#Subs_Postal_Address -->
```

```
    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;Subs_Postal_Address">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Surname -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Surname">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#Title -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;Title">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#WrittenBy -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;WrittenBy">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#associatedProject -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;associatedProject">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#callPrice -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;callPrice">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#callType -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;callType">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
```

```
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#commandDescription -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;commandDescription">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#commandName -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;commandName">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#costPrice -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;costPrice">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#databaseName -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;databaseName">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#databaseType -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;databaseType">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#departmentName -
->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;departmentName">
```

```
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasAuthor -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasAuthor">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasContent -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasContent">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasCost -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;hasCost">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasDate -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;hasDate">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasFormat -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasFormat">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasPrerequisite
-->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasPrerequisite">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#hasSection -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasSection">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasTable -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;hasTable">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasTestCase -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasTestCase">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasTestResult --
>

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasTestResult">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasTester -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasTester">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#hasWorkers -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;hasWorkers">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#isAuthor -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;isAuthor">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#isAuthorOf -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;isAuthorOf">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#isExplainedAtDocument -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;isExplainedAtDocument">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#isPerson -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;isPerson">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#locatedPlatform
-->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;locatedPlatform">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#prerequisiteContent -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;prerequisiteContent">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#projectName -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;projectName">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#subsDate -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;subsDate">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#tableName -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;tableName">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#technicalCoordinator -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;technicalCoordinator">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!--
http://localhost/TestDocumentOntology.owl#testcaseDescription -->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;testcaseDescription">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#toolName -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;toolName">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:ObjectProperty>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#usedTestNumber -
->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;usedTestNumber">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#usedTool -->

    <owl:ObjectProperty rdf:about="&TestDocumentOntology;usedTool">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://localhost/TestDocumentOntology.owl#workedDepartment
-->

    <owl:ObjectProperty
rdf:about="&TestDocumentOntology;workedDepartment">
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </owl:ObjectProperty>



    <!-- http://www.w3.org/2002/07/owl#topDataProperty -->

    <owl:ObjectProperty rdf:about="&owl;topDataProperty"/>



    <!-- http://www.w3.org/2002/07/owl#topObjectProperty -->

    <owl:ObjectProperty rdf:about="&owl;topObjectProperty"/>



    <!--

///////////////////////////////////////////////////////////////////
//////////////////
    //
    // Data properties
    //

///////////////////////////////////////////////////////////////////
//////////////////
     -->
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Age -->

    <owl:DatatypeProperty rdf:about="&TestDocumentOntology;Age"/>


    <!-- http://localhost/TestDocumentOntology.owl#Caption -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;Caption"/>


    <!-- http://localhost/TestDocumentOntology.owl#Definition -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;Definition"/>


    <!-- http://localhost/TestDocumentOntology.owl#MSISDN_Number --
>

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;MSISDN_Number"/>


    <!-- http://localhost/TestDocumentOntology.owl#Name -->

    <owl:DatatypeProperty rdf:about="&TestDocumentOntology;Name"/>


    <!-- http://localhost/TestDocumentOntology.owl#Result -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;Result"/>


    <!--
http://localhost/TestDocumentOntology.owl#Subs_Postal_Address -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;Subs_Postal_Address"/>


    <!-- http://localhost/TestDocumentOntology.owl#Surname -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;Surname"/>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Title -->

    <owl:DatatypeProperty rdf:about="&TestDocumentOntology;Title"/>


    <!-- http://localhost/TestDocumentOntology.owl#callDuration -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;callDuration">
        <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
    </owl:DatatypeProperty>


    <!-- http://localhost/TestDocumentOntology.owl#callPrice -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;callPrice"/>


    <!-- http://localhost/TestDocumentOntology.owl#callType -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;callType"/>


    <!--
http://localhost/TestDocumentOntology.owl#commandDescription -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;commandDescription"/>


    <!-- http://localhost/TestDocumentOntology.owl#commandName -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;commandName"/>


    <!-- http://localhost/TestDocumentOntology.owl#costPrice -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;costPrice"/>


    <!-- http://localhost/TestDocumentOntology.owl#databaseName -->
```

```
    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;databaseName"/>


    <!-- http://localhost/TestDocumentOntology.owl#databaseType -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;databaseType"/>


    <!-- http://localhost/TestDocumentOntology.owl#departmentName -
->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;departmentName"/>


    <!-- http://localhost/TestDocumentOntology.owl#hasDate -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;hasDate"/>


    <!--
http://localhost/TestDocumentOntology.owl#prerequisiteContent -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;prerequisiteContent"/>


    <!-- http://localhost/TestDocumentOntology.owl#projectName -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;projectName"/>


    <!-- http://localhost/TestDocumentOntology.owl#tableName -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;tableName"/>


    <!--
http://localhost/TestDocumentOntology.owl#testcaseDescription -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;testcaseDescription"/>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#toolName -->

    <owl:DatatypeProperty
rdf:about="&TestDocumentOntology;toolName"/>



    <!-- http://www.w3.org/2002/07/owl#topDataProperty -->

    <owl:DatatypeProperty rdf:about="&owl;topDataProperty"/>



    <!--

///////////////////////////////////////////////////////////////
//////////////////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////
//////////////////
     -->



    <!-- http://localhost/TestDocumentOntology.owl#Author -->

    <owl:Class rdf:about="&TestDocumentOntology;Author"/>



    <!-- http://localhost/TestDocumentOntology.owl#Call -->

    <owl:Class rdf:about="&TestDocumentOntology;Call"/>



    <!-- http://localhost/TestDocumentOntology.owl#Command -->

    <owl:Class rdf:about="&TestDocumentOntology;Command"/>



    <!-- http://localhost/TestDocumentOntology.owl#Consideration --
>

    <owl:Class rdf:about="&TestDocumentOntology;Consideration"/>
```

```
<!-- http://localhost/TestDocumentOntology.owl#Content -->
<owl:Class rdf:about="&TestDocumentOntology;Content"/>

<!-- http://localhost/TestDocumentOntology.owl#Cost -->
<owl:Class rdf:about="&TestDocumentOntology;Cost"/>

<!-- http://localhost/TestDocumentOntology.owl#Database -->
<owl:Class rdf:about="&TestDocumentOntology;Database"/>

<!-- http://localhost/TestDocumentOntology.owl#Date -->
<owl:Class rdf:about="&TestDocumentOntology;Date"/>

<!-- http://localhost/TestDocumentOntology.owl#Department -->
<owl:Class rdf:about="&TestDocumentOntology;Department"/>

<!-- http://localhost/TestDocumentOntology.owl#Document -->
<owl:Class rdf:about="&TestDocumentOntology;Document"/>

<!-- http://localhost/TestDocumentOntology.owl#Domain -->
<owl:Class rdf:about="&TestDocumentOntology;Domain"/>

<!-- http://localhost/TestDocumentOntology.owl#Environment -->
<owl:Class rdf:about="&TestDocumentOntology;Environment"/>

<!-- http://localhost/TestDocumentOntology.owl#File -->
<owl:Class rdf:about="&TestDocumentOntology;File"/>

<!-- http://localhost/TestDocumentOntology.owl#Format -->
```

```
    <owl:Class rdf:about="&TestDocumentOntology;Format"/>


    <!-- http://localhost/TestDocumentOntology.owl#IN_Platform -->

    <owl:Class rdf:about="&TestDocumentOntology;IN_Platform">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Platform"/>
    </owl:Class>


    <!-- http://localhost/TestDocumentOntology.owl#Journal -->

    <owl:Class rdf:about="&TestDocumentOntology;Journal">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>


    <!-- http://localhost/TestDocumentOntology.owl#Note -->

    <owl:Class rdf:about="&TestDocumentOntology;Note">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>


    <!-- http://localhost/TestDocumentOntology.owl#Organization -->

    <owl:Class rdf:about="&TestDocumentOntology;Organization"/>


    <!-- http://localhost/TestDocumentOntology.owl#Person -->

    <owl:Class rdf:about="&TestDocumentOntology;Person"/>


    <!-- http://localhost/TestDocumentOntology.owl#Platform -->

    <owl:Class rdf:about="&TestDocumentOntology;Platform"/>


    <!--
http://localhost/TestDocumentOntology.owl#PostpaidSubscriber -->

    <owl:Class
rdf:about="&TestDocumentOntology;PostpaidSubscriber">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Subscriber"/>
```

```
    </owl:Class>


    <!--
http://localhost/TestDocumentOntology.owl#PrepaidSubscriber -->

    <owl:Class rdf:about="&TestDocumentOntology;PrepaidSubscriber">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Subscriber"/>
    </owl:Class>



    <!--
http://localhost/TestDocumentOntology.owl#Preprod_Environment -->

    <owl:Class
rdf:about="&TestDocumentOntology;Preprod_Environment">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Environment"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Prerequisite -->

    <owl:Class rdf:about="&TestDocumentOntology;Prerequisite"/>



    <!-- http://localhost/TestDocumentOntology.owl#Procedure -->

    <owl:Class rdf:about="&TestDocumentOntology;Procedure"/>



    <!-- http://localhost/TestDocumentOntology.owl#ProcessStep -->

    <owl:Class rdf:about="&TestDocumentOntology;ProcessStep"/>



    <!-- http://localhost/TestDocumentOntology.owl#Prod_Environment
-->

    <owl:Class rdf:about="&TestDocumentOntology;Prod_Environment">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Environment"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Project -->
```

```
    <owl:Class rdf:about="&TestDocumentOntology;Project">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Project_Test -->

    <owl:Class rdf:about="&TestDocumentOntology;Project_Test">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Test"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Record -->

    <owl:Class rdf:about="&TestDocumentOntology;Record"/>



    <!-- http://localhost/TestDocumentOntology.owl#Section -->

    <owl:Class rdf:about="&TestDocumentOntology;Section"/>



    <!-- http://localhost/TestDocumentOntology.owl#Service_Test -->

    <owl:Class rdf:about="&TestDocumentOntology;Service_Test">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Test"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Software -->

    <owl:Class rdf:about="&TestDocumentOntology;Software"/>



    <!-- http://localhost/TestDocumentOntology.owl#Subscriber -->

    <owl:Class rdf:about="&TestDocumentOntology;Subscriber"/>



    <!-- http://localhost/TestDocumentOntology.owl#Table -->

    <owl:Class rdf:about="&TestDocumentOntology;Table"/>



    <!-- http://localhost/TestDocumentOntology.owl#Tariff -->
```

```
    <owl:Class rdf:about="&TestDocumentOntology;Tariff"/>


    <!-- http://localhost/TestDocumentOntology.owl#Test -->

    <owl:Class rdf:about="&TestDocumentOntology;Test"/>


    <!--
http://localhost/TestDocumentOntology.owl#TestCase_Document -->

    <owl:Class rdf:about="&TestDocumentOntology;TestCase_Document">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Document"/>
    </owl:Class>


    <!--
http://localhost/TestDocumentOntology.owl#TestDescription_Document
-->

    <owl:Class
rdf:about="&TestDocumentOntology;TestDescription_Document">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Document"/>
    </owl:Class>


    <!-- http://localhost/TestDocumentOntology.owl#TestIN_Platform
-->

    <owl:Class rdf:about="&TestDocumentOntology;TestIN_Platform">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Platform"/>
    </owl:Class>


    <!-- http://localhost/TestDocumentOntology.owl#TestProcess -->

    <owl:Class rdf:about="&TestDocumentOntology;TestProcess"/>


    <!-- http://localhost/TestDocumentOntology.owl#TestResult -->

    <owl:Class rdf:about="&TestDocumentOntology;TestResult"/>
```

```
    <!--
http://localhost/TestDocumentOntology.owl#TestStrategy_Document -->

    <owl:Class
rdf:about="&TestDocumentOntology;TestStrategy_Document">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Document"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Test_Environment
-->

    <owl:Class rdf:about="&TestDocumentOntology;Test_Environment">
        <rdfs:subClassOf
rdf:resource="&TestDocumentOntology;Environment"/>
    </owl:Class>



    <!-- http://localhost/TestDocumentOntology.owl#Testcase -->

    <owl:Class rdf:about="&TestDocumentOntology;Testcase"/>



    <!-- http://localhost/TestDocumentOntology.owl#Testtool -->

    <owl:Class rdf:about="&TestDocumentOntology;Testtool">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>



    <!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing"/>



    <!--

//////////////////////////////////////////////////////////////////
//////////////////
    //
    // Individuals
    //

//////////////////////////////////////////////////////////////////
//////////////////
     -->
```

```
    <!--
http://localhost/TestDocumentOntology.owl#7075_SMS_TestCase -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;7075_SMS_TestCase">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestCase_Document"/>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#7075_SMS_Test_Strategy --
>

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;7075_SMS_Test_Strategy">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestStrategy_Document"/>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#7076_SMS_Service_Test -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;7076_SMS_Service_Test">
        <rdf:type
rdf:resource="&TestDocumentOntology;Service_Test"/>
        <Name rdf:datatype="&xsd;string">7076 SMS Service
Test</Name>
        <usedTool rdf:resource="&TestDocumentOntology;Jira"/>
        <usedTestNumber
rdf:resource="&TestDocumentOntology;PreSubscriber1"/>
        <isExplainedAtDocument
rdf:resource="&TestDocumentOntology;SMS_Service_TestDescription"/>
        <hasTestResult
rdf:resource="&TestDocumentOntology;TestResult1"/>
        <hasTestCase
rdf:resource="&TestDocumentOntology;Testcase1"/>
        <hasTestCase
rdf:resource="&TestDocumentOntology;Testcase2"/>
        <hasTestCase
rdf:resource="&TestDocumentOntology;Testcase3"/>
        <hasTester
rdf:resource="&TestDocumentOntology;Tester_One"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#ABONE -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;ABONE">
```

```
        <rdf:type rdf:resource="&TestDocumentOntology;Table"/>
        <tableName rdf:datatype="&xsd;string">Abone</tableName>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Author_One -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Author_One">
        <rdf:type rdf:resource="&TestDocumentOntology;Author"/>
        <Name rdf:datatype="&xsd;string">Ahmet</Name>
        <Surname rdf:datatype="&xsd;string">Kaymaz</Surname>
        <isAuthorOf
rdf:resource="&TestDocumentOntology;7075_SMS_Test_Strategy"/>
        <isAuthorOf
rdf:resource="&TestDocumentOntology;Short_Message_Service_Test"/>
        <isPerson rdf:resource="&TestDocumentOntology;Tester_One"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Author_Two -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Author_Two">
        <rdf:type rdf:resource="&TestDocumentOntology;Author"/>
        <isAuthorOf
rdf:resource="&TestDocumentOntology;Terminal_Configuration_Test"/>
        <isPerson rdf:resource="&TestDocumentOntology;Tester_Two"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Babycam_Project
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Babycam_Project">
        <rdf:type rdf:resource="&TestDocumentOntology;Project"/>
        <projectName
rdf:datatype="&xsd;string">Babycam</projectName>
        <technicalCoordinator
rdf:resource="&TestDocumentOntology;Technical_Coordinator1"/>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Babycam_Provisioning_Test
s -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Babycam_Provisioning_Tests">
```

```
        <rdf:type
rdf:resource="&TestDocumentOntology;Project_Test"/>
        <associatedProject
rdf:resource="&TestDocumentOntology;Babycam_Project"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Call1 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;Call1">
        <rdf:type rdf:resource="&TestDocumentOntology;Call"/>
        <callDuration rdf:datatype="&xsd;string">5
minutes</callDuration>
        <callType rdf:datatype="&xsd;string">Test</callType>
        <hasCost rdf:resource="&TestDocumentOntology;Cost1"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Cost1 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;Cost1">
        <rdf:type rdf:resource="&TestDocumentOntology;Cost"/>
        <costPrice rdf:datatype="&xsd;string">Three TL</costPrice>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#DB_Security_Dept
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;DB_Security_Dept">
        <rdf:type rdf:resource="&TestDocumentOntology;Department"/>
        <departmentName rdf:datatype="&xsd;string">DB
Security</departmentName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#DVLMEDDB -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;DVLMEDDB">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseType
rdf:datatype="&xsd;string">Canl&#305;</databaseType>
        <databaseName
rdf:datatype="&xsd;string">DVLMEDDB</databaseName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#DVLRTC -->
```

```
    <owl:NamedIndividual rdf:about="&TestDocumentOntology;DVLRTC">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseName
rdf:datatype="&xsd;string">DVLRTC</databaseName>
        <databaseType
rdf:datatype="&xsd;string">Development</databaseType>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#DVLSAS -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;DVLSAS">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseName
rdf:datatype="&xsd;string">DVLSAS</databaseName>
        <databaseType
rdf:datatype="&xsd;string">Development</databaseType>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Date1 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;Date1">
        <rdf:type rdf:resource="&TestDocumentOntology;Date"/>
        <hasDate rdf:datatype="&xsd;date">03.05.2000</hasDate>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Development_Dept
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Development_Dept">
        <rdf:type rdf:resource="&TestDocumentOntology;Department"/>
        <departmentName
rdf:datatype="&xsd;string">Development</departmentName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Format_Exel -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Format_Exel">
        <rdf:type rdf:resource="&TestDocumentOntology;Format"/>
        <Name rdf:datatype="&xsd;string">Exel</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Format_PDF -->
```

```
    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Format_PDF">
        <rdf:type rdf:resource="&TestDocumentOntology;Format"/>
        <Name rdf:datatype="&xsd;string">PDF</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Format_Word -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Format_Word">
        <rdf:type rdf:resource="&TestDocumentOntology;Format"/>
        <Name rdf:datatype="&xsd;string">Word</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#GRIPIN_Project -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;GRIPIN_Project">
        <rdf:type rdf:resource="&TestDocumentOntology;Project"/>
        <projectName rdf:datatype="&xsd;string">Gripin New SMS
Addons</projectName>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Gizli_Numara_SMS_Servis_T
esti -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Gizli_Numara_SMS_Servis_Testi">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestCase_Document"/>
        <Caption rdf:datatype="&xsd;string">Gizli Numara Servis
Test Case Dokumani </Caption>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Gripin_Provisioning_Tests
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Gripin_Provisioning_Tests">
        <rdf:type
rdf:resource="&TestDocumentOntology;Project_Test"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#INOX1 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX1">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX10 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX10">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX11 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX11">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX12 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX12">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX2 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX2">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX3 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX3">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#INOX4 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX4">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX5 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX5">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX6 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX6">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX7 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX7">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX8 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX8">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#INOX9 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;INOX9">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestIN_Platform"/>
    </owl:NamedIndividual>
```

```
    <!--
http://localhost/TestDocumentOntology.owl#INOX_Upgrade_Test_Descrip
tion -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;INOX_Upgrade_Test_Description">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestDescription_Document"/>
        <hasSection rdf:resource="&TestDocumentOntology;Section1"/>
        <hasSection rdf:resource="&TestDocumentOntology;Section2"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#IT_Security_Dept
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;IT_Security_Dept">
        <rdf:type rdf:resource="&TestDocumentOntology;Department"/>
        <departmentName rdf:datatype="&xsd;string">IT
Security</departmentName>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#IVR_Service_TestDescripti
on -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;IVR_Service_TestDescription">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestDescription_Document"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Jira -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;Jira">
        <rdf:type rdf:resource="&TestDocumentOntology;Testtool"/>
        <toolName rdf:datatype="&xsd;string">Jira</toolName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Journal_One -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Journal_One">
        <rdf:type rdf:resource="&TestDocumentOntology;Journal"/>
        <Name rdf:datatype="&xsd;string">Telecomm Journal</Name>
        <Released rdf:resource="&TestDocumentOntology;Date1"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Mercury -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;Mercury">
        <rdf:type rdf:resource="&TestDocumentOntology;Testtool"/>
        <toolName rdf:datatype="&xsd;string">Mercury</toolName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Organization_One
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Organization_One">
        <rdf:type
rdf:resource="&TestDocumentOntology;Organization"/>
        <hasWorkers
rdf:resource="&TestDocumentOntology;Tester_One"/>
        <hasWorkers
rdf:resource="&TestDocumentOntology;Tester_Two"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#PostSubscriber1
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;PostSubscriber1">
        <rdf:type
rdf:resource="&TestDocumentOntology;PostpaidSubscriber"/>
        <MSISDN_Number
rdf:datatype="&xsd;string">5434965787</MSISDN_Number>
        <Subs_Postal_Address rdf:datatype="&xsd;string">Mutlu Mah.
Nam&#305;k Kemal Cad. No:23 &#350;i&#351;li
Istanbul</Subs_Postal_Address>
        <subsDate rdf:resource="&TestDocumentOntology;Date1"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#PreSubscriber1 -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;PreSubscriber1">
        <rdf:type
rdf:resource="&TestDocumentOntology;PrepaidSubscriber"/>
        <locatedPlatform
rdf:resource="&TestDocumentOntology;INOX4"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Prerequisite1 --
>

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Prerequisite1">
        <rdf:type
rdf:resource="&TestDocumentOntology;Prerequisite"/>
        <prerequisiteContent rdf:datatype="&xsd;string">Abone
testIN de tan&#305;mlanmal&#305;d&#305;r</prerequisiteContent>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#SMS_Service_TestDescripti
on -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;SMS_Service_TestDescription">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestDescription_Document"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#SMS_Test_Content
-->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;SMS_Test_Content">
        <rdf:type rdf:resource="&TestDocumentOntology;Content"/>
        <Name
rdf:resource="&TestDocumentOntology;SMS_Test_Content"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Section1 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Section1">
        <rdf:type rdf:resource="&TestDocumentOntology;Section"/>
        <Name rdf:datatype="&xsd;string">Voucher Charging
Tests</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Section2 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Section2">
        <rdf:type rdf:resource="&TestDocumentOntology;Section"/>
```

```
        <Name rdf:datatype="&xsd;string">IN to IN ta&#351;&#305;ma
tests</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#SecureCRT -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;SecureCRT">
        <rdf:type rdf:resource="&TestDocumentOntology;Testtool"/>
        <toolName rdf:datatype="&xsd;string">SecureCRT</toolName>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Service_Management_Dept -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Service_Management_Dept">
        <rdf:type rdf:resource="&TestDocumentOntology;Department"/>
        <departmentName rdf:datatype="&xsd;string">Service
Management</departmentName>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Short_Message_Service_Tes
t -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Short_Message_Service_Test">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestCase_Document"/>
        <Definition rdf:datatype="&xsd;string">Prepared for Short
Message Service Test, Jun 2009</Definition>
        <Name rdf:datatype="&xsd;string">Short Message Service
Test</Name>
        <hasAuthor
rdf:resource="&TestDocumentOntology;Author_One"/>
        <WrittenBy
rdf:resource="&TestDocumentOntology;Author_One"/>
        <hasFormat
rdf:resource="&TestDocumentOntology;Format_Word"/>
        <PublishedIn
rdf:resource="&TestDocumentOntology;Journal_One"/>
        <hasContent
rdf:resource="&TestDocumentOntology;SMS_Test_Content"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#THQR01 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;THQR01">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseType
rdf:datatype="&xsd;string">Canl&#305;</databaseType>
        <databaseName
rdf:datatype="&xsd;string">THQR01</databaseName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#THQR02 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;THQR02">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseType
rdf:datatype="&xsd;string">Canl&#305;</databaseType>
        <databaseName
rdf:datatype="&xsd;string">THQR02</databaseName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#THQR03 -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;THQR03">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseType
rdf:datatype="&xsd;string">Canl&#305;</databaseType>
        <databaseName
rdf:datatype="&xsd;string">THQR03</databaseName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#TOAD -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;TOAD">
        <rdf:type rdf:resource="&TestDocumentOntology;Testtool"/>
        <toolName rdf:datatype="&xsd;string">TOAD</toolName>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#TSTMEDDB -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;TSTMEDDB">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseName
rdf:datatype="&xsd;string">TSTMEDDB</databaseName>
        <databaseType
rdf:datatype="&xsd;string">Test</databaseType>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#TSTRTC -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;TSTRTC">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseName
rdf:datatype="&xsd;string">TSTRTC</databaseName>
        <databaseType
rdf:datatype="&xsd;string">Test</databaseType>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#TSTSAS -->

    <owl:NamedIndividual rdf:about="&TestDocumentOntology;TSTSAS">
        <rdf:type rdf:resource="&TestDocumentOntology;Database"/>
        <databaseName
rdf:datatype="&xsd;string">TSTSAS</databaseName>
        <databaseType
rdf:datatype="&xsd;string">Test</databaseType>
        <hasTable rdf:resource="&TestDocumentOntology;ABONE"/>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Technical_Coordinator1 --
>

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Technical_Coordinator1">
        <rdf:type rdf:resource="&TestDocumentOntology;Person"/>
        <Age rdf:datatype="&xsd;int">31</Age>
        <Name rdf:datatype="&xsd;string">Hatice</Name>
        <Surname rdf:datatype="&xsd;string">Ozsoy</Surname>
        <Title rdf:datatype="&xsd;string">Technical
Coordinator</Title>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Terminal_Configuration_Te
st -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Terminal_Configuration_Test">
        <rdf:type
rdf:resource="&TestDocumentOntology;TestCase_Document"/>
        <WrittenBy
rdf:resource="&TestDocumentOntology;Author_One"/>
```

```
        <hasAuthor
rdf:resource="&TestDocumentOntology;Author_One"/>
        <hasFormat
rdf:resource="&TestDocumentOntology;Format_Word"/>
        <Name
rdf:resource="&TestDocumentOntology;Terminal_Configuration_Test"/>
        <hasContent
rdf:resource="&TestDocumentOntology;Terminal_Test_Content"/>
    </owl:NamedIndividual>



    <!--
http://localhost/TestDocumentOntology.owl#Terminal_Test_Content -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Terminal_Test_Content">
        <rdf:type rdf:resource="&TestDocumentOntology;Content"/>
        <Name
rdf:resource="&TestDocumentOntology;Terminal_Test_Content"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#TestResult1 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;TestResult1">
        <rdf:type rdf:resource="&TestDocumentOntology;TestResult"/>
        <Result rdf:datatype="&xsd;string">Successful</Result>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#TestResult2 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;TestResult2">
        <rdf:type rdf:resource="&TestDocumentOntology;TestResult"/>
        <Result rdf:datatype="&xsd;string">Unsuccessful</Result>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_Dept -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_Dept">
        <rdf:type rdf:resource="&TestDocumentOntology;Department"/>
        <departmentName
rdf:datatype="&xsd;string">Test</departmentName>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX1 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX1">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX10 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX10">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX11 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX11">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX12 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX12">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX2 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX2">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX3 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX3">
```

```
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX4 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX4">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX5 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX5">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX6 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX6">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX7 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX7">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>


    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX8 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX8">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#Test_INOX9 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Test_INOX9">
        <rdf:type
rdf:resource="&TestDocumentOntology;IN_Platform"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Testcase1 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Testcase1">
        <rdf:type rdf:resource="&TestDocumentOntology;Testcase"/>
        <testcaseDescription rdf:datatype="&xsd;string">INOX1
platformundaki abone 7075 servisine sms
g&#246;nderir</testcaseDescription>
        <hasPrerequisite
rdf:resource="&TestDocumentOntology;Prerequisite1"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Testcase2 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Testcase2">
        <rdf:type rdf:resource="&TestDocumentOntology;Testcase"/>
        <testcaseDescription rdf:datatype="&xsd;string">INOX2
platformundaki abone 7075 servisine SMS
g&#246;nderir</testcaseDescription>
        <Name rdf:datatype="&xsd;string">sms Testcase1</Name>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Testcase3 -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Testcase3">
        <rdf:type rdf:resource="&TestDocumentOntology;Testcase"/>
        <testcaseDescription rdf:datatype="&xsd;string">INOX3
platformundaki abone 7075 servisine sms
g&#246;nderir.</testcaseDescription>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Tester_One -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Tester_One">
        <rdf:type rdf:resource="&TestDocumentOntology;Person"/>
```

```
        <Age rdf:datatype="&xsd;int">25</Age>
        <Name rdf:datatype="&xsd;string">Ali</Name>
        <Surname rdf:datatype="&xsd;string">Atar</Surname>
        <Title rdf:datatype="&xsd;string">Senior</Title>
        <isAuthor rdf:resource="&TestDocumentOntology;Author_One"/>
        <isAuthorOf
rdf:resource="&TestDocumentOntology;Short_Message_Service_Test"/>
        <workedDepartment
rdf:resource="&TestDocumentOntology;Test_Dept"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Tester_Three -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Tester_Three">
        <rdf:type rdf:resource="&TestDocumentOntology;Person"/>
        <Surname rdf:datatype="&xsd;string">Sayar</Surname>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#Tester_Two -->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;Tester_Two">
        <rdf:type rdf:resource="&TestDocumentOntology;Person"/>
        <Age rdf:datatype="&xsd;int">23</Age>
        <Title rdf:datatype="&xsd;string">Junior</Title>
        <Surname rdf:datatype="&xsd;string">Sayar</Surname>
        <Name rdf:datatype="&xsd;string">Vedat</Name>
        <isAuthor rdf:resource="&TestDocumentOntology;Author_Two"/>
        <isAuthorOf
rdf:resource="&TestDocumentOntology;Terminal_Configuration_Test"/>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#create_command -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;create_command">
        <rdf:type rdf:resource="&TestDocumentOntology;Command"/>
        <commandName
rdf:datatype="&xsd;string">Create</commandName>
        <commandDescription rdf:datatype="&xsd;string">Create
komutu prepaid aboneyi IN platformunda yaratmak i&#231;in
kullan&#305;l&#305;r.</commandDescription>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost/TestDocumentOntology.owl#delete_command -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;delete_command">
        <rdf:type rdf:resource="&TestDocumentOntology;Command"/>
        <commandName
rdf:datatype="&xsd;string">Delete</commandName>
        <commandDescription rdf:datatype="&xsd;string">Delete
komutu IN platformundaki prepaid aboneyi silmek i&#231;in
kullan&#305;l&#305;r.</commandDescription>
    </owl:NamedIndividual>



    <!-- http://localhost/TestDocumentOntology.owl#modify_command -
->

    <owl:NamedIndividual
rdf:about="&TestDocumentOntology;modify_command">
        <rdf:type rdf:resource="&TestDocumentOntology;Command"/>
        <commandName
rdf:datatype="&xsd;string">Modify</commandName>
        <commandDescription rdf:datatype="&xsd;string">Modify
komutu IN platformundaki prepaid abonenin &#246;zelliklerini
de&#287;i&#351;tirmek i&#231;in
kullan&#305;l&#305;r.</commandDescription>
    </owl:NamedIndividual>
</rdf:RDF>



<!-- Generated by the OWL API (version 3.0.0.1413)
http://owlapi.sourceforge.net -->
```