A Query System for UNESCO's World Heritage at the WWW

Elena Viñuela Diaz

Kgs. Lyngby 2004
IMM-THESIS-2004-29

A Query for UNESCO's World Heritage at the WWW

Elena Viñuela Diaz

Kgs. Lyngby 2004

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Abstract

The Web is probably the largest and richest information repository available today. Search engines are the common access routes to this valuable source but as they only do a keyword search sometimes their role is limited to the retrieval of potentially relevant documents but they cannot "understand" their semantic.

The aim of this thesis is to develop a semantic query system for World Heritage sites on the WWW. Web pages are usually semi-structured, and therefore very difficult to query for information. Advanced extraction processes of the information needs to be performed. This study evaluates an ontology driven approach for extracting reliable information from web pages about World Heritage.

An ontology that models the important concepts has been constructed, based on the analysis of the World Heritage domain. The ontology language DAML+OIL has been chosen for the ontology.

A prototype web-based application has been developed to perform the whole process, from getting the user's query to presenting the results of the semantic extraction processes.

Keywords: Information Extraction, Information Retrieval, annotations, ontology, DAML+OIL, Knowledge base, Internet, World Heritage

Preface

This master thesis has been written for the Informatics and Mathematical Modelling (IMM) department at the Technical University of Denmark (DTU). The study has been carried out in the period between 13rd October 2003 and 30th April 2004.

Work on the thesis has been supervised by prof. Jørgen Fisher Nilsson and assoc. prof. Hans Bruun.
I would like to thank both for their valuable help and advice during the whole process of this thesis.

Kongens. Lyngby, 30 April 2004

Elena Viñuela Díaz

Acknowledgments

I would like to thank my supervisors at DTU Hans Bruun and Jørgen Fisher Nilsson as well as my supervisor in Spain Angel Neira Alvarez for their support and for giving me the chance to develop this project in collaboration with IMM department.

I would like to add that I am very grateful to the Technical University of Denmark and the Escuela Politécnica Superior de Ingenieros de Gijón (University of Oviedo) for giving me the opportunity to come to Denmark to study.

I would like to thank Sheffield University for the use of GATE software and the GATE team for their invaluable technical support. Particularly I would like to mention Kalina Bontcheva, research associate in the Natural Language Processing Group at Sheffield University.
Also in that respect I would not like to forget all the people that is part of the GATE-Discuss mailing list. They have been very helpful for solving doubts.

Sincere thanks as well to Stanford Medical Informatics for the use of Protégé and to the people belonging to Protégé discussion mailing list.

My personal gratitude to my family and friends for their emotional support and constant encouragement. Finally, I owe my best and special thanks to my boyfriend Thomas Pedersen who has helped me a lot with the English and who has been taking good care of me during these hard months.

Overview

Contents

List of figures

Chapter 1: Introduction

This introductory chapter gives an overview of the general objectives of this master thesis. It also presents the different phases followed in the development of this study, providing a brief explanation on each one.

1.1 Motivation and thesis definition

1.1.1 Motivation

Usually data on the Web comes in the form of html pages which can be understood for the human but it is impossible for machines to do the same since the pages lack of a known schema. Machines can not find "meaning" in html web pages. Therefore, to extract data from web pages requires knowledge of both their structure and contents. With the growth of the WWW information extraction has become very important. Semantic Web and ontologies are also concepts that are been very actively researched during the recent years.
This project will try to deal with all these matters and apply some of these new techniques to build a query system that performs "semantic" search over a specific domain.

1.1.2 Thesis definition

A general outline of the main goals of this study is given below; later on some of the concepts mentioned will be briefly introduced.

- Use Information Extraction (IE) techniques to automatically extract relevant and reliable information from online documents with respect to World Heritage (WH) sites.
- Design an ontology specially for the World Heritage domain.

- Use ontologies to drive the information extraction process.
- Implement a basic prototype (in the shape of a search engine) that generates html pages in response to user requests about WH sites.
- Populate and maintain for further inference a structured or semi-structured data store with the knowledge extracted in the IE process.
- Make surveys about the main concepts researched in this study (information retrieval, information extraction, ontologies and semantic web)

By the words "information extraction" it is meant the process of identifying relevant fragments in documents while discarding extraneous text. Once the information is extracted in this manner, it can be manipulated in many different ways.
In the goals above it was mentioned that the IE process should be made over online documents, also called web documents. But there are several types of documents that can easily be found in the Web nowadays (html, images, pdf, doc, ps and so on). This thesis will focus in html documents.

Knowledge about the application domain is, in general, one of the most important cornerstones of successful software projects. In this thesis it is even more important since the characteristics of the domain will drive the decisions taken in the design of the system. It is very important to gather a good understanding of the concepts that are relevant to the working domain and it is also very useful to make some kind of domain model. Sometimes it is enough to present that model in paper but it is much better to have models that can be directly translated into a Java program. Here is where the construction of an "ontology" comes up. An ontology is a collection of domain concepts and their relationships. Further information about this concept, a survey about ontology editors, the choice of software and the ontology for the World Heritage domain will be topics covered along this master thesis.

1.2 Methodology Preview

The actions that are going to be followed to develop this master dissertation are mentioned in this section. This way of proceeding aims to make this master thesis to fulfil the set of goals presented in the previous section.

The first task to be performed is to do some background research to put the master thesis into perspective. Some general research about semantic web technologies will be done, focusing in some specific concepts like ontologies. Also a survey on information extraction techniques will be carried out.

In the second place some research about the tools to be used will be carried out, choosing the most suitable for the system purpose and getting the necessary skills to use the technology.
The next step to be taken is to follow the stages of formal methodologies for building software. These stages will be: system analysis phase (what the system should do), system design (how the system will work) and system implementation.

Once the product is built it should be tested properly in order to know how well it performs and to know if the system successfully achieves its goals.

1.3 Overview of the Study

This section explains how this study has been organized and gives the structure of the whole document[1].

This master thesis has been accomplished following the software development process: analysis, design, implementation and test phases. Each of these phases will be considered as a different part of this study. Furthermore, in order to provide the reader with a theoretical framework a whole section containing summaries of the main research done is given.

This document has been divided into five main parts that group chapters with similar contents. Their titles are sufficiently explanatory for themselves. These parts are the following:

PART I: SYSTEM ANALYSIS (chapters 2 and 3)
PART II: SYSTEM FOUNDATIONS (chapters 4, 5 and 6)
PART III: SYSTEM DESIGN & IMPLEMENTATION (chapters 7 and 8)
PART IV: SYSTEM TEST & CONCLUSION (chapters 9 and 10)
PART V: APPENDICES

A brief overview of each chapter's content is given below:

- **Chapter 2, Domain Analysis:** Both description and analysis of the system domain, which is the World Heritage sites, are given in this chapter. The scope of the system is partially defined within this chapter.
- **Chapter 3, Requirements Specification:** The requirements and functionality that the application should fulfil are described along this chapter. Therefore, laying down more detail about the scope of the system.
- **Chapter 4, Semantic Web & Ontology Survey:** This chapter summarises the information gathered during the research process about Semantic Web and ontologies.
- **Chapter 5, Information Retrieval & Information Extraction Survey:** This chapter provides a background of existing research in the fields of Information Retrieval and Information Extraction.
- **Chapter 6, Annotations Survey**: Research made about the annotation process is presented in this chapter.

[1] The same information, but summarised, can also be found under the title *Overview* prior to the table of *Contents*.

- **Chapter 7, System Design:** On the basis of the problem analysis done, this chapter describes how the application is going to be built. All the information about design decisions is gather here.
- **Chapter 8, System Implementation:** This chapter presents some details about the system implementation and the tools that have been used.
- **Chapter 9, System Testing:** The test of the prototype of the system is presented in this chapter.
- **Chapter 10, Conclusion:** This chapter draws the final conclusions got from the development of this master thesis and presents some suggestions for future work to be done.

Right after the last chapter some other useful information is included: the list of references, a very complete glossary and a list of abbreviations (LOA).

Some of the terms in the glossary have been extracted from online computing dictionaries like Webopedia [K], whatis?com [L], FOLDOC (Free On-line Dictionary of Computing) [M] or Die.net [N].

Finally some extra information can be found in the appendices part.

NOTE: The source code is not included in this document due to its extension. It is provided in a separate document called *SourceCode.doc*.

Part I

SYSTEM ANALYSIS

Chapter 2: Domain Analysis

The firs step within a system analysis phase is always to make a detailed analysis of the working domain. And so the purpose of this chapter is to provide a good understanding of the World Heritage domain, its features and possible problems. This chapter will contribute, to a large extent, to the subsequent development of the system.

To start with some basic information about World Heritage and UNESCO will serve as an introduction to the subject. After that a detailed survey about the characteristics of a site (talking now about "virtual" sites in the WWW, and not about the physical ones) will be presented; including significant URLs, keywords, structure of a site and a preliminary taxonomy.
Some Entity Relationship (ER) diagrams, gathering the knowledge acquired, will help to clarify the main concepts of the domain and their relationships. Finally a description of the problems inherent in the domain will be given.

2.1 Introduction: WH background

Protecting natural and cultural properties of outstanding universal value against the threat of damage in a rapidly developing world [A]

2.1.1 World Heritage Organization & World Heritage List

UNESCO´s World Heritage is an organization that has the aim of making people aware of the existence and present situation of some special sites located all around the world.

World Heritage sites are areas of "outstanding universal value" for their natural features, their cultural value, or for both natural and cultural values. Any nation (also known as State Party in the convention terminology) that participates in the *World Heritage Convention* may nominate a site. It can also happen that more than one State Party nominates and manages a site (also known as property), which is then called *transboundary property*.

The *World Heritage Convention* was adopted by UNESCO in 1972 and it is founded on the premise that certain places on Earth are of "outstanding universal value" and as such should form part of the common heritage of humanity. Its main purpose is to define the cultural and natural sites of the world which represents our common heritage and which would represent an irreplaceable loss should it disappear. Another one of its purposes is to describe the function of the *World Heritage Committee.* As of November 2003, 177 States Parties have signed the Convention.

There is an official list that groups all these sites. This list is called the **World Heritage List** and includes **754** properties forming part of the cultural and natural heritage which the *World Heritage Committee* considers as having outstanding universal value. These include 582 cultural, 149 natural and 23 mixed properties in 129 States Parties (some countries are signatories to the World Heritage Convention but do not yet have any sites on the List). There are also 13 World Heritage transboundary properties but none of them are managed by more than two States Parties. The rest of the properties are managed by a single State Party[2].

There is also a parallel list where sites in urgent need are placed. This list is called the **List of World Heritage Sites in Danger** and, as of November 2003, includes 35 properties.

UNESCO's World Heritage mission is:

- to encourage countries to sign the *Convention* and ensure the protection of their own natural and cultural heritage
- to encourage States Parties to the *Convention* to nominate sites within their national territory for inclusion on the **World Heritage List.**

The *World Heritage Committee* mentioned above has been in place since 1976 and is made up of representatives of 21 States Parties elected by the General Assembly (made up of all States Parties). The main duties of the Committee are:

- to select new sites for the World Heritage List from those nominated by each country;
- to monitor the state of conservation of sites on the List;
- to decide in cases of urgent need which sites on the List should be placed on the List of World Heritage Sites in danger; and,

[2] This information can be found in the publication of the World Heritage List at: http://whc.unesco.org/archive/WHList03-ENG.pdf

- to administer the World Heritage Fund for the protection of sites on the World Heritage List.

This Committee meets once a year to discuss all matters relating to the implementation of the Convention and in particular those matters relating to the duties mentioned above.

To conclude what makes the concept of World Heritage so exceptional is its universality. World Heritage sites belong to all people around the world, independently of the place in which they are located.

2.1.2 Criteria for selection

These criteria define "outstanding universal values" that are the fundamental features for a nominated property to qualify for inscription in the **World Heritage List**. These criteria are explained in detail in the Operational Guidelines (http://whc.unesco.org/opgulist.htm) and are revised regularly for matching the evolution of the World Heritage concept itself.

For a property to be included on the **World Heritage List** as cultural heritage, the *World Heritage Committee* must find that it meets one or more of the following criteria. Sites nominated should therefore:

i. *represent a masterpiece of human creative genius, or*

ii. *exhibit an important interchange of human values over a span of time or within a cultural area of the world, on developments in architecture or technology, monumental arts, town planning or landscape design, or*

iii. *bear a unique or at least exceptional testimony to a cultural tradition or to a civilization which is living or has disappeared, or*

iv. *be an outstanding example of a type of building or architectural or technological ensemble, or landscape which illustrates a significant stage or significant stages in human history, or*

v. *be an outstanding example of a traditional human settlement or land-use which is representative of a culture or cultures, especially when it has become vulnerable under the impact of irreversible change, or*

vi. *be directly or tangibly associated with events or living traditions, with ideas or with beliefs, or with artistic and literary works of outstanding universal significance (a criterion used only in exceptional circumstances, and together with other criteria) [A].*

Equally important are the authenticity of the site and the way it is protected and managed. The *World Heritage Committee* also has to test that before including a property in the list.

For a property to be included on the **World Heritage List** as natural heritage, the *World Heritage Committee* must find that it meets one or more of the following criteria and fulfils the conditions of integrity. Sites nominated should therefore:

i. *be outstanding examples representing major stages of the earth's history, including the record of life, significant ongoing geological processes in the development of landforms, or significant geomorphic or physiographic features, or*

ii. *be outstanding examples representing significant ongoing ecological and biological processes in the evolution and development of terrestrial, fresh water, coastal and marine ecosystems and communities of plants and animals,*

iii. *contain superlative natural phenomena or areas of exceptional natural beauty and aesthetic importance, or*

iv. *contain the most important and significant natural habitats for in situ conservation of biological diversity, including those containing threatened species of outstanding universal value from the point of view of science or conservation [A].*

Finally to mention that to be inscribed in the **World Heritage List** a site must satisfy at least one of these criteria, either natural or cultural. As mentioned before, currently there are 23 properties (or sites) that satisfy both types[3].

2.2 WH site's features

As mentioned before, there are 754 sites currently inscribed by the World Heritage Committee in the official **World Heritage List**.
On the other hand, there are thousand of web pages in the Web holding information about these sites.
From now on when referring to WH site it will be implicit that it is being referred to WH web sites.

2.2.1 More significant URLs

A preliminary research stage was performed in order to find the most relevant places in the WWW containing information about World Heritage sites.

Some of the most relevant information sources found about sites inscribed on the World Heritage List were the following:

- *WHSites SITES (World Heritage Centre)*

[3] A complete list of mixed sites with natural and cultural criterion inscribed on the World Heritage List can be found at http://whc.unesco.org/sites/mixed.htm#debut.

http://whc.unesco.org/nwhc/pages/sites/main.htm

This page is under the official site for World Heritage, called the **World Heritage Centre** [A]. It offers a world map where one can select a region by clicking on it. By doing that the user is led to another page where all the sites belonging to the selected region are shown grouped by both country and continent. Once there it is just to click the link of the site and one will be automatically forwarded to the information. According to this web site, the list will be updated following the next meeting of the Committee in July 2004.

- *WH sites brief descriptions (World Heritage Centre)*

  http://whc.unesco.org/brief.htm

  This is not exactly a web site itself but part of the official site mentioned above. It differs from the former resource in that all the sites are offered to the user in the same web page, together with a brief description of each. It also offers a published version in PDF to download.

- *Protected Areas Program – World Heritage Sites*
  http://www.wcmc.org.uk/protected_areas/data/wh/index.html

  This page works in a similar way as the first one mentioned above. An index page offers all the links to the sites classified by country. When somebody clicks over a link he is forwarded to a new page containing all the information about the site. Actually the information offered in these pages is richer and more detailed than the one in the official site, but this one only comprises sites that are under the Protected Area Program so the rest of the sites are missing. This program is one of the several ones that the UNEP World Conservation Monitoring Centre is running.

- *UNESCO World Heritage List* http://www.thesalmons.org/lynn/world.heritage.html

  This URL is a personal home-made page that shows all the information about World Heritage sites including some extra links for every site. It also provides with an index in which sites are grouped by country and ordered by year of inscription. When one clicks over a certain site another web page is displayed with data of all the sites that belong to the same country as the site previously clicked. In other words, there is not a single page for every WH site, but it groups all the information regarding sites under a country.

- *World Heritage Explorer Prototype* http://www.vrheritage.org/engine/explorer/

  This URL holds a prototype of a WH explorer developed by the non-profit-making VRheritage.org association. Some functionality is not yet developed and some other requires a membership. At the end it offers the same information as the official site but the main feature here is that it provides the user with three ways of searching: by theme, by region or keyword search. The search by region is exactly the same as found in the other URL mentioned, just with a different formatting. The search by theme option is just a classification in which the sites are matched. The keyword search is more or less similar to the functionality our system aims to have offer.

  This prototype can help the new system that is going to be built by comparing some of the search results. But that will be in the system test phase.

Those above were just the most relevant web pages found at this stage of the survey but many other pages about this topic can be found in the WWW. Some of them are specific for a country or a region and some others are written in other languages rather than English. Some WH sites even have their own and exhaustive home-page, like for instance the famous Tower of London or Kronborg Castle (www.tower-of-london.com and www.kronborgcastle.com respectively).
So, in order to avoid having to deal with a huge amount of websites, two constraints were made for this preliminary research: web pages had to be written in English and had to contain (with more or less detail) information about all the sites inscribed in the list.

From the URLs mentioned before the one chosen to work with was the first of the list - the one offering information about sites under the **World Heritage Centre (WHC) website** domain; first because it contains the official information about the different properties but mainly because its pages follow a certain fixed structure in their content. As said in the introductory chapter, by working with this kind of pages it is more likely to get precise recognitions and extractions. Another reason to choose this one is because it gives a single web page for every single WH site. Being the official site also ensures that the information is going to be updated regularly.
From now and on when referring to the **WHC web site** in fact it will be meant the set of pages that are under this domain and contain information about WH sites. The domain itself is huge and offers all kind of sections and information.

The second source mentioned could not be chosen because, even though it offers the more relevant information about each site very well summarized, everything is written on a single page. That would make the information extraction task very difficult and complex.

The Protected Areas Program web page was also dismissed because it does not give information about all the sites contained in the World Heritage List.

The personal web site containing data about WH was soon rejected (even though it provides with very thorough information) because it is not official and probably it will not be updated in the same way as an official site is. Also because, although it gives a lot of information, it is shown in a slightly messy way not following any pattern or structure. Some of the links it provides with are for web pages in other languages too.

2.2.2 WH Keywords survey

A short period was spent on surfing the Web again, looking this time at several web pages within the *World Heritage Centre* website domain that was the one chosen to make the study.
This was made in order to achieve a better understanding of the domain and getting to know the more relevant keywords that form part of the “WH language”.

Figure 2.1 shows a table containing some of the keywords collected. It speaks for itself, for every search concept there is a list of keywords found and a list of sites (matching the search concept) where these keywords were used.

WH KEYWORDS SURVEY		
Search concept:	(keywords)	(sites visited)
Castle		
	Fortress, château, residence, tower, moat, drawbridge, gardens, defense, settlement, outer wall, curtain wall, crag, gatehouse, king, prince, duck, swan	Durham Castle, Kronborg Castle, Litomysl castle, The Castles of Augustusburg and Falkenlust Brühl, The Mir Castle Complex, Beaumaris Castle
Cave		
	Prehistoric, stone, paintings, fresco, relief, mural, grotto, gallery, passageway, chamber, bison, fawn, wild boar, carving, cliff, stalactite, stalagmite, erosion, temple, groundwater flows, coal mine	Altamira Cave, Mogao Caves, Ellora and Amanta caves, Caves of Elephanta, Cave of Aggtelek, Skocjan Caves, Mammoth Cave National Park, Yungang Grottoes
Forest		
	Rainforest, virgin forest, Atlantic forest, mangrove forest, tropical forest, tree, shrub, thicket, creek, cedar, hectare, reserve	Australian Central Eastern Rainforest Reserves, Sundarbans Mangrove forest, Southeast Atlantic Forest Reserves, Sinharaja Forest Reserve, Wet Tropics of Queensland, Comoé National Park, Sinharaja Forest Reserve
Island		
	Islet, beach, sand, dune, cliff, geological, rocks, wildlife, seabird, penguin, albatross, seal, elephant seal, volcanic (limestone, granite), reef, coral, marine, algae, lagoon, mollusc	Fraser Island, Macquarie Island, Henderson Island, Gough Island, Hinchinbrook Island, Heard Island and McDonald Islands, New Zealand Subantarctic Islands, Lord Howe Island Group, Robben Island, The Galapagos Islands, Aeolian islands
Lake		
	Freshwater, shore, stream, river, fish, cascade, waterfall	Lake Baikal Basin, Lake Malawi National Park, Plitvice Lakes National Park
Monastery		
	Church, abbey, basilica, cloister, convent, chapel, sacristy, refectory, stained glass, altar, cross, tomb, crypt, mausoleum, sarcophagus, gargoyle, pantheon, cell, vault, dome, cupola, belfry, nave, façade, turret, monks, calligrapher, Cistercian, "holy"	Monastery of Batalha, Monastery of the Hieronymites, The Hurezi Monastery, Maulbronn Monastery, Monastery of The Escorial, The Monasteries of Haghpat and Sanahin, The Monastery of Geghard
Mount		
	Peak, mountain, crest, massif, ridge, geological, altitude, era, reserve, park, forest, rock, cave, valley, stream, waterfall, canyon, cliff, gorge, volcanic, lava, sandstone, glacial, "holy"	Mount Huangshan, Mount Wuyi, Mont Perdu, Mount Athos, Mount Kenya, Mount Nimba, Mount Emei

Figure 2.1 – Table of WH keywords

These keywords (and many others) can serve as an example of different interests that a potential user of the system can use as query concepts. Combining some keywords with names of countries, architectonic styles or time periods for instance can be a way

of formulating some potential user's requests, as seen in Figure 2.2. Generally it is not recommended to use verbs and adverbs as keywords.

Some of these keywords will be used during the development stage and later on during the testing phase.

User interest	Search query
Sites that have a volcano	*volcano*
Sites that have a tower	*tower*
Sites where you can find an specific animal	*crocodiles* *whales* *bears …*
Castles that are in Germany	*castle Germany*
Spanish cathedrals relevant for having tombs	*cathedral tomb Spain*
Caves relevant for having paintings	*cave paintings*
Caves relevant for being used as sanctuaries	*cave sanctuaries*
Gothic style cathedrals in Europe	*cathedral gothic Europe*
Buddhist temples everywhere	*temple Buddhist*
Cultural landscapes everywhere	*cultural landscape*
A specific site which name is already known	*Kronborg*
Sites in a specific country, region, area, place…	*Sri Lanka* *Asturias* *Luleå ...*

Figure 2.2 – Potential queries to the system

2.2.3 Typical structure of a WH site in the WWW

Once the working web domain is chosen next step is to analyze its structure in detail. As said before, the set of web pages that will be used within the scope of this project are the ones belonging to the World Heritage Centre web site [A].
To be more precise they are under the URL **whc.unesco.org/sites/** and their names are currently based on the official number (which for the time being can vary from 1 to 1130) given by the *Committee* follow by the extension of the web page. For instance *http://whc.unesco.org/sites/925.htm* is a valid name for those pages.

Below are some examples of what those web pages look like. They are different screenshots from the web domain of study. In each example some differences or special cases are shown.

Figure 2.3 shows the typical layout of a WH site on the web. It has all the typical features that a site can have. Most of the sites look like this one below.

Figure 2.4 shows a WH site in which the location is not given and also, as it is inscribed in the list of sites in danger, it offers more information that the standard ones. Quite a number of sites do not provide with location and coordinates information. The

number of sites also inscribed in the parallel List of World Heritage in Danger is not so high.

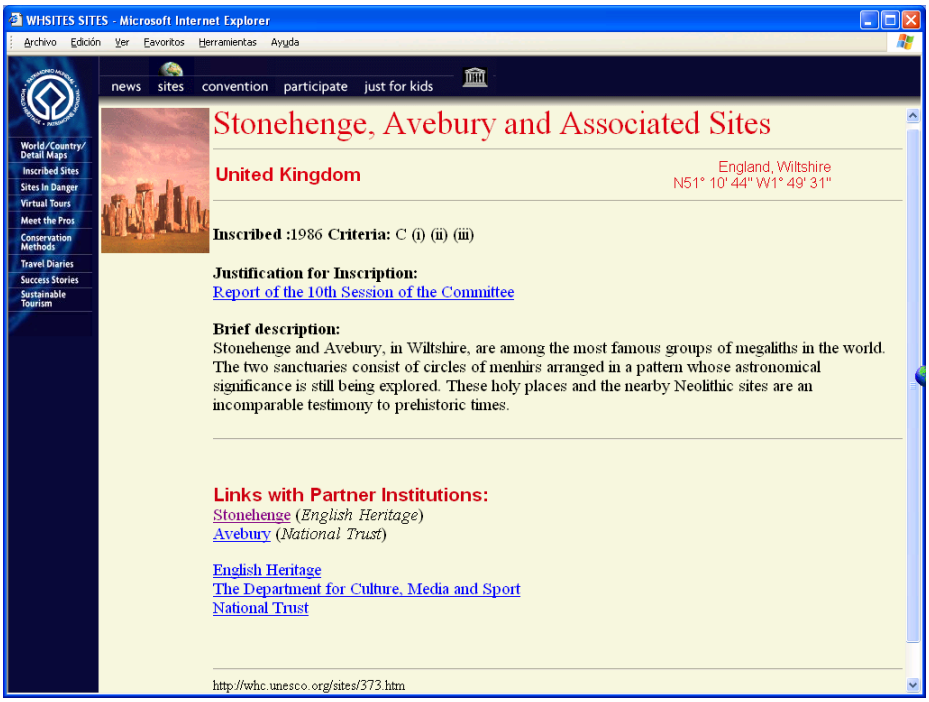


Figure 2.3 – Stonehenge UK (WHC screenshot)

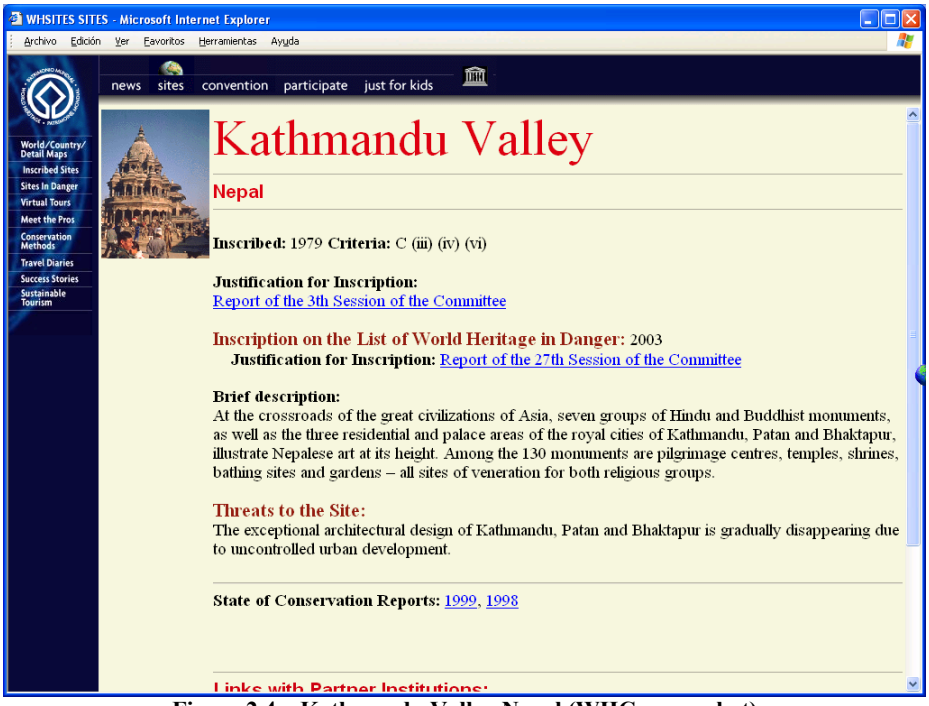


Figure 2.4 – Kathmandu Valley Nepal (WHC screenshot)

Figure 2.5 shows an example of a site that fits in both cultural and natural categories since it has criteria of both types[4]. This property does not have any partner institutions to link with.

[4] A complete list of sites with Natural and Cultural criterion mixed on the World Heritage List can be found in http://whc.unesco.org/sites/mixed.htm#debut.

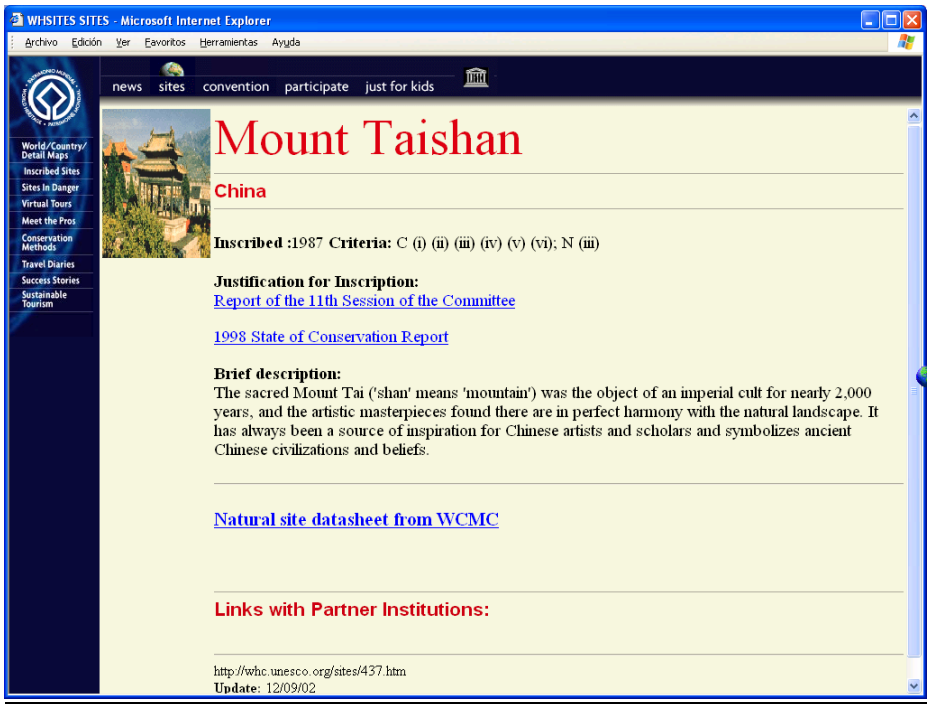


Figure 2.5 – Mount Taishan China (WHC screenshot)

Figure 2.6 shows the rare case of a site that has more than one inscription year and multiple locations. Details about locations are given in other pages through a link.

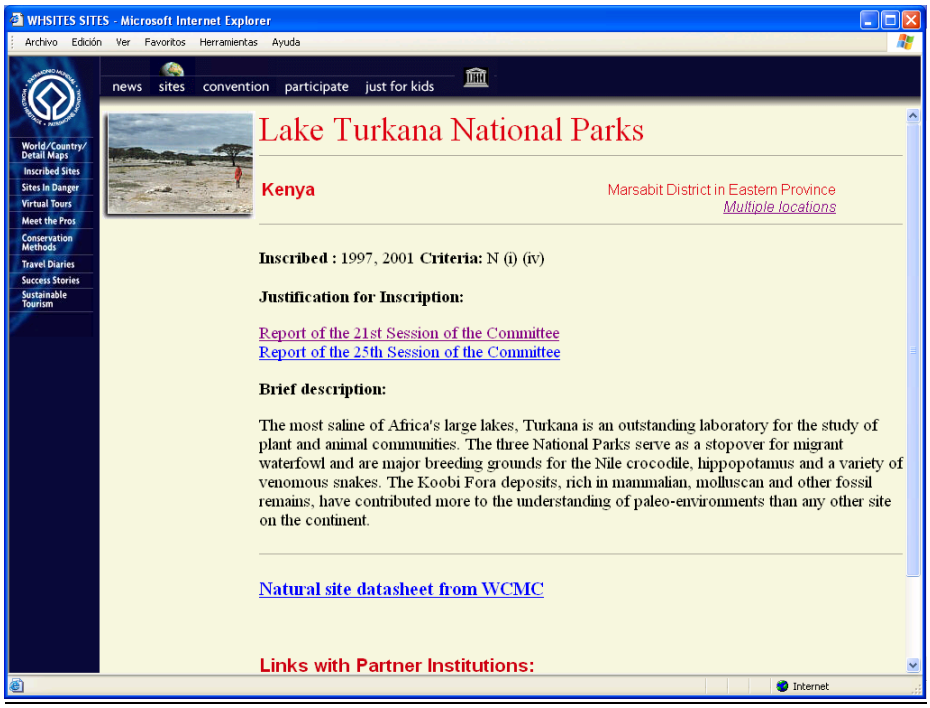


Figure 2.6 – Lake Turkana National Parks Kenya (WHC screenshot)

These screenshots examples cover mainly all the presentation possibilities that a property can have within this domain.

A study of the information contained in these pages and the way it is related lead to the following schema containing the typical structure in these pages:

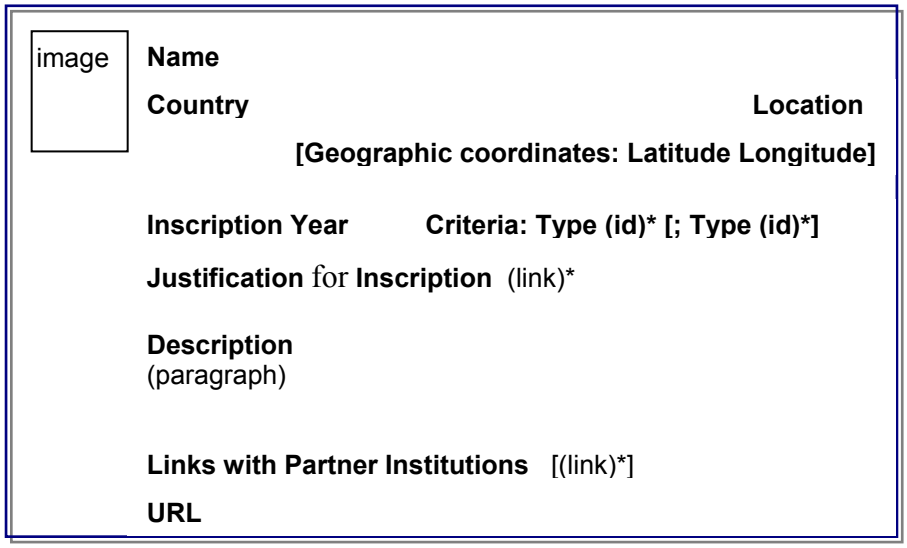


Figure 2.7 – Structure of a WH site in the working domain

FINDINGS of the study

These are the main findings of this study:

- Name, country (State Party), description, inscription year, type and identifier of criteria and URL are the main data to focus on. They constitute compulsory information about a WH site.
- Some sites can have more than one inscription year. This can happen because:
  a) the property was extended by adding more relevant places or areas to it.
  b) the property was considered to fulfil more criteria and so was inscribed again in the List (considering this time the criteria found applicable).

  It can also happen that in the process of revising a property its name can change as request of the State Party that manages the site.

  Some examples of sites with more than one year of inscription are for instance: Lake Turkana National Parks in Kenya, Tongariro National Park in New Zealand, Butrint in Albania or the Historic Centre of Lima in Peru, among others.
- Not all the sites provide a location. This is not so uncommon. On the other hand, there are sites with multiple locations. For instance James Island and Related Sites in Gambia or the Brazilian Atlantic Islands: Fernando de Noronha and Atol das Rocas Reserves, among others.
- Coordinates don't have to be required in all the sites. Many sites do not provide with this information.

It can happen that a property provides with location information but not with geographic coordinates. The other way around could not be found.

- Not all the sites have links with Partner Institutions.
- Some sites have in the *Justification for Inscription* part not only the link to the corresponding report (for a site to be included in the list a report has to be written) but also a justification of every criterion given before in the criteria section. See Rock Shelters of Bhimbetka at whc.unesco.org/sites/925.htm for an example.
- Some of the sites can also be inscribed in the list of sites in danger, so those will have more information about that.

All sites provide with a small image but images are not going to be taken into account within the scope of this project, since their extraction and later storing requires a specific treatment.

Critical information

As seen before, some of the information of a property is optional. That is why the critical information to collect knowledge about will be the one that is always present. Here is a table that shows the data considered critical for a first stage of a prototype.

WH site
Name
Country
Inscription year
Criteria
Description
URL

Figure 2.8 – Critical information to collect

2.2.4 Preliminary classification

After the keywords survey, the structure analysis and a period of browsing within the website domain a preliminary taxonomy of WH sites is made.

This classification divides the sites into two main groups or categories: natural sites and cultural sites. Obviously some other classification could have been done, focusing in other criteria like geographic location or time-location for instance. But the approach of classifying sites for their category was chosen as the most suitable for a preliminary .

Classification of WH sites:

- **Natural Category**
  - Biological Interest
    - Fauna
      - Fish
      - Birds
        - Pelican
        - Heron
        - Ibis
        - Flamingoe
        - Duck
        - Geese
        - Stork
      - Mammals
        - Wolf
        - Bison
        - Lynx
        - Otter
    - Flora
      - Trees
        - Evergreen
        - Conifer
        - Aspen
        - Birch
        - Pine
      - Forest
        - Rainforest
        - Atlantic forest
        - Tropical forest
        - Mangrove forest (Ex. The Sundarbans)
        - Mountain forest
        - Virgin forest
        - Palm forest (Vallée de Mai Nature Reserve)
      - Peat bog
  - Geological Interest
    - Cliff
    - Canyon
    - Gorge
    - Passage
    - Peak
    - Volcano
    - Water related
      - Lake
      - Lagoon
      - Marsh
      - River
      - Glaciers

- Streams
- Waterfall

- General Interest

- **Cultural Category**
  - Architectonic construction
    - Cultural Landscape
      - Garden
      - Park
      - Agricultural landscape
    - Relics
      - Prehistoric relics
      - Excavations
    - Historic Centre/Area
      - Town/Town Center
      - Village
      - City
      - Ruins
        - Ancient ruins
        - Medieval ruins
    - Religious construction
      - Christian construction
        - Church
        - Abbey
        - Cathedral
        - Tomb
      - Muslim construction
        - Mosque
        - Minaret
        - Tomb
      - Temple
    - Secular construction
      - Residential construction
        - Castle
        - Palace
        - Residence
        - Mansion
      - Health construction
        - Hospital
        - Bath
        - Spa
      - Industry construction
        - Mine
        - Mill
        - Saltwork
        - Ironwork
        - Deposit
        - Pumping station
      - Technical construction

        - Bridge
        - Harbour
        - Tunnel
        - Canal
        - Aqueduct
    - Military construction
        - Fortress
        - Tower
        - Wall
        - Castle
        - Defense line
    - Public construction
        - Theatre
        - Library
        - University
        - Town hall

2.3 World Heritage Model

The final step within the domain analysis phase, after all the previous survey done, is to model the structure of the information. This will help to a better understanding of the domain.

A point to start with is an ER model, which serves as a semi-formal tool for modelling the system domain. An introductory section about ER model concepts can be found in appendix A1.

2.3.1 ER Model: Introduction

The ER Model is a conceptual data model that sees the real world as consisting of entities and relationships among them. The model visually represents these concepts by the Entity-Relationship diagram (ERD). These diagrams are very suitable to model data structures. In the next section the diagram that models World Heritage concepts will be given.

For a non experienced reader in these issues, the general concepts of the ER model are given in appendix A1.

Chen style (see Figure 2.9 for a brief summary of this notation) will be used in the ER diagram of World Heritage sites.

Just to mention at this point a brief comment about Mr. Chen's notation. In his original work, only one number appeared at each end, showing the maximum cardinality. This would not indicate whether or not an occurrence of an entity had to have at least one occurrence of the other entity. For this reason, the technique can be extended to use two numbers at each end to show the minimum and maximum cardinalities. This extension of the notation will be applied in the World Heritage ERD.

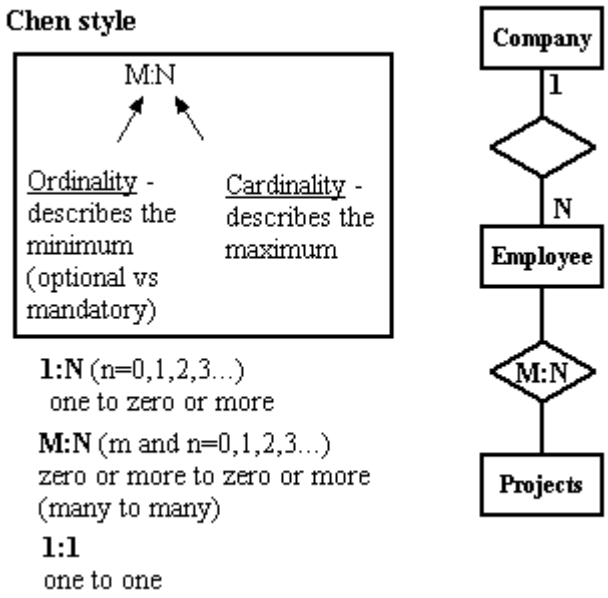


Figure 2.9 – Chen notation (source [O])

2.3.2 ERD for WH Sites

In order to help a better understanding of the information being managed, a diagram is made modelling this behaviour. This diagram is shown below:

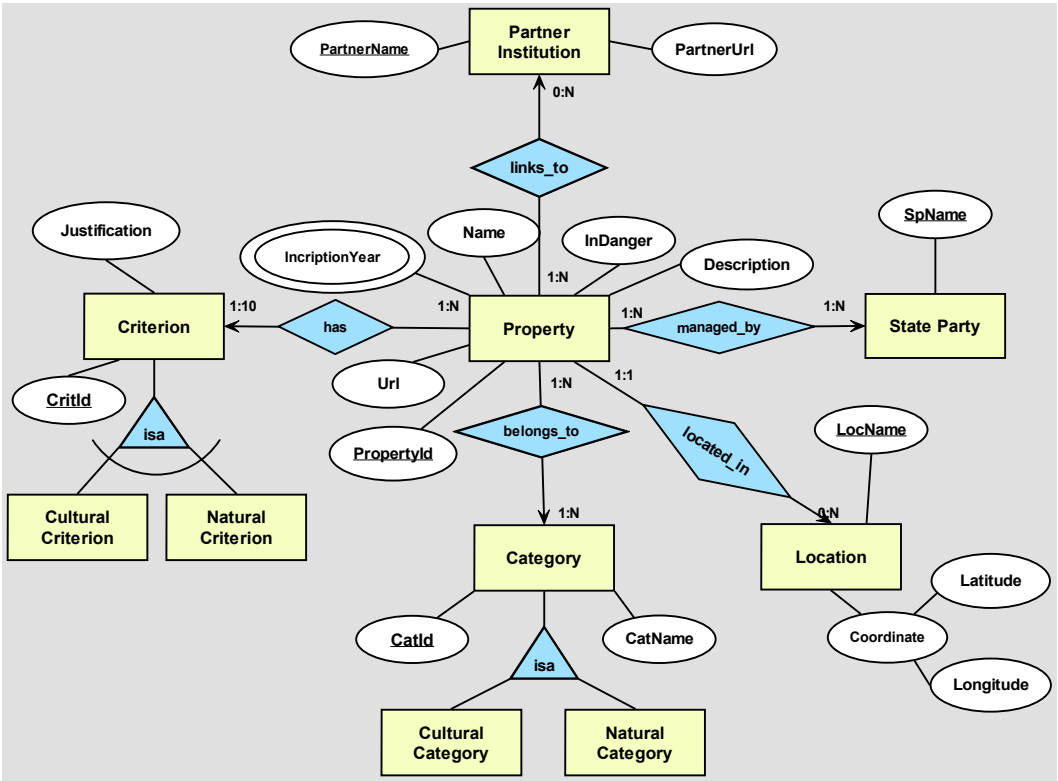


Figure 2.10 – World Heritage ERD

NOTE: To be more consistent, the terminology used by the *Convention* is the same used here (property instead of site and State Party instead of nation/country).

The ER diagram is made up of the following entity sets with its attributes:

- **Property** (PropertyId, Name, InscriptionYear, Description, Url, InDanger)
- **Category** (CatId, CatName)
  - **Cultural Category**
  - **Natural Category**
- **Criterion** (CritId, Justification)
  - **Cultural Criterion**
  - **Natural Criterion**
- **Location** (LocName, Coordinate: Latitude and Longitude**)**
- **State Party** (SpName)
- **Partner Institution** (PartnerName, PartnerUrl)

The meaning and justification to be in the model of each of these entity sets is explained below.

Property

This is the main entity set. It represents a World Heritage property and its attributes are:

- **PropertyId:** A unique identifier (key attribute) that is the original number given to a property when is inscribed in the *World Heritage List*.
- **Name**: The name of the site. This attribute cannot be chosen as the key attribute for this entity set since it was seen that in some cases the name of a property can change. For instance if the State Party managing it decides so.
- **Description:** A short text describing its outstanding universal value.
- **Url**: The URL address of the web page containing the information about the property. There is a single URL for each site in the system domain.
- **InDanger:** A boolean attribute that tells if the property is in danger or not (if in danger it will also be included in the list of World Heritage sites in Danger).
- **InscriptionYear:** The year in which a site was inscribed in the list by the *Committee*. This is a multivalued attributed, thereby representing the fact that a property can be inscribed in the *List* more than once.

Multivalued attributes should be used, as a rule, with great caution because they represent situations that can be modelled in many cases with additional entities linked by one-to-may (or many-to-many) relationships to the entity which they refer. This is the case of **Inscription Year**, which was modelled as a multivalued attribute for the sake of simplicity of the model. If it were to be modelled as an entity set (anticipating an optimum way to keep the information in a relational database) it should be like follows:

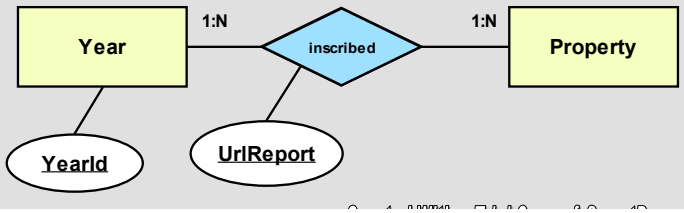


Figure 2.11 – World Heritage ERD (extension I)

Such a model of these entity sets would bring the possibility of adding an attribute to the relationship **inscribed**, thus holding the hyperlink to the correspondent report of the *Committee* where the reasons for a certain property to be inscribed in the *List* are stated. For example, *Butrint* in Albania (with PropertyId = 570) would have been related to two instances of entity **Year**, *1992* and *1999*. Every instance of the relationship would have had *http://whc.unesco.org/archive/repcom92.htm#570* and *http://whc.unesco.org/archive/repcom99.htm#570* respectively as attributes.
The report that justifies the inscription of a site in the *World Heritage List* was not considered as critical information to harvest in a first stage of the prototype. Thus, it not appears in the ERD. An additional entity set holding the features of a report could have been added to the model. Another entity set holding the image (or images) attached to a site could have been added too, thus foreseeing future needs. A complete diagram modelling all the information related to a site is given in appendix A2.

The relationships that link this entity set with the others will be explained later while describing the rest of the entity sets.

Category, Cultural Category and Natural Category

These three entity sets are explained together since they form a generalization or inheritance. Generalization hides differences and emphasizes similarities.
The entity set **Category** represents a specific group in a classification system according to the type of the site (such as island, cave, military construction, biosphere reserve, religious building, historic city and so on). A preliminary classification of categories for the World Heritage domain was made in section 2.2.4.

The model reflects, by means of an ISA relationship, the fact that all the possible categories (for a site to belong to) are split into two main groups: cultural categories and natural categories. In fact these two main groups could be again divided into many other subgroups. However, for the sake of a better understanding of the WH model the ER diagram remains as simple and small as possible.

It can also happen that a category is considered to be in both sub-groups at the same time. This fact is also reflected with this kind of relationship, an overlapping generalization. For instance an occurrence of entity **Category** could be *cave*, which can be considered both natural (due to the physical environment, stalactites stalagmites, grottoes, and so on) and also cultural (for having mural paintings, for being used as a temple and so on).

Usually properties that fulfil natural criteria are classified as belonging to one (or more) **Natural Category**. While those ones fulfilling any of the cultural criteria will be considered in one (or more) category within the **Cultural Category** entity set.

The overlapping (inclusive) ISA relationship in this part of the ER diagram solves the problem of where to locate, in a classification by category, sites like for instance *Ukhahlamba Drakensberg/Park* or the *Göreme National Park and the Rock Sites of Cappadocia* for instance. Both sites are included in the World Heritage List for fulfilling both types of criteria. Therefore, they are classified under the two types of categories.

Criterion, Cultural Criterion and Natural Criterion

Again these entity sets are explained together since they form a generalization. With these entities it is represented the fact that there are two different types of criteria, according to the *World Heritage Convention*.
The notation used (an arc across the two relationships) represents that the generalization is exclusive, meaning that a criterion can be either natural or cultural but not both at the same time.

Entity sets **Criterion** and **Property** are connected through a relationship called **has**. The real world restriction that at least one criterion must be met for a property to be inscribed and at most ten (considering the extreme case of a site belonging to both categories and fulfilling all the criteria on each) is shown by means of the cardinality. There is a maximum of 6 cultural criteria and 4 natural criteria that a site can meet.

To see again the natural and cultural criteria refer to section 2.1.2 of this chapter.

The only attributes of the entity sets **Cultural Criterion** and **Natural Criterion** are:

- **CritId:** A unique identifier for a criterion. A letter from i to vi in the case of **Cultural Criterion** and form i to iv in the case of **Natural Criterion**.
- **Justification:** This attribute is inherited from the supertype **Criterion**. It states the reason why a site fulfils a particular criterion. This attribute is optional. It is in the model to match those sites that offer more explanations than a link to the official report in the Justification for inscription section (see section 2.2.3 for further detail).

For instance the previously mentioned *Ukhahlamba/Drakensberg Park in South Africa* fulfils four criteria: criteria *iii* and *iv* from the natural ones and criteria *i* and *iii* from the cultural ones (http://whc.unesco.org/sites/985.htm).

Another approach for modelling the criteria of a property could have been as shown in the following figure:

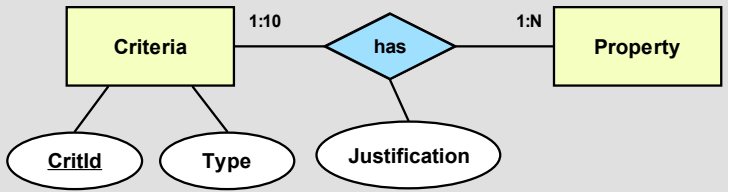


Figure 2.12 – World Heritage ERD (another approach to model Criteria)

Using only one entity set and making the distinction between the two types of criteria through and attribute called **Type**. This approach is more efficient in terms of not repeating information but is not so clear when it comes to show the real distinction between the types of criteria.

Location

By means of this entity set the general location of a World Heritage site is modelled. Its attributes are:

- **LocName:** A location is identified by its unique name so this is a key attribute.
- **Coordinate:** This is a composite attribute. Sometimes is convenient to group attributes of the same entity set that have closely connected meanings or uses. This is the case, since attributes **Latitude** and **Longitude** build up together the Coordinate attribute. If they appear they always appear together.

  This attribute is optional as it was seen from the WH Site's features survey (previous section in this chapter). If it is not given neither their sub-attributes are given.

This entity set is related to **Property** by means of the relationship **located_in** which may be zero (optional). This relationship models the fact that not all web pages containing information about WH sites provide with data about location. Moreover, if they do not have a location they will also lack of geographic coordinates. A **Property** can be located in at maximum N locations, thus solving the multiple locations issue.

Looking at the relationship the other way around (from Location to Property with the role **locates**), note that the cardinality of one-to-one means that each location is unique for a certain site. Although this is not very common, it may happen that more than one site is located in the same place, like for example in the case of *The City of Vicenza and the Palladian Villas of the Veneto* and *The City of Verona*, both situated in the region of Veneto (Italy). But then the geographic coordinates are always going to be different, that is why a location (comprising name of the place plus coordinates) is considered unique.

Another way of modelling the location of a site could have been to have an entity set just for representing geographic position and make the relationship between **Location** and this new entity set be optional. Next figure shows this alternative.

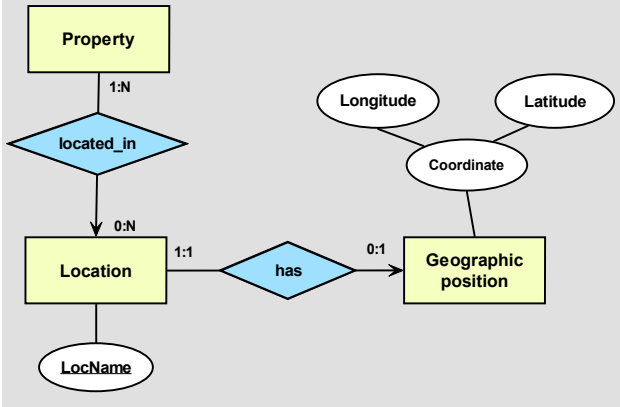


Figure 2.13 – World Heritage ERD (extension II)

State Party

This entity set represents only the countries that have a property included in the *World Heritage List* and not those ones that signed for the *Convention* but do not have a site inscribed. This is important to remark since cardinalities are based in this modelling decision.

The only attribute of this entity set is:

- **SpName:** A country is identified by its unique name so this is a key attribute.

As seen before, a site can be managed for more than one country; being this fact modelled in the cardinality of the relationship **managed_by**. A minimum of one **State Party** and a maximum of N make this relationship to be compulsory. In other words, every property has to be managed by at least one country. It could only be found a site managed by three States Parties, the *Kakadu National Park* in Australia. However, it could happen that a property would be managed for more, that is why the relationship **managed_by** has a maximum cardinality of N.
A real example could be *Pyrénées - Mount Perdu*, which is manage by France and Spain. There are plenty of examples of properties managed by only one country.

Talking a look now at the other direction of the same relationship (it would be something like **to_manage**), it is also compulsory. Although in the real world there are nations related to WH with no properties in the List (see the Overview section in this chapter for further detail) it was decided not to model this fact. Mainly because if the model had reflected this fact, the relationship between entity set **State Party** and **Property** would have been optional, thus leading to misunderstandings. Remind that what is being modelled by means of the ERD is the structure of the data presented in the web pages and evidently there are no web pages of countries which do not have a site... Besides, it was said that information about countries was considered critical because it was compulsory; having an optional relationship now would have been a contradiction.
As a summary, a State Party instance may be associated with a minimum of one and a maximum of many occurrences of entity Property.

There are plenty of examples of States Parties that managed more than one site.

Partner Institution

This entity set represents those institutions that can also hold information about a site. In a WH web page they are given in the form of hypertext links (as seen in some of the former sections). That is why the only attributes of this entity set (at least the ones relevant for the system domain) are:

- **PartnerName:** Every institution is identified by its unique name.
- **PartnerUrl:** Necessary to link the web page of the property and the web page of the partner institution.

Entities in this entity set are optional for an entity **Property**. This is reflected by means of the relationship **links_to** which is 0:N (ordinality:cardinality). It means that a **Property** can link to a minimum of zero (optional) and a maximum of N (many) partner institutions. As relationships are bidirectional it can be said that a **Partner Institution** may be associated with a minimum of one and a maximum of N occurrences of entity Property.
The minimum of one is because there is no sense in storing information about partner institutions that have nothing to deal with any of the properties.

Here is a real example of a relationship between occurrences of these entities. The partner institution *Historic Scotland* which can be found at http://www.historic-scotland.gov.uk/ appears as a hyperlink in the following sites: *New Lanark*, *Old and New Towns of Edinburgh* and *The Heart of Neolithic Orkney*.

In this section a very simple entity relationship diagram has been presented and explained. The aim at this point of the study is to show the main concepts of World Heritage sites and the relationships between them, and not to design a database schema. Some other approaches to model the data of the real world were considered but this one seemed the most clear to understand.

Variations to the model have been sketched along the explanation. A different and more detailed ERD is presented in the appendices part (appendix A2.), together with the transformation to relational tables (appendix A3.). This secondary study has been carried out in order to have a vision of the data storage schema (a schema describes the structure of a database) in case this information should be stored in a relational database. The tables in this study are not normalized because they are oriented (and optimised) to achieve fast queries over the information.

For further detail about all these matters refer to APPENDIX A.

2.4 Problem Description

So far, a description of what World Heritage is and a deep analysis of the chosen working domain have been given. In this section, it will be explained what the problem of making semantic queries over this domain is as well as what the system is aimed to achieve. Therefore, stating the scope of this system.

The *World Heritage Centre* web site [A] has its own search facility, which is briefly going to be analysed as a case study and as an introduction to the main problems of the chosen domain.

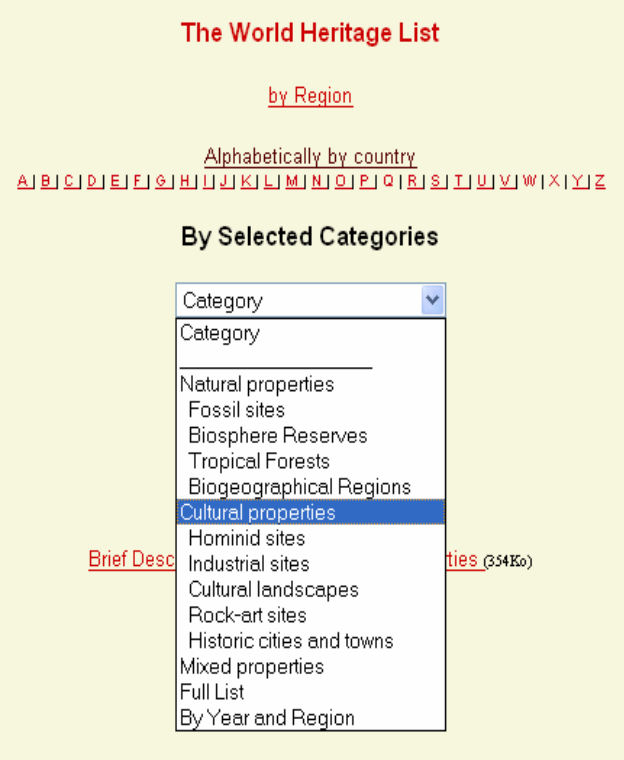


Figure 2.14 – Search functionality (WHC Screenshot)

The figure on the left is a screenshot from the search functionality of the working domain. The *WHC* web site offers visitors three different ways of locating sites:

1. Search by region
2. Search by country
3. Search by category

The first two options are pretty similar. In both cases the user is presented with a list of sites and he just has to click in the ones of his interest. In the first option all the sites are shown in alphabetic order of country, while in the search by region they are shown (also in alphabetic order of country) along different pages but grouped by region (it considers the following regions, not necessarily subdivisions by continent: Africa, Arab States, Europe and North America, Latin America, Asia and the Pacific). So it can actually be considered that the "Search by region" result lists are subgroups of the "All sites by country" list.
The "Search by category" option groups the sites in categories and offers lists with them, then the user has to navigate all the sites to find the exact topic in which he was interested. As it can be seen from the screenshot the number of categories is quite small, so only a few chances to choose from are offered. To sum up, in the end all the search options the *WHC* web site offers are the same: browse a list of sites that is presented to the user and click in every site to check if it is what he wanted.

What happens if the user wants to know about, for instance, sites in Europe that are relevant for having a castle? Obviously he can search by region and choose "Europe" and then he can look at all the names of the sites and click in those ones that contain the word "castle". But he is losing a lot of time by doing this and probably he is missing a lot of interesting sites with castles. For example the wonderful castles of Sintra in Portugal that are under the site called *Cultural Landscape of Sintra* or the

royal castle in *Cracow's Historic Centre* in Poland. In conclusion, it is important for a search to be able to "see" the contents of a site. This way the user could search, with much more freedom, for any concept included in the web page of a WH site.
What happens if the categories offered in the *WHC* search are not matching exactly the user's interests? What happens if the user wants to know about sites containing crocodiles? A kind of keyword search becomes necessary.
One of the system goals for solving this kind of problems is to implement a search engine that allows the users to make queries very easily, just typing some keywords, and offers them relevant results about World Heritage sites. The key question here is in the word "relevant", the content of the results has to be as "semantically" precise as possible.

Another one of the goals proposed for the system is to populate any kind of data storage with the information extracted from the user's requests. The idea underlying the prototype search engine is that users populate the data storage by making successive requests to the system. If the information being requested is already in the system it will be presented to the user immediately (after have been formatted). If it is not yet in the data store the system has to search for this information and extract it from the online source, populate the data storage with it and then present it to the user.

Closely related to these matters is another problem of the *WHC* domain: its enormous variety. All the sites have some features in common like the name, description, criteria for inscription and so on. But on the other hand, each of them is somehow unique in other features. For instance, a cathedral would probably belong to some architectural style, have a certain number of architectural elements (naves, cupolas, stained glass window etc.), be constructed in a certain period and many other features that are almost impossible to control. It would make no sense to talk about these features in a site like, for instance, a cave or a forest.
Properties inscribed in the *World Heritage List* are radically different from each other. Therefore, they have a lot of features that can only be relevant for a certain subset of the sites. How should one model all this variety?

To conclude, it is important to make a good model of the World Heritage concepts and this is not so trivial. The technique that will be used to model this domain is ontologies. Another one of the goals of this project is exactly that, to design an ontology of the domain and use it to extract "meaning" from the unstructured web pages. How this could be done will be seen and studied in further chapters.

Chapter 3: Requirements Specification

Together with the Domain Analysis and also based on it, this chapter sets up the system analysis phase.

System requirements will be partitioned into **functional requirements** and **non-functional requirements**. Functional requirements are associated with specific functions, tasks or behaviours the system must support. Non-functional requirements are constraints on various attributes of those functions or tasks. That is the approach taken for the requirements in this study.

The following sections gather the two different types or requirements. Finally a summary of all system requirements is given.
But before starting to dig into these matters a brief section at the beginning of this chapter will identify who the end-users of the system will be.

3.1 Identifying end-users

To identify the profile of the end-user of the system is important prior to establishing some of the system requirements.

An end-user can be any person interested in the World Heritage topic wishing to obtain some information about it. The user has to have access to the Internet and be able to use a web browser. No relevant skills are required to a user of this system.
If a user is a little bit more “ambitious” in his request, the system has to let him perform those requests and give him support on how to do so. For instance, a user may want to know about all the WH sites in the world that have castles and that is all; while another user may want to know about the WH sites that have something to deal with both mines and railways and that are situated in any northern Europe country.

So to conclude, there is just one type of system end-user with more or less the same characteristics and skills. The only difference is the level of ambition that a user has while making queries to the system.

3.2 Functional Requirements

3.2.1 Functionality requirements

R1. The system has to be web-based (based on client-server architecture).

R2. The system must be accessible by means of a web browser.

R3. The system domain is World Heritage, and particularly the information managed by the UNESCO's World Heritage Centre website.

R4. The system should be able to select relevant web pages to extract information matching a user's query.

R5. The system should be able to extract knowledge from html documents based on ontology.

R6. The system should be able to store the knowledge extracted (on the server side).

R7. The system should return the results of the search to the user.

3.2.2 User Interface requirements

R8. A prototype of the user interface with basic functionality should be provided.

R9. User interface has to follow same guidelines of style and presentation for all the pages of the web site.

R10. The user interface must allow a user to enter a request to the system.

R11. The user interface must present the output to the user.

R12. The user interface must provide the user with some kind of help of instructions if necessary.

3.2.3 Data requirements

R13. Documents being managed must be html pages.

R14. Documents being managed have to be written in English.

R15. Data managed will be within the system domain (selected documents and data collected).

R16. Information managed (extracted, stored and showed) will be at least the one considered critical in a first stage of the development of the system.

R17. There has to be an ontology that includes all the relevant concepts of the system domain

R18. A knowledge base stored on the server side should keep the information extracted.

3.3 Non-functional Requirements

3.3.1 Access requirements

R19. Admission to the system must be none restricted.

3.3.2 Usability requirements

R20. Ease of use. The system should be appropriate to the user's ability.

R21. Ease of learning, intuitive. No need to spend time learning how to use the system.

R22. System should try to achieve user's satisfaction (both in presentation and in results obtained). This requirement also deals with the two "levels of ambition" that a user can have. All users have to be satisfied with the system.

R23. System must be suitable for the task, allowing users to complete tasks efficiently without presenting unnecessary problems or obstacles.

R24. System should prevent the users from errors and allow error recovery.

3.3.3 Performance requirements

R25. The system has to be designed to be as automatic as possible.

R26. Response times for the extraction tasks have to be as less as possible.

R27. Response times for interaction with knowledge base have to be as less as possible.

3.3.4 Reliability requirements

R28. The process of selecting relevant documents to extract information from has to be the as accurate as possible.

R29. The extracted information has to be as accurate and reliable as possible (precision).

3.3.5 Extensibility requirements

R30. System must be designed in a way that can be able to hold future extensions (reasonably expected since the system is a prototype based on a research work) without having to be drastically redesigned.

R31. System must be implemented (codified) in a way that allows for future extensions and improvements.

3.3.6 Capacity requirements

R32. System must be able to handle the processing of simultaneous users requests.

R33. The initial capacity of storage of the system is zero. Users requests will populate the knowledge base gradually

R34. The knowledge base has to be prepared to store information of at least all the WH sites inscribed in the list

3.3.7 Scalability requirements

R35. The system must be able to increase its data capabilities to cope with new volumes, since it is probable that with the time more sites will be add to the WH List (or it can be considered in a future to store more information about each site, for instance to store pictures).

R36. The system architecture has to be prepared to grow as the needs grow (ensure scalability with a minimum of code changes by adding additional resources such as servers, processors or memory).

3.3.8 Portability requirements

R37. The system must be able to run at least under Internet Explorer and Netscape Navigator (which are the two most popular browsers).

R38. The system is expected to run under Windows and Linux platforms on the client side without having to adapt any software tool.

3.3.9 Flexibility requirements

R39. The system architecture should provide significant flexibility. To cope with decisions like for instance where to store the data, where to store the ontology, where to perform the processing and so on.

3.4 Summary of System Requirements

A summary table containing all the requirements of the system can be found in APPENDIX C.
The table is organized as follows: the first column tells the type of requirement (F for functional and NF for non-functional), the second identifies the requirement and the last one provides with a short description about it.

Part II

SYSTEM FOUNDATIONS

Chapter 4: Semantic Web & Ontology Survey

The concept of ontology was introduced very briefly in the former chapter. In this one it will be reinforced and covered in more detail. From ontologies main features and editors to the way they are populated. The notion on ontologies plays a central role in the emerging Semantic Web. So, before moving on to the ontology world, some background would be given about the "source" of all this new techniques: the so many times named Semantic Web.

4.1 Semantic Web

4.1.1 The Semantic Web idea

The problem of the huge amount of information, designed mainly for human consumption, can be solved by associating meaning with content, thus enabling computers to understand and process information. This idea, named the **Semantic Web**, was first introduced in 1996 by Tim Berners-Lee [1], the inventor of the World Wide Web. He describes it as follows:

> *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [2].*

From a high perspective, it can be said that the current Web is a provider of pages containing information and links between them designed for human consumption. The **Semantic Web** aims to increase the existing human-readable Web by adding machine-readable descriptions to Web pages and other Web resources. This is accomplished by giving the information an explicit well-defined meaning, also referred to as metadata, or data about data. Information is considered as a resource, where each resource can be

linked to any other resource uniquely identified by its URI[5]. An URL, a Web hyperlink, is the most commonly used URI.

The Semantic Web's ability to reference and identify resources is one of its foundations and makes it look like a great global mesh, or a big global database. Berners-Lee describes it as "Weaving the Web". Using this global mesh, software agents will be able to roam from page to page readily carrying out sophisticated tasks for the users [2]. The vision is to be able to make a query where an agent gathers information from the Web and provides the right and full information, even though it is presented at different pages or sources.

4.2 Introduction to ontologies

Ontologies have become more and more common on the WWW in recent years. Its development has moved from the realm of Artificial Intelligence to the desktops of domain experts.

4.2.1 Why to develop an ontology?

There are several reasons for developing ontologies [10]; some of them are for instance:

- *To share common understanding of the structure of information* - This is probably the more common reason to develop ontologies. This sharing of common vocabulary can be done not only among people but also among software agents.
  For example, if some web sites that have the same type of content (for instance DVDs, music CDs and books) were sharing a common underlying ontology of the terms they use, then computer agents could extract and aggregate information from those sites and use it to answer user queries, make price comparisons and so on.
- *To enable reuse of domain knowledge* – This was one of the main reasons of the recent increase in ontology research. A lot of domains need to represent common notions like for example the notion of time. An ontology about this can be developed in detail and then simply be reused in other domains. Another reuse case is when a general ontology is extended to describe a particular domain of interest o when an ontology (or more) is integrated in another one to describe a portion of a large domain.
- *To separate domain knowledge from the operational knowledge* – This is another common use of ontologies. For example, it can be possible to have an ontology of PC-components and then "feed" it to a separate program that configures PCs according to some specifications (made to order computers). Then it could also be possible to "feed the" same algorithm with another ontology, an escalator-components one for instance.
- *To analyze domain knowledge* – This is possible once a declarative specification of the terms is available.

[5] URI - Unified Resource Identifier

Sometimes to develop an ontology of a particular domain is not a objective in itself, it is more important to define the structure of data for other programs to use it.

4.2.2 Types of ontologies

There is not consensus when it comes the time of defining the types of ontologies. Some researches propose different kinds of ontologies taking several criteria into account, such the formality of the language or the level of dependence on a particular task or point of view. Guarino [17] for instance considers the last approach to classify ontologies and identifies the next basic types:

- *Top level ontologies* - describe general concepts like space, time, matter, object, event or action, concepts which do not depend of a particular problem or domain. However, the development of an enough general top level ontology has not been accomplished yet.
- *Domain ontologies* – describe the vocabulary related to a generic domain (like medicine or vehicles for instance) by means of specializing the terms introduced in the top-level ontology.
- *Task ontologies* - describe the vocabulary for a generic task or activity (such as selling) by means of specializing the terms introduced in the top-level ontology.
- *Application ontologies* - describe concepts which depend on both a particular domain and task. The concepts often respond to roles played by domain entities while performing certain task. This is the most specific type of ontology.

The following figure shows the different kinds of ontologies according to their level or generality (arrows represent specialization relationships).

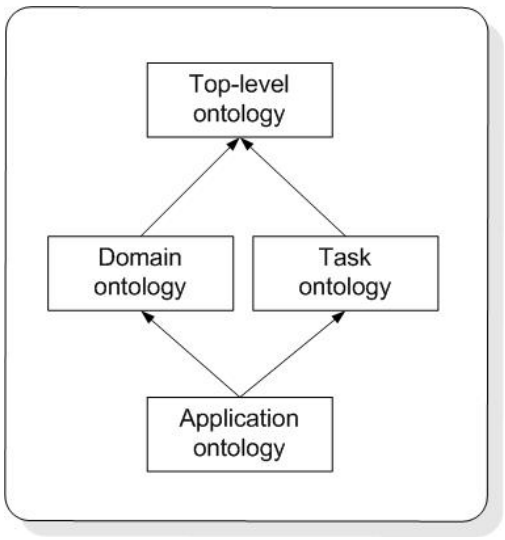


Figure 4.1 – Types of ontologies [17]

Dutra [18] on the other hand, identifies five kinds of ontologies. Note their similarity to the ones used in Guarino's classification.

- *Upper-Level* ontology, defines the base concepts upon which other ontologies are created.
- *Domain* ontology (also called *classic* ontology), defines the terminology and concepts relevant to a particular topic or area of interest.
- *Process* ontology, defines the inputs, outputs, constraints, relations, terms, and sequencing information relevant to a particular business process or set of processes.
- *Interface* ontology, defines the structure and content restrictions relevant for a particular interface *(e.*g., application programming interface (API), database, scripting language, etc.).
- *Role-based* ontology, defines terminology and concepts relevant for a particular end-user.

This classification could easily fit in Guarino's classification, which seems clearer and more general.

4.2.3 Ontology languages

Ontologies have languages for representation. RDF/RDFS is one of these languages. Ontology languages, like many other programming standards and languages, have evolved through the recent years. Among the early languages are Ontolingua and LOOM, to later ones like RDF/RDFS. The most frequently used at this time are probably DAML+OIL followed by RDF/RDFS. DAML+OIL is a joined force of DAML, an RDF Schema based language, and OIL for better performance [15].

OWL is a language derived from DAML+OIL and is still a working draft at W3C [I]. All these languages are based on the previous ones and extend their properties. Key features of OWL, compared to the previous languages, are that it adds more vocabulary for describing properties and classes. Examples of this are new relations between classes, like disjointness, cardinality, equality and richer typing properties. OWL is only supported by a very few number of the currently available ontology editors, because of still being a working draft. There are some free converters that can transform an ontology into OWL format.

Here is a table containing some brief information about the more relevant ontology languages nowadays.

	UML	RDF(s)	DAML+OIL	OWL
Description	Universal Modelling Language	Resource Description Framework	DARPA ML + Ontology Inference	Web Ontology Language
Years since propose	>5	>3	3 or less	3 or less

Open Source Support	√	√	√	Coming

Figure 4.2 – Ontology Languages summary

4.3 Ontology Population

Ontology population refers to the insertion of information into the Knowledge Base (KB) following the ontology domain representation. But, what it is exactly the so many times mentioned knowledge base? A knowledge base is not other thing that an ontology together with a set of individual instances of classes. In fact, there is a fine line between where the ontology ends and when the knowledge base begins.

To provide valuable ontology-based knowledge services, ontologies must be populate with a high quantity and quality of instantiations.

There are several approaches to ontology population: manual, semi-automatic and fully automatic. A fair amount of literature has been written about this field [6].

To populate an ontology manually is very laborious and time consuming. This option will not even be considered for this system.
There is also a bunch of semi-automatic approaches that have been studied in this field. These approaches create document annotations and store the results as ontology assertions. MnM [12] and S-CREAM [13] are some examples frameworks for user-driven ontology-based annotations, reinforced with Amilcare (an information extraction learning tool). However, these examples lack the capability to identify relationships reliably.

In the system discussed a fully automatic approach of feeding the ontology is aimed.

4.4 Ontology editors & Protégé

Ontology editors are tools that allow users to visually manipulate ontologies. The number of tools for building ontologies developed in the last years is high. In this section, some of the ontology editors which are most frequently used in the Semantic Web community are evaluated.

4.4.1 Ontology editors

Ontology editors make easy the development and management of ontologies, the definition and modification of concepts, properties, axioms and restrictions. Some of the editors also provided the capabilities of querying and browsing of ontologies. This section describes some of the most relevant ontology editors.

Several of the ontology editors available today, especially among the freeware, are developed by an university. Examples of these are Protégé by Stanford University [C] or OilEdit by

University of Manchester. These universities are among the groups that are very active in this field. This is reflected in the numerous papers and projects performed, and in the updating rate of the editors and different plugins they develop.

Ontolingua[6] is an ontology development environment for browsing, creating, editing modifying and using ontologies. It contains ontology authoring tools by assembling and extending ontologies extracted from a library of reusable ontologies. It combines axioms, definitions and non-logical symbols termed "*words*" from multiple ontologies.

OilEd[7] is a freely available editor for OIL and DAML-OIL developed by the University of Manchester. It is an ontology editor with a frame interface. It uses reasoning to support ontology design and maintenance. Its main advantage is the extension of a frame editor paradigm to deal with an expressive language and the incorporation of a descriptive logic reasoning engine which is used to check class consistency and inference relationships. It enables class definition, slots and axioms. In contrast to other frame systems, it facilitates developers the use of arbitrary boolean combinations of frames or classes connected by *and*, *or* and *not*.

Chimaera[8] is a merging and diagnostic web-based browser ontology environment. It accepts over 15 designated input formats such as ANSI Knowledge Interchange Format (KIF), Ontolingua, Protégé, CLASSIC, iXOL as well as any other OKBC-compliant form (OKBC stands for Open Knowledge Base Connectivity). **Chimaera** has an analysis capability to make incompleteness tests, syntactic or semantic checks as well as taxonomic analysis. Some of the main tasks supported by **Chimaera** are: loading knowledge bases in different formats, reorganizing taxonomies, resolving name conflicts, browsing ontologies and editing terms.

The most popular ontology editor is probably Protégé from Stanford University. This editor will be covered separately in the next section.

There are many other ontology editors that could have been mentioned in this section. For a complete comparison among then please refer to the summary table that appears at http://www.xml.com/2002/11/06/Ontology_Editor_Survey.html and it is part of [19]

4.4.2 Protégé

From all the tools available for modelling knowledge, Protégé-2000 from the Medical Informatics group at Stanford University is arguably the most successful open source knowledge-modelling platform. Although Protégé was originally developed 15 years ago to support knowledge acquisition for medical experts systems, it has also become very popular for many other purposes [20].

[6] Ontolingua is available at: http://www.ksl.stanford.edu/software/ontolingua/

[7] OilED is available at: http://oiled.man.ac.uk/

[8] Chimaera is available at: http://www.ksl.Stanford.EDU/software/chimaera.

Protégé is a Java tool, is easy to install and use and its web site [C] provides with a good documentation. In addition an extremely active discussion list is available as well as frequently updates and new builds.

Protégé can be used to perform the following tasks: class modelling, instance editing, model processing and model exchange. Protégé can edit ontologies and saved them in several formats. It can also see its functionality extended by means of many plugins available; this mechanism ensures the adaptation of the system to specific needs.

In Protégé, as in many other knowledge-modelling tools, classes represent concepts of the domain, which can have attributes and relationships. Classes are arranged in an inheritance hierarchy and displayed in a tree format. Protégé supports multiple inheritance and classes can be abstract or concrete. Like in Java and other object-oriented tools, only concrete classes can be instantiated.

In Protégé classes' attributes and their relations are called *slots*. Apart from primitive values and enumerations, slots can also refer to other classes and instances of the model (therefore allowing building relationships and associations between instances). Slots store either single or multiple values and can be attached to multiple classes; the latest is one of the main differences between a slot and a conventional object-oriented attribute. Another difference is that it is possible to specify constraints (similar to cardinalities) on slots values. Protégé also allows defining default values for slots and inverse slots.

One of the best things of Protégé is that it is very helpful when it comes the time of entering instances, because it automatically generates graphical forms from the class definition. A change to a class automatically rearranges its forms. Therefore, this ontology editor is excellent for rapid prototyping.

But Protégé's main advantage among other ontology editors is, as mentioned before, the possibility of extending its functionality with plugins. It supports three kinds of plugins: *storage backend plugins*, *slot widgets* and *tabs*.

Storage plugins are non-visual modules that save and load ontologies in certain file or database format. The currently supported storage formats are [20]: **CLIPS** (C Language Integrated Production System), **XML**, **XML Schema**, **RDF**, **OIL** (Ontology Inference Layer), **DAML+OIL** (DARPA Agent Markup Language + OIL), **UML**, **XMI** (XML Metadata Interchange) and the recently supported **OWL** (Web Ontology Language). The compatibility of formats allows reusing ontologies.
Slot widgets are graphical components placed in Protégé's instance forms to view and edit slot values. There are plenty of them in the Protégé's plugin library.
Tab plugins are GUI panels displayed as a tab in the main window of the Protégé environment. This editor has several tabs by default: Classes tab, Slots tab, Forms tab and Instances tab. Additional tabs can be enable.

For further information about Protégé plugins please refer to the Protégé Web Site [C].

Chapter 5: Information Retrieval and Extraction Survey

This chapter is devoted to the Information Retrieval (IR) and Information Extraction (IE) techniques since both are used in the system being developed.

The organization of this chapter is as follows: main features of Information Retrieval an Information Extraction are described, then a summary of the issues involved in using IR as a filter for the input to an IE system. Finally, after the survey done, the system approach concerning these techniques is introduced.
But before starting to dig into these matters the difference between these two terms often mixed up will be clarified.

5.1 Overview

These two techniques are sometimes mixed up or grouped together as if they were the same one. Far from that, actually they are complementary and very suitable to work together in order to transfer information from the WWW to the user and to potentially create powerful tools for text processing.

As an introductory summary clarify that IR retrieves relevant documents from collections while IE extracts relevant information from documents. In other words, by using IR techniques one gets relevant documents to analyze and by using IE techniques one gets fact out of the documents and analyses those facts.

Information Retrieval recovers from a collection a subset of documents which are (hopefully) relevant to a query, based on keyword searching. Information Extraction is different; its aim is to extract from the documents salient facts about pre-specified types of events, entities or relationships. These facts are then usually entered automatically into a database for its subsequent use (to be analyzed, to give a natural language summary, or simply to serve for on-line access).

A visual comparison between both techniques is shown next.

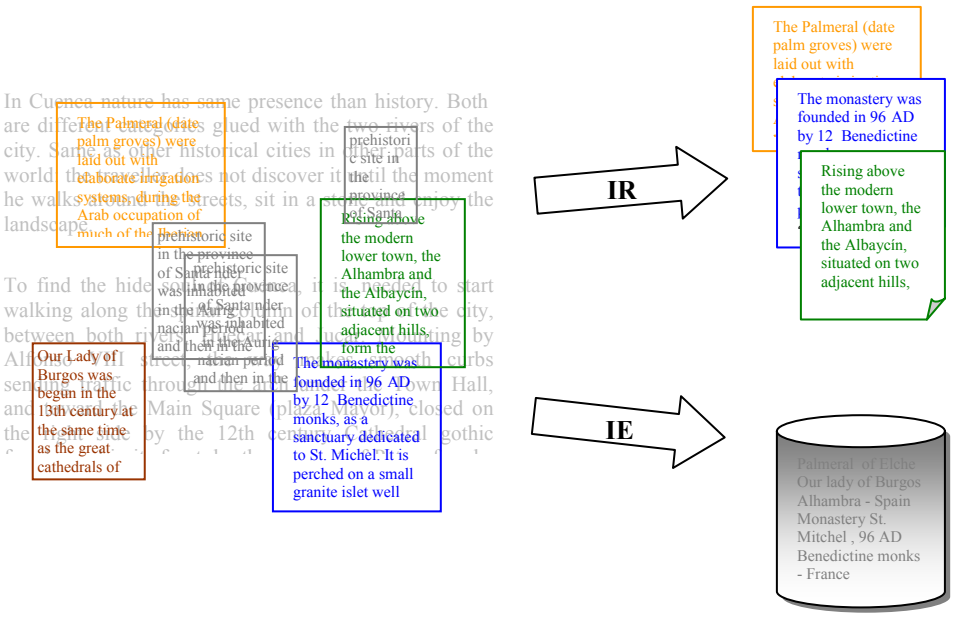


Figure 5.1 – Information Retrieval vs. Information Extraction

5.2 Information Retrieval

5.2.1 IR Systems

Information Retrieval systems deal with the representation, organization and access of information items. IR identifies documents which match a query as presented to the system and which may, or may not, contain the desired information.
There are two main approaches in IR, *Boolean* and *ranked-output* or best match. Giving more detail about these approaches is out of the scope of this work; please refer to specialised literature on this field.

A typical IR system illustrated by means of a black box will look like this:

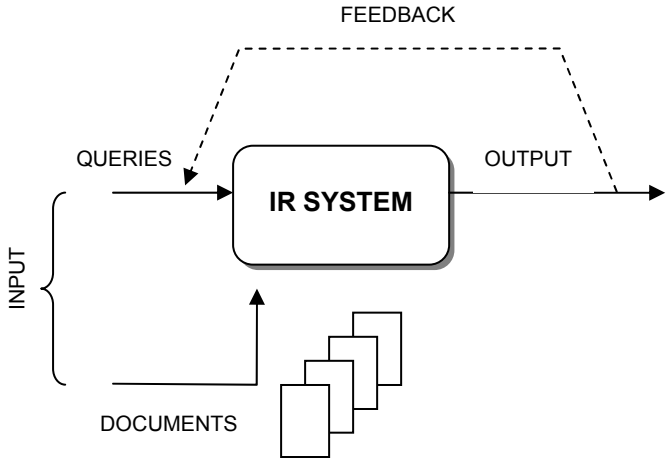


Figure 5.2 – A typical IR system

5.2.2 Search Engines

In order to select documents from the World Wide Web one of the many search engines available nowadays can be used. There are a lot of these tools in the market and each of them has its own features. A good place to find detailed information about them is at http://www.searchenginetrends.com/overview.php. In this web site one can find a complete ranking of search engines and plenty of statistical information.

A comparison of search engines was done. As a result some summarised information can be seen in the following figure. More search engines than the ones shown in the table were surveyed, particularly those that offer a free API. However, due to a lack of space only the most important ones can be shown here. For further information please refer to the web site http://www.searchenginetrends.com.

	Excite	Google	Yahoo
Parent Site	www.excite.com	www.google.com	www.yahoo.com
Description	Still enjoys a high degree of popularity among internet surfers. Fast servers and a large audience make it a hard publisher for advertisers to pass up	It has taken the world of search engines by storm in a very short time. Not only supports free submissions and allow public use of its api, Google has made a sincere commitment to exploring and expanding the role of search engine technology. If ever a search engine deserved to be labeled "state of the art", Google would be it.	One of the most popular internet sites, now it is making major advances in the search engine market with its recent aquisition of Inktomi. As of this writing free submission into Yahoo's directory is still available.
Search Traffic	401.45	114702.00	17475.00
Free API	none	http://www.google.com/apis/	none
Pay API	none	http://www.google.com/appliance/	none

Figure 5.3 – Search Engines comparison

As a conclusion to this brief survey, to state that probably the best search engine that is currently available is Google.

5.2.3 IR as input to IE process

The idea of combining IR and IE is not new.
IE systems are usually very intensive computationally talking; they have to support a big burden and carry out a lot of processing. Therefore, it is necessary to make sure - as far as possible - that documents becoming the input to these systems are likely to be within the domain of the specific system. Here is where the help of an IR system can be decisive.
So in this context an IR system can be used as a front-end to an IE system performing the task of retrieving from the source collection only those documents that are relevant to its extraction scenario.

The linkage of these two techniques at a conceptual level is pictured in the figure below. A user submits a query via the IR system; this system retrieves from the original set of documents a reduced subset of relevant ones that are then passed to the IE system which extracts information from them.

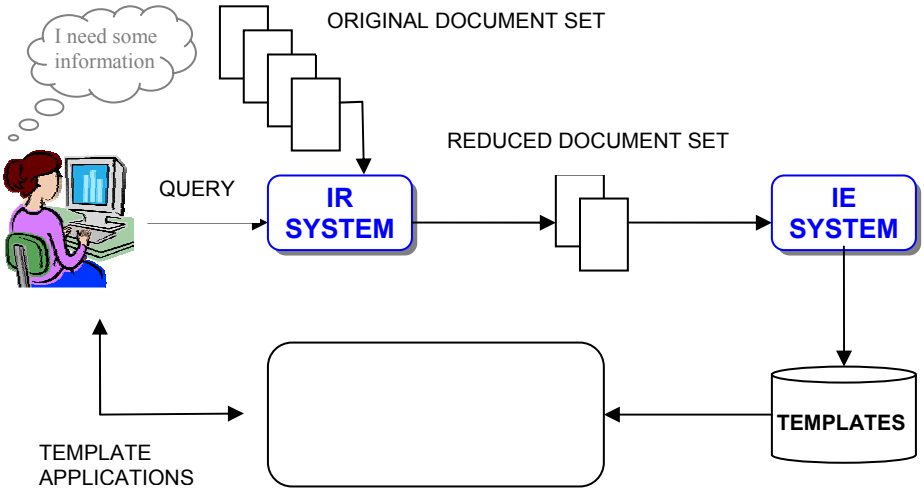


Figure 5.4 – Architecture of a coupled IR-IE system

5.3 Information Extraction

Basically, Information Extraction is a method of filtering information from large volumes of text. Gathering information involves retrieval of documents from relevant sources and the tagging of certain terms in the text. The output that one would want from an IE tool is a structured representation, such as a database of selected types of information from the texts.

In this section some of the basic techniques and tools of Information Extraction will be presented. It is not intended as a fixed or exhaustive classification but just as a basic easy-to-understand one. For more information about this wide field refer to specific literature. The same can be applied to the tools classification that follows the techniques classification.

5.3.1 Why the interest in IE?

Usually, Web data is retrieved by browsing and keyword searching, which are intuitive forms of accessing data on the Internet. However, these search strategies present some limitations.
Browsing, for example, is not very suitable for finding particular items of data because following the links is tedious and can easily lead to get lost.
Keyword searching is usually more efficient than browsing. On the other hand, it often returns vast amounts of data that the user can hardly handle, so a more structured output is needed when searching for information.

During the last decade Information Extraction has become increasingly more interesting, mainly due to the explosive growth of the Web. Vast amount of information is available on the Web today but most of it only exists in natural language form as for example in newspaper articles and such. Extraction of information from such natural language sources into traditional database form could provide easier access to information that is already present on the Web.

5.3.2 IE Techniques

Information Extraction relies on several different techniques which will be mentioned in the following.
It is important to note that one IE tool would normally be tailored to perform well in one specific domain. A domain in this sense could be for instance a collection of newspaper articles, police reports, medical reports, a branch of scientific journals or fiscal reports. In each of these domains information is structured in different ways, so the same IE tool would perform differently in these domains. An IE tool needs of course to be configured to the type of domain it is supposed to operate in.

Below is a brief description of the main techniques:

Pattern matching

In many Information Extraction systems most of the text analysis is performed by matching the text against regular expressions. If a text segment matches one of these regular expressions the particular text segment is given a label, this might be a “name”, “time”, “place” or similar label.

Syntactic structure

The identification of the complete syntactic structure of a sentence is a difficult task, but the identification of some syntactic structure can simplify the information extraction phase itself. Often the arguments to be extracted are noun phrases and corresponding relations, so it is very important to be able to identify noun groups in the text. Also verb groups should be identified as these contain information on tense (past, present or future).

Name recognition (NE)

Names appear very often in natural language texts and identifying and classifying them as person names or place names are important as argument values for many extraction tasks.
Names can be identified in several ways; either by using large dictionaries or by using a set of patterns common for names. Such patterns could be capitalization of the first letter in a name, a preceding title such as “Mr.”, and often companies can be recognized by their final token such as “Inc.” or similar. It is also important to be able to identify aliases, i.e. to identify IBM as Industrial Business Machines.
Name recognition systems that work at nearly the same level as manual name recognition are available today.

Applying an ontology

Before start explaining this technique an important concept has to be defined for a better understanding. A minimalist introduction to ontologies is given below. More details about ontologies will be given in further sections (see chapter 4).

What is an ontology?

The word "ontology" comes from the world of philosophy. It has also been used for a long time within the artificial intelligence and knowledge representation community.

In the context of knowledge sharing a short answer for this question could be that an ontology is a specification of a conceptualization. That is, an ontology is a description of the concepts and relationships that can exist for an agent or a community of agents. This definition is certainly a different sense of the word than its use in philosophy.

Apart from other uses, ontologies help to guide and detail what kind of knowledge to harvest from unstructured text on the Web. They use concepts and relations for classifying domain knowledge. Those are basic elements in an ontology.

Ontology-based technique

To use this technique an ontology has to be previously constructed to describe the data of interest, including relationships, lexical appearance and context keywords. Tools that use this technique parse the ontology to automatically produce a database by recognizing and extracting data from web pages given as input. Before applying the ontology it is necessary to automatically extract chunks of text containing data “items” of interest.

NLP-based Tools

A couple of existing Natural Language Processing tools for Information Extraction will be mentioned here.

Robust Automated Production of Information Extraction Rules (RAPIER) is such a tool that extracts information from free text. The input for the tool is the document from which to extract information and a filled template that tells the tool which data to extract. From this template RAPIER “learns” data extraction patterns to be followed during information extraction. This is a “single-slot” tool as it generates one record per document.

WHISK is another tool for extraction of information from natural language text. WHISK starts out with an empty set of extraction rules and the user induces the relevant extraction rules through a series of training documents, where the user tags all information to be extracted. From this tagging WHISK creates a set of extraction rules to be used. This is a “multi-slot” tool as it creates several records from one document.

5.3.3 Evaluation metrics for Information Extraction

A lot of the research about IE in the last decade has been connected with the MUC (Message Understanding Conferences) competitions. These competitions were sponsored by the Defense Advanced Research Projects Agency (DARPA) from 1991 to 1998. They consisted in competitions where the participants compared their results with each other and against human annotators' key templates. By doing this, a lot of IE systems and methods for formal evaluation of IE systems were developed (some of them are still in use by the US government).

So it is not surprising that the MUC evaluation metrics of precision and recall are still tend to be used with slight variations. These metrics have a very long tradition in the field of IR [22].

"***Precision*** *measures the number of correctly identified items as a percentage of the number of items identified* [22]". In other words, it measures how many of the items that the system could identify were actually correct, regardless of whether it also failed to retrieve correct items. The higher the precision, the better the system is at ensuring that what is identified is correct.
There is another metric called **Error rate** that is the inverse of precision, and measures the number of incorrectly identified items as a percentage of the items identified. Sometimes it is used as an alternative to precision.

"*Recall* *measures the number of correctly identified items as a percentage of the total number of correct items* [22]". In other words, it measures how many of the items that should have been identified were actually identified, regardless of the number of false identifications made. The higher the recall, the better the system is at not missing correct items.

Obviously, there must be a balance between these two rates because a system can easily be made to achieve 100% precision by identifying nothing (and so making no mistakes in what it identifies), or 100% recall by identifying everything (and so not missing anything).

Chapter 6: Annotations Survey

Please note that the annotations survey presented in this chapter will be carried out having GATE software tool as a basis. This survey is part of the research phase done along the thesis and has been included in this document to help the reader to acquire a better comprehension of the tools being involved in the system development.

6.1 ANNIE System

ANNIE (standing for A Nearly-New Information Extraction) is an Information Extraction system that comes with GATE (developed by Hamish Cunningham, Valentin Tablan, Diana Maynard, Kalina Botcheva, Marin Dimitrov and others).
Why is that GATE comes with its own IE system? The answer is immediate: it was originally developed in the context of Information Extraction and has been used for building all sorts of IE systems since. ANNIE is currently in active use in a number of industrial and academic projects.

Basically ANNIE works like an assembly line in a factory, it takes documents as inputs (which can be of any of these types: XML, HTML, email, plain text, etc.) and produces outputs (Named Entities for instance). The whole task is performed in several steps, following a given sequence. One step's output is the next step's input.
ANNIE is based in finite state algorithms and the JAPE language (see sections below for more information about JAPE).

It can be used both in the development environment (GUI) loading and running its components or it can be embedded in other software.

6.1.1 ANNIE modules

ANNIE consists of several resources which have to be called in the right order forming a pipeline that is showed in the following figure (ANNIE components are marked in red).

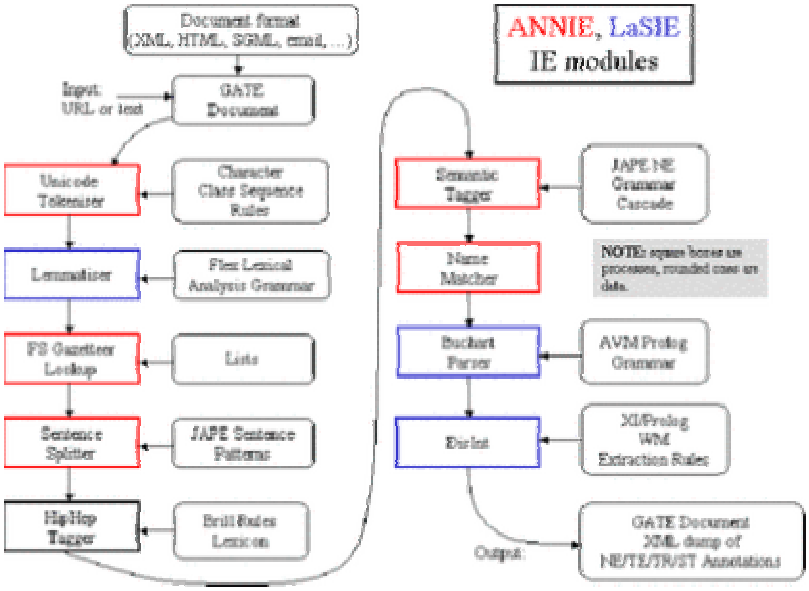


Figure 6.1 – ANNIE and LaSIE [source [D]]

When ANNIE system is loaded in GATE's environment, the following modules are automatically loaded too: ANNIE English Tokeniser, ANNIE Gazetteer, ANNIE Sentence Splitter, ANNIE POS Tagger, ANNIE NE Transducer and ANNIE OrtoMatcher. Any other module that needs to be used can be separately loaded as a Processing Resource.

Some ANNIE components (or modules) use finite-state techniques to implement various tasks. A brief description of each will be offer below. Those modules that will be used for the system purpose will be explained in more detail.

GATE Unicode Tokeniser

This component is a very important one because a lot of the subsequent processing work will relay on its results. The tokeniser is in charge of splitting the document into very simple tokens such as numbers, punctuation and words of different types.
Its work is limited in order to place the burden on the grammar rules and so enable more flexibility. This means that the tokeniser does not need to be modified for different text types.

The Unicode Tokeniser is the default tokeniser and only produces annotations of type **Token** and **SpaceToken**. Token annotation can be of different kinds like Word, Number, Symbol and Punctuation. On the other hand, SpaceToken annotations can be only of two types: space and control.

To clarify this see a manipulated screenshot of GATE GUI in Figure 6.2.

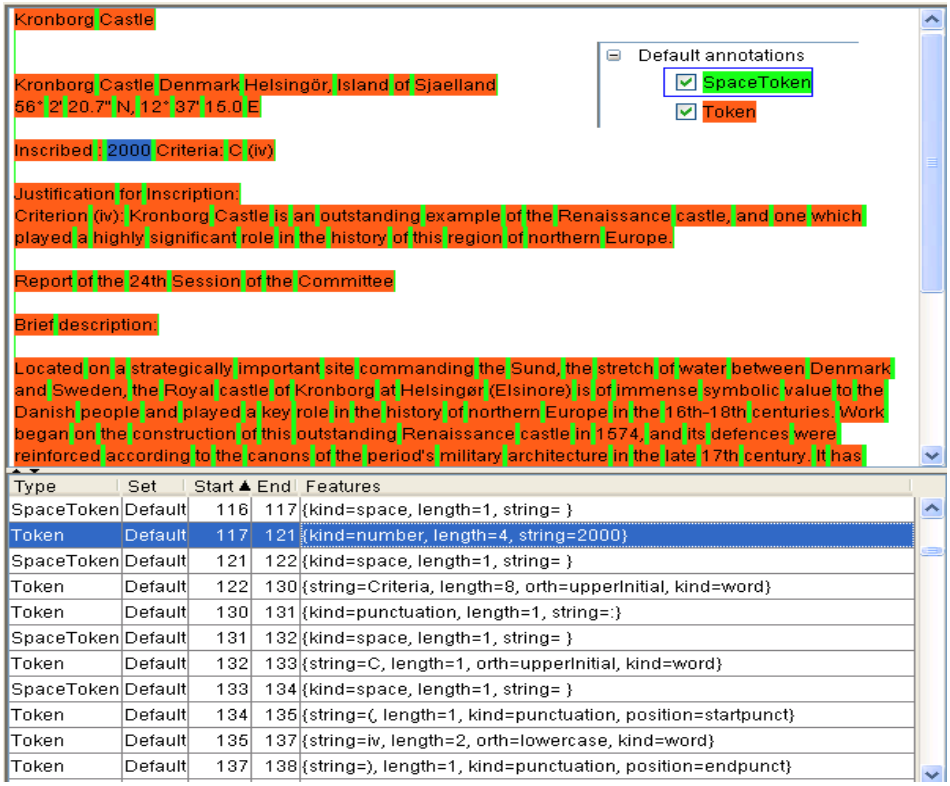


Figure 6.2 – Unicode Tokeniser results (GATE screenshot)

The use of the Unicode Specification for input symbols (UTF-8 encoding) makes the same tokeniser able to process text in virtually any language, giving more generality this way.

As it was said before, the Unicode Tokeniser is the default one but other ones can be created if necessary. For instance the ANNIE English Tokeniser comprises the normal one and a JAPE transducer (see sections below). This tokeniser should always be used on English texts if it is necessary to perform POS Tagger afterwards.

Gazetteer

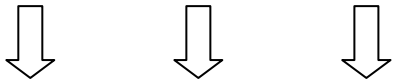
A gazetteer list is not more than a plain text file containing one entry per line. Each gazetteer represents a set of things with a common semantic. For example, names of countries, days of the week, colours, names of organizations and so on.

There has to be an index file (called **lists.def**, for an example see figure 6.3) that comprises all those lists. This file is used to access the gazetteers and has to keep a certain format. For each gazetteer three things can be specify (at least the first two of them have to be): the list file name, the mayor type and the minor type which is optional. All the gazetteers have to be stored in the same folder as the index file.

The gazetteer does not need or depend of any other processing resource since it runs directly over the text being processed. It also handles Unicode input that makes it usable for text in any language.

This component is one of the ones using finite-state techniques. These lists are compiled into finite state machines. When a token is matched by these machines it will be annotated with features specifying the mayor type and minor type. Then grammar rules specify the types to be identified in a particular circumstance.
An example will make things more clear. So, imaging we have a line in the index file like the following one:

```
country.lst:location:country
```



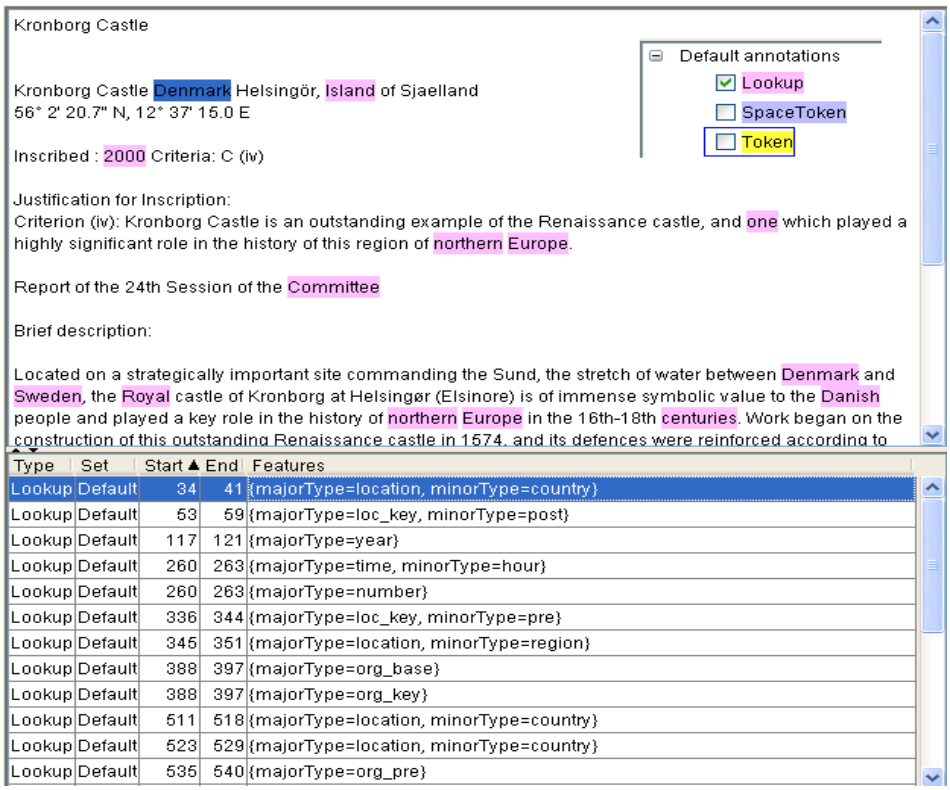
list name **mayor type** **minor type**

If, for example, a specific country has to be identified then the minor type "country" should be specified in the grammar rule in order to match only information about countries. But if, for instance, any location has to be identified (no matter what kind of location is) then the mayor type "location" should be the one specified and so produce annotations that comprises things like countries, mountains, provinces, regions and so on all gathered under the generic type location.

```
abbreviations.lst:stop
charities.lst:organization
city.lst:location:city
company.lst:organization:company
company_cap.lst:organization:company
country.lst:location:country
currency_prefix.lst:currency_unit:pre_amount
currency_unit.lst:currency_unit:post_amount
date_key.lst:date_key
date_unit.lst:date_unit
day.lst:date:day
festival.lst:date:festival
govern_key.lst:govern_key
government.lst:organization:government
hour.lst:time:hour
jobtitles.lst:jobtitle
months.lst:date:month
mountain.lst:location:r
```

Figure 6.3 – Example of a gazetteers index file

Gazetteer produces **Lookup** annotations that are part of the default annotations set. Like with the former component it is presented here a screenshot showing the results of running a pipeline with Unicode Tokeniser + Gazetteer. See Figure 6.4 for a visual explanation about this component operation mode.



Type	Set	Start ▲	End	Features
Lookup	Default	34	41	{majorType=location, minorType=country}
Lookup	Default	53	59	{majorType=loc_key, minorType=post}
Lookup	Default	117	121	{majorType=year}
Lookup	Default	260	263	{majorType=time, minorType=hour}
Lookup	Default	260	263	{majorType=number}
Lookup	Default	336	344	{majorType=loc_key, minorType=pre}
Lookup	Default	345	351	{majorType=location, minorType=region}
Lookup	Default	388	397	{majorType=org_base}
Lookup	Default	388	397	{majorType=org_key}
Lookup	Default	511	518	{majorType=location, minorType=country}
Lookup	Default	523	529	{majorType=location, minorType=country}
Lookup	Default	535	540	{majorType=org_pre}

Figure 6.4 – Gazetteer results (GATE screenshot)

Sentence Splitter

This component is also very important, especially if a lot of processing is going to be done over the text. It is required for the tagger module too. It provides the document being processed with annotations that can be of a great use in the construction of jape rules.

The task of the sentence splitter, as it is implicit in its name, is to split the text into sentences. It provides two kinds of annotations: Sentence and Split.
Each sentence is annotated with the type **Sentence**. Each sentence break is given a **Split** annotation. This last one has four possible situations: a full stop ".", a line break "CR", any kind of punctuation mark "punctuation" or a series of punctuation marks "multi".

Once again a screenshot will be used for a better understanding of this component task. In Figure 6.5 is shown the result of running Unicode Tokeniser + Gazetteer + Sentence Splitter over the - so far well known - example test about Kronborg Castle.

Taking a look, in the figure, at the sentence that is highlighted one can appreciate how the splitter makes differences between those Split annotations that are just a full stop (kind = internal) and those others that are a break line or a series of break lines (kind = external).

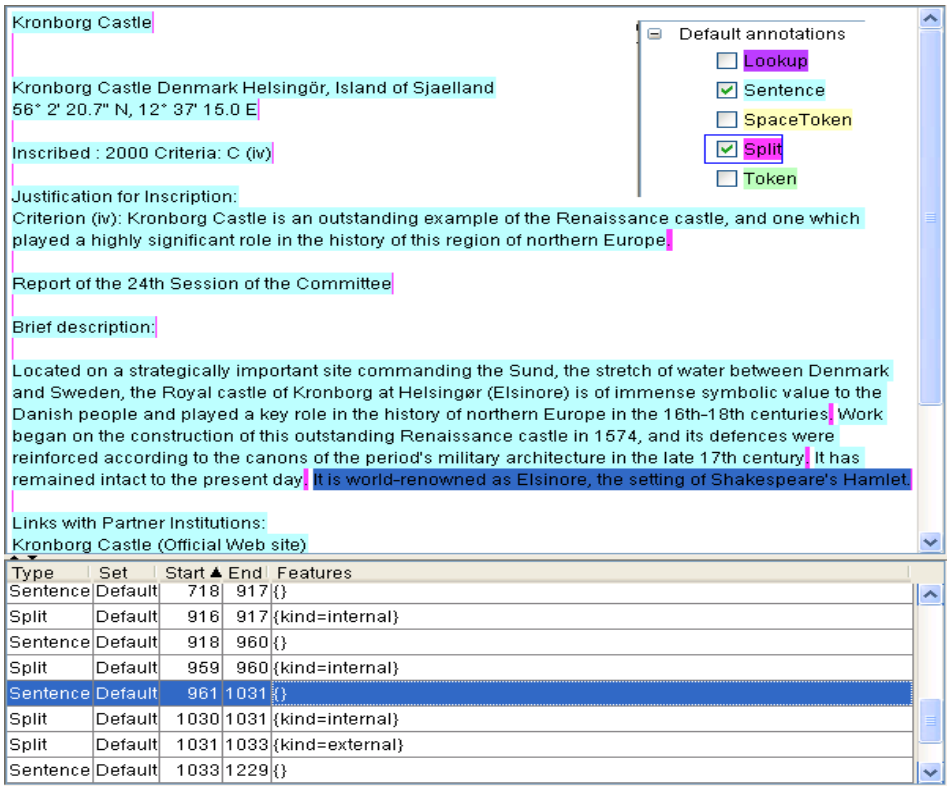


Figure 6.5 – Sentence Splitter results (GATE screenshot)

Part of Speech Tagger

This tagger is a modified version of another tagger called the Brill tagger. The task that performs this module is to assign a part-of-speech tag as an annotation to each word or symbol.
It uses a default lexicon and rule set that was the result of training on a large corpus taken from the Wall Street journal. To modify its behaviour POS tagger has to be re-trained on relevant annotated texts. Two additional lexicons also exist and can be used to replace the default one at load time. One of them is for text in all uppercase and the other one in all lowercase.

Semantic Tagger

It is based on the JAPE language and contains rules that transform the annotations assign in early stages into other output annotations for the entities.
Tagger modules are not going to be used for the system purpose.

6.1.2 Technicalities about ANNIE

The existing ANNIE resources are a little bit hidden in GATE. To get access to them one has to do the following:

- Locate the file gate.jar in the ‘bin’ directory of the installation
- Copy it somewhere else and rename it to gate.zip
- Unpacked the zip file and locate the existing resources there

The physical location of the PR is given below:

Resource	Location
Gazetteers	/gate/resources/creole/gazetteer/default
ANNIE jape grammar	/gate/resources/creole/transducer/NE
Sentence Splitter	/gate/resources/creole/splitter
POS Tagger	/gate/resources/creole/heptag

If for any reason (to fit one's purposes) one need to modify some of the default ANNIE resources one have to do this:

- Make a copy with a new name
- Edit the file with our modifications
- Load the new resources into GATE as a new Processing Resource

6.2 JAPE Language

JAPE (standing for Java Annotations Pattern Engine) is a language developed by GATE's working team at Sheffield University. This language is used to recognize regular expressions in annotations on documents and also to create more annotations. It is a version of CPSL – Common Pattern Specification Language9 developed in the TIPSTER programme. JAPE is quite easy to understand and especially flexible.

JAPE is used, amongst other tools, in GATE's built-in information extraction mechanism. So, in that way, it could be said that JAPE is a processor for IE.

JAPE grammars will be covered in more detail in the subsections below. Afterwards it will be applied to create a customized grammar for the system domain.

NOTE: For readers unfamiliar with regular expressions, pattern matching and the like concepts mentioned in this section see a brief guide in the appendices.

6.2.1 JAPE grammars

A JAPE grammar consists of a set of phases, each of which consists of a series of hand-crafted rules that describe patterns to match and annotations to be created. These rules can be adapted and customized to deal with different kinds of text and are compiled into finite state machines.

In the real world there is a main file containing the JAPE grammar set. This file contains a list of the grammars to be used, in the correct processing order. The ordering of the grammars is crucial, because they are processed in series, and later grammars may depend on annotations produced by earlier grammars.

[9] The original version of this language was made by Doug Appelt (www.ai.sri.com/~appelt). GATE's JAPE was first implemented using TextPro system (www.ai.sri.com/~appelt/TextPro) .

For instance, a grammar that recognises the preliminary part of an url address (an example with this case will be analysed later) has to be processed before the one that recognises the main part of an url.

This could be, for instance, the format of a file containing a grammar:

```
MultiPhase:    TestMyGrammars
Phases:
  first
  firstname
  name
  name_post
  date_pre
  date
  reldate
  number
  address
  url_pre
  url
  email
  final
  unknown
  name_context
  loc_context
  clean
```

Figure 6.6 – JAPE grammar main file example

Apart from this file there are several other files, each containing a phase with its corresponding rules. It always has to be a first phase that initialises the whole grammar set and a last one that cleans up temporary annotation created along the way.
Input annotations must be defined at the start of each grammar as well and if they are not then the default ones will be used (Token, SpaceToken, Lookup).

The following figure gives part of the BNF (Backus-Naur Format) description of the grammar. For a whole formal description and a more detailed explanation of JAPE grammars please refer to GATE's User Guide [22].

```
Rule ::=
    <rule> <ident> ( <priority> <integer> )?
   LeftHandSide "-->" RightHandSide

LeftHandSide ::=                          RightHandSide ::=
    ConstraintGroup                           Action ( <comma> Action )*

ConstraintGroup ::=
    ( PatternElement )+ ( <bar> ( PatternElement )+ )*

PatternElement ::=
    ( <ident> | BasicPatternElement | ComplexPatternElement )
```

```
BasicPatternElement ::=
    ( ( <leftBrace> Constraint ( <comma> Constraint )* <rightBrace> )
    | ( <string> ) )

ComplexPatternElement ::=
    <leftBracket> ConstraintGroup <rightBracket>
    ( <kleeneOp> )? ( <colon> ( <ident> | <integer> ) )?

Action ::=
    ( NamedJavaBlock | AnonymousJavaBlock | AssignmentExpression | <ident> )
                                    …
```

Figure 6.7 – BNF of JAPE's grammar

6.2.2 JAPE rules

Each rule in a JAPE grammar has and LHS (left-hand-side) and a RHS (right-hand-side) part meaning the pair pattern/action. The way the information about the annotation is transferred from the LHS of the rule to the RHS is by means of a label attached to pattern elements (denoted by a preceding semi-colon).

The LHS of the rule consist of an annotation pattern that can contain regular expression operators as well as macros. Macros are used to avoid repeating information and can be use inside other macros.
On the other hand, RHS consists of statements to manipulate annotations and can contain Java code and macros too. As it can be deduced, it doesn't really make much sense to use Java code in LHS since this is just for pattern matching.

All this terminology might result a little bit confused up to this point, so the best way to clarify the JAPE rule's format is by giving an example and analysing it.

```
Phase:  UrlPre
Input:  Token SpaceToken
Options: control = appelt

// http://www.amazon.com
// ftp://amazon.com
// www.amazon.com

Rule: Urlpre

(           (({Token.string == "http"}|
              {Token.string == "ftp"})
             {Token.string == ":"}
             {Token.string == "/"}
        {Token.string == "/"}
       )    |
            ({Token.string == "www"}
        {Token.string == "."}
       )
):urlpre
-->
:urlpre.UrlPre = {rule = "UrlPre"}
```

Figure 6.8 – Example of a JAPE rule

This is the one of the phases of a JAPE grammar called UrlPre and it contains only a rule called Urlpre (notice the different spelling, case sensitive). The grammar input annotations are Token and SpaceToken and the control option defining the method of rule matching is applet (more about this topic will be seen later in this section).
The rule is created to identify the three possible types of preliminary url address: “http://”, “ftp://” or “www.” .
Everything that is between round brackets (is a complex constraint specification) and before the symbol “-->” is the LHS part. Then it is found the label “:urlpre” and just after the arrow it is the RHS part. It means that this rule binds the set of annotations representing types of url prefixes to a new annotation.

6.2.3 Technicalities

The different grammars that build up any JAPE grammar are not more than plain text format files. So actually, they can be edited in any text editor such as Notepad or WordPad.

The grammars do not need to be compiled because they are automatically analyzed and executed by the JAPE Transducer module in GATE GUI. This module is a finite state transducer over the annotations in the document.
JAPE is very good for a quick prototyping once the basic concepts are learnt.

6.3 Conclusions

This brief study about two of the main features provided by GATE does not try to be exhaustive or become a user manual for annotate in GATE. For further information please refer to the User Guide [22].

Part III

SYSTEM DESIGN & IMPLEMENTATION

Chapter 7: System Design

This chapter discusses the design of the system. First a general overview of the system architecture is given and explained following two different points of view: horizontal and vertical. Later on, along several sections the design of each component that makes up the system will be explained in further detail.

7.1 System Architecture

The system is based around a three-tier client/server architecture. This kind of software architecture, which has evolved from the two-tier architecture to overcome its limitations, has become very popular.
This architecture partitions the system into three logical tiers or layers: the Presentation tier, the Application server tier and the Data storage tier. They will be explained in detail in the following sections, pointing out some of the design decisions taken. But before that a small introduction to 3-Tier architecture is given.

7.1.1 Introduction: 3-Tier Architecture

The three-tier architecture model is the emerging standard for scalable web applications. This architecture ensures both horizontal and vertical application scalability, thus fulfilling the scalability requirements defined in early stages of the project (for further details refer to section 3.3.7).

In general, a three-tier architecture is any system which enforces a general separation between the following three parts:

1. **Client Tier** or user interface
2. **Middle Tier** or business logic
3. **Data Storage Tier**

Applied to web applications (as is the case of the system being considered) and distributed programming, the logical tiers usually correspond to the physical separation between three types of devices or hosts:

1. Browser or GUI Application
2. Web Server or Application Server
3. Data Server

There are two ways of looking at this architecture, but at the end it is all the same. From a hardware (physical) point of view the 3-Tier architecture consists of the three device layers mentioned before, as shown next.

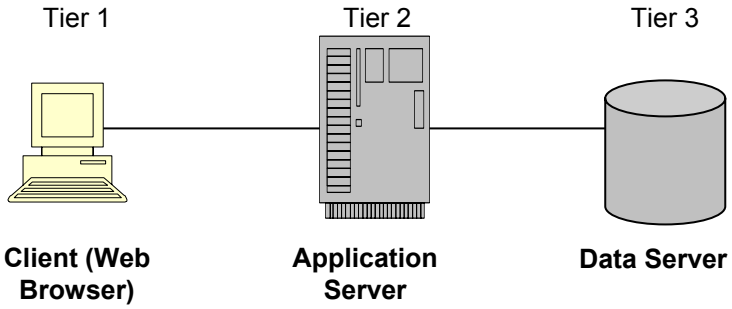


Figure 7.1 – 3-Tier Architecture (hardware view)

The application and data servers as well as the user interaction elements can be easily split across multiple servers and each of these servers can in turn be expanded (by adding either more resources or adding new servers).

It is important to note that boundaries between tiers are only logical. It is also possible to run the three tiers on the same (physical) machine. What matters more is that with this architecture a system is neatly structured, and that boundaries between the different layers are well defined.

Under a software (logical) point of view the architecture would look like:

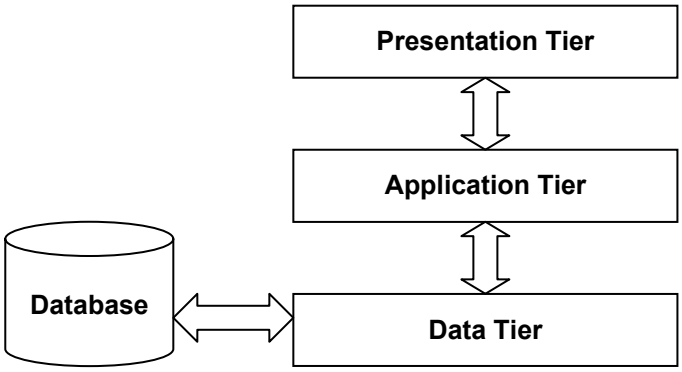


Figure 7.2 – 3-Tier Architecture (software view)

In this type of architecture the Presentation tier communicates only with the Application tier and never directly with the Data tier. The Application tier is in the middle and communicates with both the Presentation and the Data tier.

The Presentation Tier contains the presentation logic, Application Tier contains the business logic and Data Tier contains the data storage logic. Each layer can have its own components. Presentation tier components do not access the database, all data is provided by the Application layer.

Does 3-Tier Software require 3-Tier Hardware?

The answer is No. It is possible to have all types of component stored and executed on the Client device. Indeed, this is how the system prototype is built and tested.
However, as this architecture has all its business logic contained within service components it is an ideal candidate for being deployed on 3 layers of hardware.

To conclude, some of the advantages of the 3-tier architecture are:

- Encapsulation - To separate functionality from presentation. Actually this is the main principle of this architecture and that is why the 2-tier architecture evolved into this one.
- Adaptability – It is easier to modify or replace any tier without affecting the other tiers. For instance, the presentation and application tier are not affected by changes in the data tier in the case that it were a re-definition of the storage strategy.
- Reuse - To save development manpower. To code each bit only once and use it for similar functional needs.
- Quality - For each layer a specialist can contribute with his specific expertise. A GUI designer for the user interface, an expert programmer for the business layer, and a database or ontology designer for the knowledge base layer[10].

7.1.2 Horizontal architecture design

This system is partitioned into the following logical tiers: the Presentation Tier, the Application Tier and the Data storage Tier. Thus, previous Figure 7.2 can also be applied to the system with the condition that instead of a database as the option to store the data other alternatives can be found.
At this point only a brief approach to the tiers that build up the system architecture will be given, since each layer will be treated in detail in further sections.

The Presentation Tier

This is the "top layer". It contains all things that are visible to the user, the 'outside' of the system, such as screen layout and navigation. It also contains all the logic for accepting input from the user and displaying results (“presentation logic”). That is why sometimes this layer is also called User Interface tier.

[10] Obviously that is not the case of this project, since its nature is purely academic and is accomplished by a single person from beginning to end.

Physically this tier corresponds with the web browser running on the client device. By means of this tier, the user will be able to send requests to the Application Server tier and will also be able to navigate static web pages as well as the web pages dynamically generated in the server side.

The design and later implementation of this layer uses techniques like HTML, JavaScript, Servlet and JSP.

The Application server tier

This is the core of the system, the linking between the other layers. The two main functions of the application server are to isolate data connectivity and to provide a centralized repository for business logic. Sometimes it can also be called the Business logic tier (see the Glossary for a definition).

Inside of the application server tier, there is a further division of program code into three logical tiers. This is kind of fractal: the part (application server design) resembles the whole (physical system architecture design). A classic JSP/Servlet system usually implements this subdivision as:

1. JSPs or Servlets responsible for creating HTML user interface pages
2. Servlets or Java classes responsible for business logic
3. Servlets or Java classes responsible for data access.

For the system it will be considered that those servlets in charge of presenting results to the user are part of the “presentation logic” of the previous tier. Otherwise, it can also be seen like a slight logic overlapping between these two tiers.

In this layer are found things like classes, objects, instance variables, methods, polymorphism, encapsulation and inheritance. The objects mostly have a temporary nature. They "live" just in memory for the duration of a transaction or session.
This layer will be responsible for retrieving relevant documents and extracting information from them, among other tasks.

Data storage Tier

This tier takes care of persistency. The underlying technology to implement this layer could be a variety of things including server side files, a knowledge base or a database. The several approaches that are suggested to store data will be discussed in detail in section 7.4. For now, just to release that after the research made on the Protégé software tool (within the ontology survey carried out in Chapter 4) the favourite approach is the one that uses a knowledge base to store the information extracted in the IE process.

What is for certain known, at this point of the design phase, is that an ontology will be used to drive the information extraction process and that this ontology has to be kept somewhere. For this system the ontology will be considered to be part of the data managed and therefore it will be included in this tier, although it is used in some of the processing of the “business logic” tier. Usually ontology and knowledge base are terms that always come together (sometimes even in the same tool). So the system will be consistent with this natural association.

To conclude note that business-objects and data storage should be brought as close together as possible and that ideally they should be together physically on the same server. This way - especially with complex accesses - network load is eliminated. The prototype of this system is built that way.

7.1.3 Vertical architecture design

Along the former section the system was looked at in "horizontal" by describing, in broad terms, the three different tiers. In this section a vertical look at the system will be offered and so every layer will be divided into components or modules. See figure below for a summarized overview (horizontal and vertical) of the system architecture.

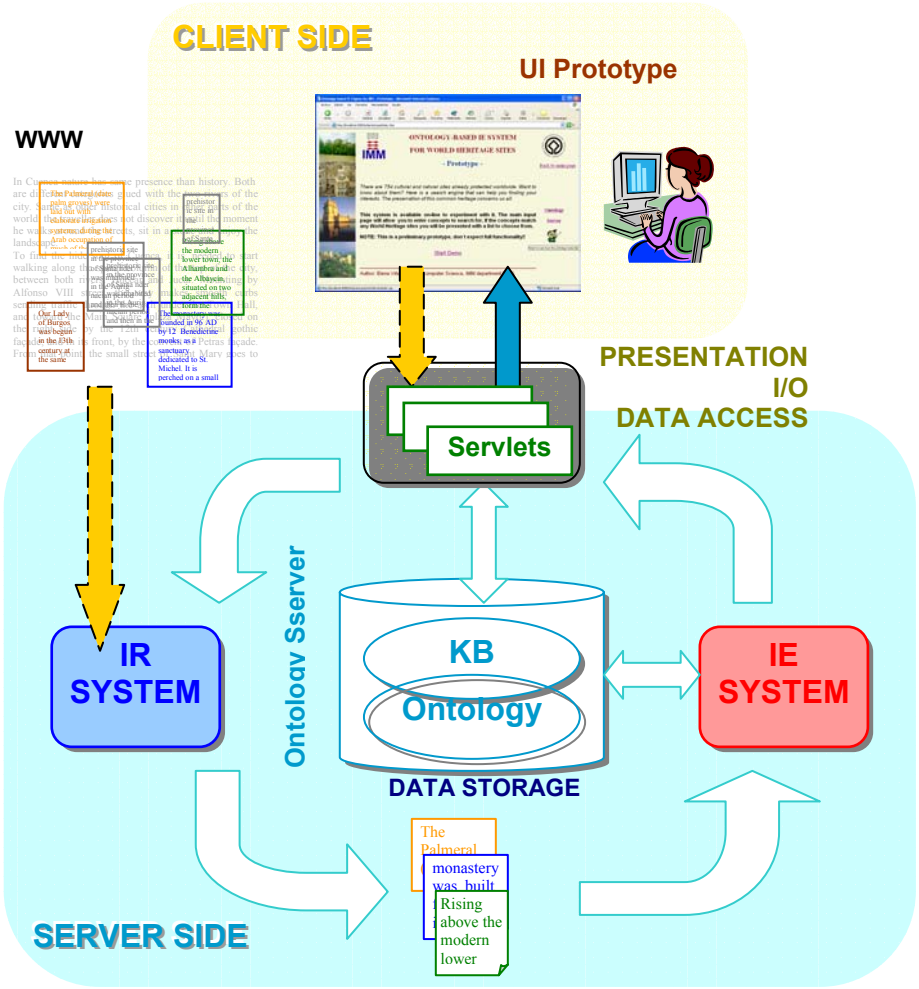
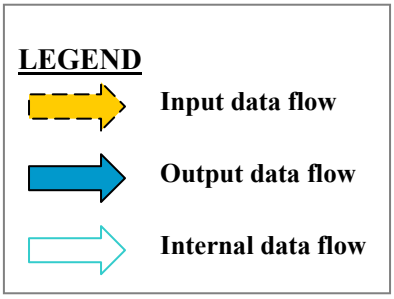


Figure 7.3 – System Architecture



In this figure it has been considered to use a KB as the approach to store data, but it could easily be adapted to other possibilities of making the extracted data persistent. After all a KB is not more than a repository of information referring to a particular domain and so can be considered a database or a collection of server side files to a certain extent. In the system architecture

the KB and the ontology (data tier) are in the same server as the business logic. But both could be somewhere else if needed; the 3-tier architecture of the system easily allows this. This way the non-functionality requirements on flexibility and scalability are fulfilled.

As mentioned in the introduction to the 3-tier architecture given, layers can be divided into components. Today the term component pre-dominantly describes visual components on the client-side. In the non-visual area of the system, components on the server-side can be defined as configurable objects, which can be put together to form new application processes.

Within the presentation tier the following components can be identified:

A. User Interface component – Several HTML, JSP pages that shape the layout of the prototype web site
B. I/O component – Servlets responsible for user's input and output

Within the middle tier or application server tier the following components can be identified:

A. Information Retrieval component – IR system in charge of providing the IE system with a relevant corpus of documents.
B. Information Extraction component – IE system in charge of annotating the web pages received and transform then to an XML format
C. Data access component – Objects (classes, methods, even servlets if necessary etc) in charge of the access to the data storage. If the data storage would be located in another server (not the case of this system) some specific servlets would be necessary to access the information and serve as a gateway between the client tier and the data tier.

Within the data tier the following components can be identified:

A. Population component – Objects in charge of populating the data storage with the new information extracted
B. Querying component – The way of making queries to the data has been separated into a component because it can become very complex
C. Data control component – This component will be in charge off all the things dealing with the data management, validation if necessary, control of duplicates and so on.
D. Ontology component – the ontology modelling the system domain. To a certain extend it could also be considered as a mediation sub-layer between the IE process and the KB. As mentioned in the previous section a design decision makes it be included within the data tier.

The system ontology and knowledge base that are managed in the data tier should be accessible via an ontology server.

Several sections will follow to explain in greater detail the internal design of some of these components. Not all of them will be explained, just the most relevant in terms of design.

To conclude on this part it should be stressed that the system architecture has been designed to allow different approaches for data storage, for information extraction and for information retrieval. This is achieved by encapsulating these logics in tiers and then in components.

7.2 Presentation Tier

This tier has to do with the way the user will perceive the system and how to get from the user and present to the user the results of the processing made. Therefore, all the internal matters have to remain transparent for the user. The "business logic" of the system is carried out in the application server layer as explained in previous sections.

The presentation tier deals with not only the User Interface (UI), but also the way the data is displayed to the user as well as the way the user's input is accepted.
In an informal way it could be said that the presentation layer is made of several items: a **web site** (that is in fact what the user will perceive from the system) and "something" that will link it to the server processing.
The web site (UI component) will consist of several **html** and **jsp** pages conveniently linked. The "something" that will link the two worlds will be a series of servlets (I/O component) stored in the server and being invoked in the html and jsp files. Technical details about how to implement and link these files are out of the scope of this chapter and will be explained in the next one. This section will focus more in the design, features and layout of the UI.

A preliminary layout of the web site is made, trying to fulfil the requirements concerning user interface and usability (refer to Chapter 3 for further detail). Some design patterns are given as well. With those bases the prototype of the web site is built step by step, adding more functionality throughout the phases.

7.2.1 Web site layout and design

Before moving on to build a preliminary prototype of the web site it seems necessary to make a sketch of it.
The following diagram shows the relationships between the pages that are going to be part of the web site.

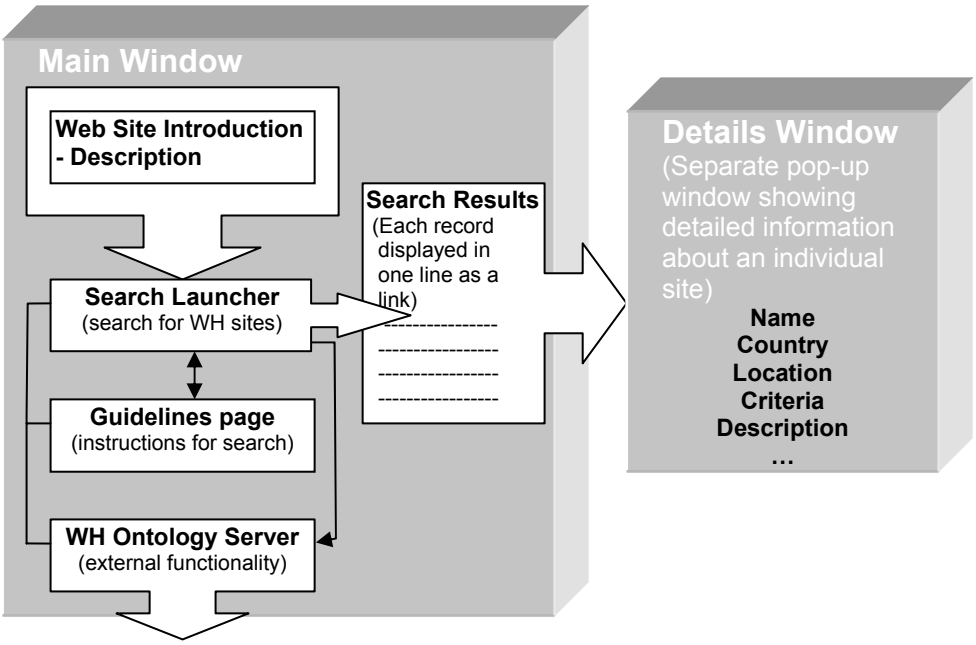


Figure 7.4 – Web Site Diagram

Most of the pages will be shown in the main window. Only when getting results from the server, the user will be given the chance to choose the site and thus get a separate window showing the details about that WH site (in principle the information shown there will be the one found critical while doing the Domain Analysis phase).

An external functionality has been included in the web site; it is about the ontology server. When doing the ontology survey (see section 4.4.2 in Chapter 4 for further details) it was seen that Prótegé Ontology Editor has a lot of plugins that add extra functionality to the editor. One of those plugins is the Protégé Web Browser and that is what will be used in the web site (conveniently adapted to the system needs). For more information about this plugin refer to section 8.2).

All the pages concerning the visualization of the ontology and its instances will be shown in a different window since this functionality is not considered the core of the system and was not specified while settling the System Requirements[11].
This functionality has been added to give the user a "bonus" and make him able to see the ontology that is being used and also make some queries to it.

Concerning the design of the web site, some guidelines to follow will be given below:

Design & Style Guidelines

- Separate what is presentation of what is functionality
- The design suggests a minimum set of data elements that should be present if a WH site is to be listed on the web site. The elements are:
    - Name

[11] The code concerning this extra functionality is not included in the appendices.

- Country (State Party)
- Year/s of inscription in the list
- Criteria
- Description
- URL of the source web page

These elements were selected because: (1) they are likely to be the minimum elements that a user interested in a WH site requires to know and (2) it is reasonable to assume that these elements will always be available for a large number of properties (even in the case that the working domain would change to include more web sites to search on).

- Provide visual continuity between all pages. A banner graphic should appear at the top of each page (optionally a lateral banner too) and load quickly.
- Provide with links to the home page in all the web pages.
- Provide with tips in the pictures that are links to external web pages.
- Use along the web site the same kind of font, color and size for text with similar function.

7.2.2 GUI Prototype

A prototype[12] is built following the design and style guidelines and the web site layout settled in the former section.
Some screenshots are given and subsequently explained (although some of them are quite self explanatory).

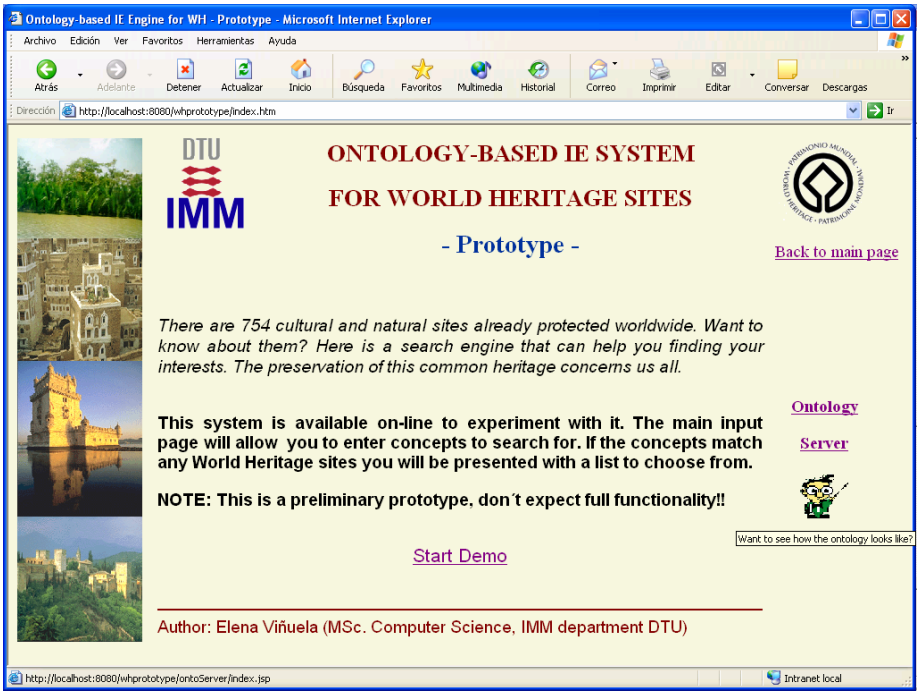


Figure 7.5 – Home page (Prototype screenshot)

[12] All the screenshots used in this section have been captured from the prototype already running and show real results.

The home page is the first page that the users see. It has to capture the attention of users. To include a header and a graphic banner with some pictures (and optionally footer) in all the web pages gives uniformity to the web site. The remaining space on the home page screen is filled with introductory text and hypertext links to other parts of the web site. External links to other web sites are included only in the header.

The prototype's home page has several hypertext links with the following purposes:

- The "Start Demo" link leads to the main search page, where the user can build a query and find specific WH sites.
- The "Ontology Server" link leads to the extra functionality that allows the user to see and browse the ontology used in the background.

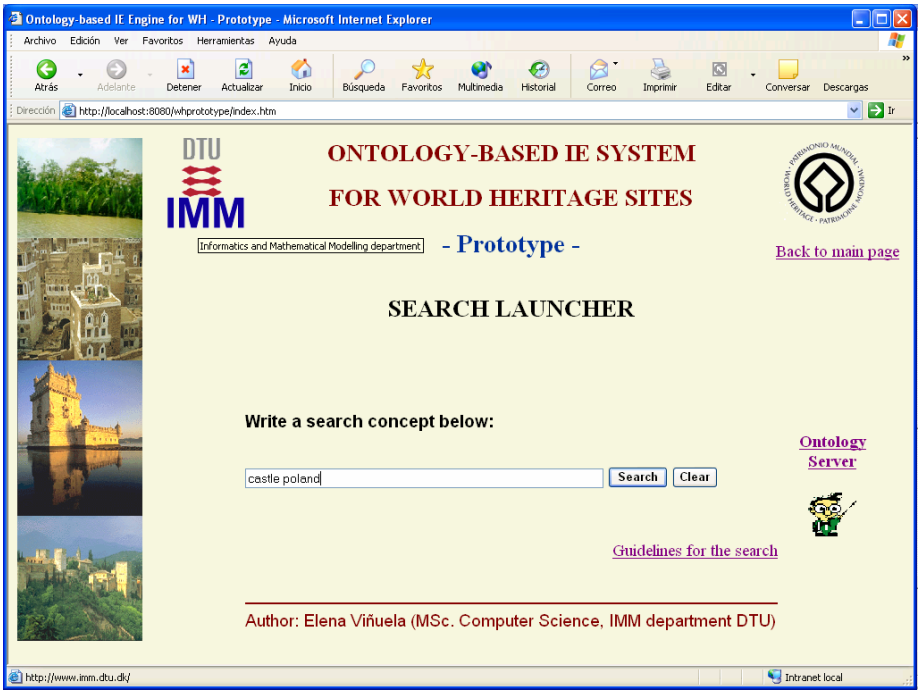


Figure 7.6 – Search launcher (Prototype screenshot)

The search launcher page is the core of the web site. It allows users to enter their queries and launch the search process by clicking the "Search" button or pressing "enter" in their keyboards. It also allows the user to clear the input text box by clicking the "Clear" button. When a user submits his query the next page displayed should show a list of all the WH sites that satisfy the request (see Figure X).

The process of building a search query should be simple and straightforward for the user. The search function should default to a logic, such as Boolean, requiring that search results match all the selected concepts. Exceptions to the default might occur when the user applies some advances options for the search, such as the Boolean operators "OR" and "-". Some help about the search is offered in this page by means of a link to some guidelines.
The search function should also control that the user does not leave the text field in blank, warning him otherwise. See figure below.

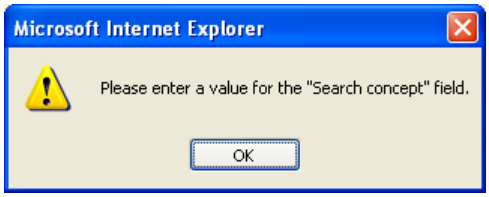


Figure 7.7 – Warning dialog (Prototype screenshot)

The search launcher page has several hypertext links with the following purposes:

- The "Guidelines for search" link leads to a page with some instructions for the search function and information about restrictions (see Figure X).
- The "Ontology Server" link leads to the extra functionality that allows the user to see and browse the ontology used in the background.

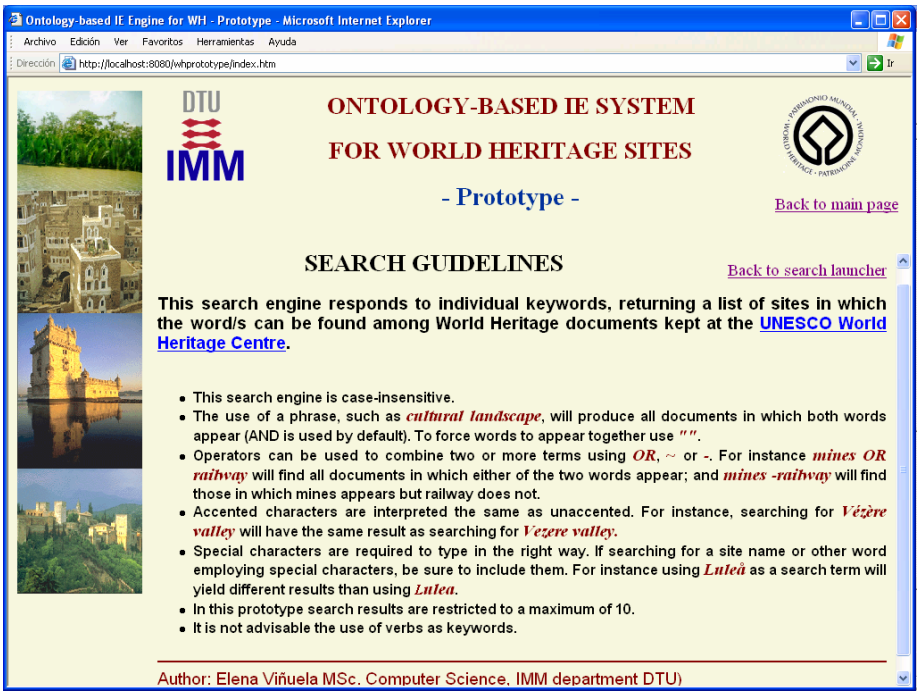


Figure 7.8 – Search guidelines (Prototype screenshot)

This page offers some hints to help the user in his search. It can only be called from the search launcher page. It offers a link to come back to the mentioned page.
For a more detailed explanation of these guidelines refer to section 7.2.3.

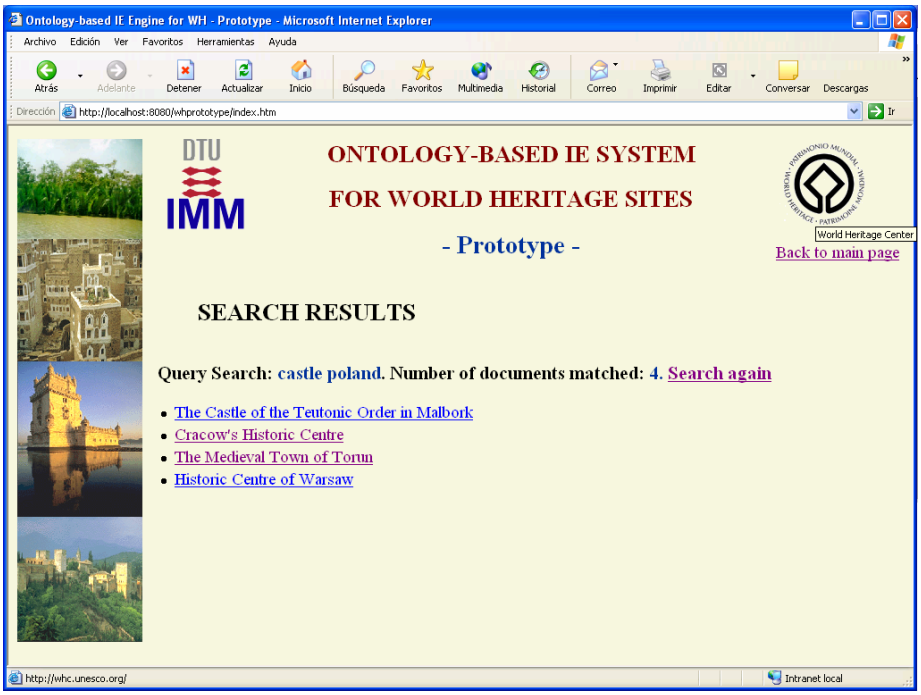


Figure 7.9 – Search results summary (Prototype screenshot)

The search results page shows a summary of all the WH sites found matching the user request (each in a line with the format of a link). It also gives some feedback information about the query made by the user and the number of sites found.

The user can view detailed information about any of the WH sites shown in the summary list. When the user clicks on a site in the summary list, more information about it appears in a separate window.

The information that appears it has to be at least the one considered critical during the System Analysis phase.
There are several ways to show the user the results of his request to the system. The way chosen for this preliminary prototype is to show this information by highlighting (by using another background colour) the annotations previously made over the source page by the Information Extraction System.
The page shown will not be the original but a copy of it adequately annotated. This copy will live temporarily in the system server. See Figure 7.10 for an example.

Apart from the links that lead the user to the sites found, the only link in this page is the one that allows the user to make another request.

The following screenshot, as mentioned above, shows an example of the way the user perceives the results of the processing. It is opened in a separated window so the user can have as many windows as he wants open at the same time.

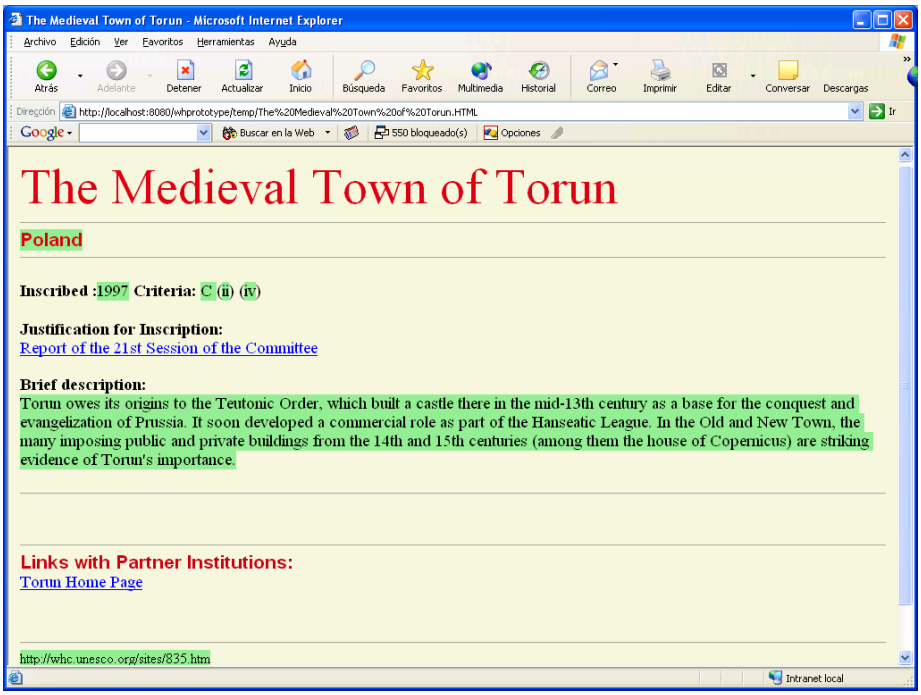


Figure 7.10 – A WH Site annotated (Prototype screenshot)

The last screenshot to be presented is concerning the ontology server (considered as an external application as it was mentioned before). Only the main page of this sub-system will be shown. This pages leads to a series of web pages where one can visualized the Protégé ontology. It only allows (through a hyperlink) to see the ontology in DAML+Oil format.

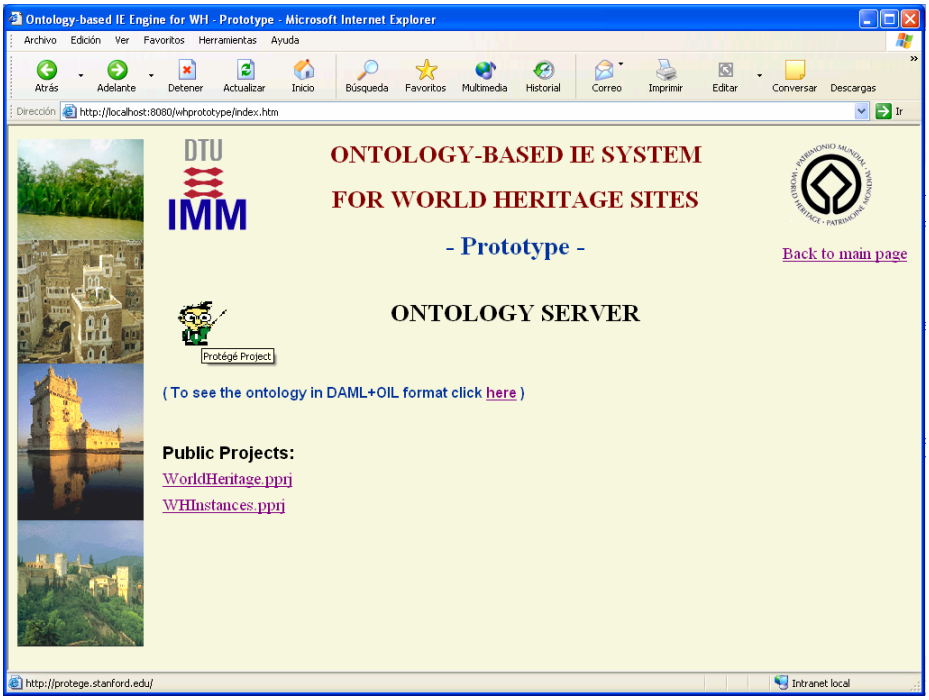


Figure 7.11 – The WH ontology server (Prototype screenshot)

This preliminary prototype offers basic functionality for the moment. It can be improved and extended in coming phases.

7.2.3 Detailed Search Guidelines

Search guidelines are made to please the "search ambitions" of the users and to help them to build a request by giving some useful tips.
By providing the user with this guide the system fulfils some of the functional and non-functional requirements specified in the analysis phase (see Chapter 3 for more details).

These guidelines are explained in detail below. The sentence in quotation marks and italics is the original one that appears in the help page of the web site.

- *"The search engine of the system is case-insensitive."*

  This means that no matter that the user types **Europe** or **europe** or even **euRope** since the search results are not going to be affected at all.

- *"The use of a phrase, such as* **cultural landscape***, will produce all documents in which both words appear (AND is used by default). To force words to appear together use* **""** *."*

  This means that by default all the concepts typed in the text box are logically linked together with an "AND" boolean operator. For instance, if a user types **gothic cathedral** he will get all the documents referring to sites where these two concepts can be found (in any order or in any place).

  Search for complete phrases is also allowed by enclosing them in quotation marks. Words (two or more) enclosed in double quotes ("like this") will appear together in all results exactly as the user has entered them. Phrase searches are especially useful when searching for proper names like for instances **"New Mexico"**.

  It is very important to be aware of this fact because searches like **New Mexico** and **"New Mexico"** will lead to quite different results (probably the second search will get a subset of the documents found by the first one). It is also very important to be aware of the order of the words if using this advanced option for search. Probably a search like "**cathedral gothic"** will lead to zero results, using the example mentioned before.

- *"Boolean operators can be used to combine two or more terms using* **OR** *and* **-***. For instance* **mines OR railway** *will find all documents in which either of the two words appear; and* **mines -railway** *will find those in which mines appears but railway does not."*

  This engine (as internally based on Google search) supports the logical "OR" operator. To retrieve web pages of sites that include either word A or word B, the user has to type an uppercase OR between terms.

  For example, to search for sites that have either a waterfall or a lake, just type **waterfall OR lake**.

This engine also supports the discrimination operator "-". Sometimes is useful to exclude a word from the search by putting a minus sign ("-") immediately in front of the term avoid. (Be sure to include a space before the minus sign.)

For example, to find sites about mines but not about gold mines type: **mines –gold**.

Another very useful operator is the tilde sign ("~") which allows the user to search not only for a particular keyword but also for its synonyms. The tilde has to be placed immediately in front of the keyword.

This is particularly handy when applied to plurals. For example, to search for crocodiles (one or more…) one could type **~crocodile** and so get results of sites where both words "crocodile" and "crocodiles" appear. Another more serious example could be **~church**. By typing that, one will get sites containing both "church" and "churches". In this way a concept is not discriminated by its cardinality.

Another example of the use of this operator is the query search **~volcano**. It will give the user not only the web pages containing the terms "volcano" and "volcanoes" but also those ones containing terms like "earthquake" or "volcanic activity".

These operators are extremely useful to increase the accuracy of the searches, they fine-tune keywords. They are likely to be used for "more ambitious users" (in the context of ambition it is always meant with respect to the system end-users).

- *"Accented characters are interpreted the same as unaccented. For instance, searching for* **Vézère valley** *will have the same result as searching for* **Vezere valley***."*

  This rule is referring only to accentuation marks. This means that words like **Córdoba** (in the south of Spain) and **Cordoba** are the same for the search engine.

- *"Special characters are required to type in the right way. If searching for a site name or other word employing special characters, be sure to include them. For instance using* **Luleå** *as a search term will yield different results than using* **Lulea***."*

  This rule is a little bit like the contrary of the former one. Special letter of other alphabets different to English have to be typed in their correct way. To keep on with Scandinavian places another example of this could be the word "Helsingør". For the search engine **Helsingør** and **Helsingor** are two different words. Probably if a user types the second search he will get no results at all (considering that proper names are written in this domain with all the rigour demanded to an official site).

- *"In this prototype search results are restricted to a maximum of 10."*

  This is a restriction of this prototype and it is due to the internal use of the Google API. Even though the use of this API is free it has some restrictions (despite of this fact, Google API was found very suitable for a prototype purpose).

- *"It is not advisable the use of verbs as keywords."*

  This is in general a good advice for any search since verbs are very unpredictable words due to their tenses.

  Also at this point to remind that this engine, like the rest of the search engines, ignores common words and characters as well as certain single digits and single letters (because they tend to slow down the search without improving the results).

7.3 Application server Tier

As seen from the division of the Application server Tier into several components, done in the definition of the system architecture, the two more important ones are the Information Retrieval (IR) component and the Information Extraction (IE) component. After concluding from latter chapters that Information Extraction IS NOT Information Retrieval and that these two techniques link very well helping each other lets move on to explain how these techniques are going to be used within this system.

7.3.1 IR component Design

Some research prior to the design of the best strategy for retrieving documents was done in order to have a better view of the real needs for this component.

Preliminar work

A preliminary test of possible queries against the system was prepared and executed using a general search engine. Searches were customized (by means of the *Advanced Search* option) to match the range of pages chosen in the domain analysis phase (see section 2.2 for more details). That means that they were restricted to the **English** Language and the **whc.unesco.org** site.
To be more precise Google was the search engine used in these tests after trying some others and concluding that its *Advanced Search* option was the more powerful and easy to use and the one offering more accurate results as well (as far as it can).

The "tower problem"

While performing the preliminary tests a problem was found dealing with the content of the documents obtained. That problem is going to be called from now on the "tower problem" but could also have been called the "bear problem" and so on. The example of the "tower" keyword search will be used to generalize about these issues and will be next explained.

Let us personalized a little bit at this point. As users we are now we are curious to know about the WH sites in the world that are of an outstanding nature for having a tower or anything dealing with a tower. It can be a tower itself, a tower of a castle, a tower of a cathedral or whatever is relevant to mention dealing with a tower.

If introducing the keyword "tower" in a search engine like Google for instance (previously set up in its advanced search options) more than a hundred web pages are found matching this word. From all those pages only 13 belong to the domain of interest.
Taking a look at each page, as human readers we are, we found out that some of the pages offered to us were not at all what we were expecting from our search. We got some results like for instance the *Old City of Sana'a* in Yemen (http://whc.unesco.org/sites/385.htm) or the *Historic Centre of San Gimignano* in Italy (http://www.unesco.org/sites/550.htm) which are very nice places indeed but are relevant for having "tower-houses" and not for having towers that was what we were looking for.
We keep on checking those 13 pages and now we found a site called *Hawaii Volcanoes National Park* in the USA (http://whc.unesco.org/sites/409.htm) and we wonder why is that we got that site that is far from being dealing with "our longed-for towers". Reading a little bit closer in the page we see that there is a sentence that says:

> "*This site contains two of the most active volcanoes in the world, Mauna Loa (4,170 m high) and Kilauea (1,250 m high), both of which* ***tower*** *over the Pacific Ocean".*

Here the word "tower" is used as a verb instead of as a noun. Now we understand why we got that page in the set of web pages matching our query but it should not be there anyway.
Finally only 9 pages among those 13 are really "relevant" for us semantically talking. Some of towers found are very famous by the way… (The tower of London, Eiffel Tower, the Pisa 'Leaning Tower' or the Tower of Belem is Lisbon for instance). But we are not so satisfied since we lose some time looking at pages that were not of our interest.

This is an example of a problem while finding in the WWW relevant documents within a specific scenario. Some more similar examples can be found about this problem. For instance, while trying to find information about "bears" (the animal) a lot of pages are obtained in which this word is used as a verb.

Results obtained from these preliminary work were not to satisfactory; vast amount of documents in many cases and with contents far from being the aim of the queries. It became obvious that some sort of Information Retrieval subsystem was necessary to obtain a set of relevant documents from the World Heritage scenario to pass as input to the IE system.

Approach to Information Retrieval

The IR sub-system of this system must be able to obtain a set of relevant documents in the subject area of World Heritage by just getting some **keywords** from the user. All the aspects of the search within the specific web domain must remain transparent for the user. This means that somehow the IR system will internally build a query

considering the user keywords and the domain scope and constraints. This query will be used to select relevant documents from the original set which finally will be the input to the IE sub-system.

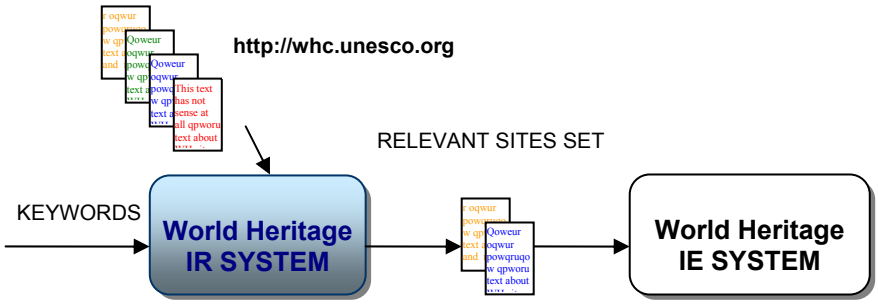


Figure 7.12 – Approach to WH IR system

How the IR system will internally perform the selection and solve the "tower problem" and the like will be discussed next.

After the brief survey on search engines done in the research phase, it was decided to use Google to make the first "harvesting" of documents. This tool was seen the most suitable not only because it is one of the best search engines available but also because it is one of the few that offers its API for free for anybody whot wants to add search functionality to an application.
These are the steps to follow inside the IR process:

- Take the keywords from the user and with them construct an appropriate search string, adding as parameters and constraints as necessary.
- Use the API of Google to launch the search string.
- Filter the result received from Google's API based on the names of the domain (see section 2.2.3 for further details).
- Add some sort of mechanism to "semantically" filter the documents
- Pass the set of documents that is left to the Information Extraction system.

7.3.2 IE component Design

From all of the IE techniques studied and summarized in Chapter 5, the ontology-based approach is going to be used for this system. Also some pattern matching and NE techniques will be applied in certain cases, thus taking advantage of the layout of the system domain. So, in fact, a combination of several techniques will be used.
A tool that supports all these techniques of information extraction is GATE, from the University of Sheffield [D]. GATE is the tool that will be used in this system to parse and annotate the web pages received from the IR component. For further detail about this tool please refer to the System Implementation chapter.

Using an ontology to support IE is also aimed to increase the system's portability to other domains. By replacing the current World Heritage ontology with an ontology of

another domain, the Information Extraction component should still be able to function and extract some relevant knowledge. However, certain domain specific extraction rules will eventually have to be modified to fit the new domain.

The first step in the IE process is to create a corpus of documents with the documents received from the IR component. All the later processing will be done over this corpus through a pipeline. Some of the processing resources (PR) that ANNIE system provides with (see section 6.1 of this documentation for more detail) will be executed over the corpus: English Tokenizer, Gazetteer and Sentence Splitter.

Specific grammar rules have to be built in certain cases. That is the case of this system; which relies to a certain extent in the layout of the static web pages of the working domain. To help to recognize some of the information considered critical a JAPE grammar was built. This grammar can be found in APPENDIX E (the files include commentaries). No further explanation about the grammar will be given here since it is considered enough explanatory and, in addition to the files containing the JAPE rules, a study about GATE's JAPE language and some information about regular expressions were given in Chapter 6 and APPENDIX D respectively.
A JAPE transducer (another of the processing resources) associated with the grammar specifically created to match the domain layout will be executed over the corpus. The files that make up this grammar will also be stored in the web server.

Once the right annotations over the input corpus are performed by the IE tool they have to be directly mapped with the classes (concepts) of the World Heritage Ontology. Some problems came out at this point due to some lacks of the annotation tool. The problem with GATE is that it does not recognize attributes and relationship in an ontology. It is only capable to recognize and map classes. This circumstance made that another ontology were created specially for this reason. The new ontology is a reduced version of the domain ontology where the relevant slots (attributes) have been converted into classes and with no attributes. This reduced ontology can be seen in appendix B2.

7.4 Data storage Tier

This tier is responsible for data storage. As data it will be considered not only the information extracted and stored but also the ontology itself.

After the ontology survey done in Chapter 4 and have been chosen Protégé-2000 as the ontology-environment tool, this section will deal with the specific design of an ontology for the system domain. As mention while describing the problem scope, the study of the ontology itself is part of the work carried out towards a semantic-based application.

Also in this section some approaches to the problem of data persistence will be given, as well as the approach chosen for this system.

7.4.1 World Heritage Ontology design

One of the requirements for this system is to build an ontology that represents the reality of the World Heritage domain. The ontology editor chosen to develop the ontology is Protégé-2000 version 1.9 [C]. The ontology language chosen is DAML+OIL (refer to [15] and [16]). From the study of the types of ontologies, done in section 4.2.2, the ontology needed for this system is an *application ontology*. The final ontology developed can be found in APPENDIX B.

The approach to building the WH ontology was to start it from scratch and follow an iterative process along the entire lifecycle. The option of reusing an existing ontology was not considered since the ontology is very specific for these domain and application and the system does not have to interact with any other application already committed to a particular ontology.

According to [10] some fundamental rules in ontology design are:

1) *There is no one correct way to model a domain – there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate*
2) *Ontology development is necessarily an iterative process.*
3) *Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.*

All the knowledge about the domain, acquired along the domain analysis phase, was used to build the ontology. The first steps were to develop the class hierarchy and to define properties (slots) for the concepts. The entity relationship model and the preliminary classification done during the domain analysis were used as a base to perform these tasks; after that the ontology was revised and refined several times to add more detail.

In the development of the class hierarchy a top-down approach was followed, starting with the definition of the most relevant concepts and specialising them subsequently. Some stages of the building process will be presented next. Please note that not all modelling decisions can be given in this section, which nature is more introductory and explanatory than technical[13].

Naming conventions

Some naming conventions were defined and strictly applied in the process of building the ontology to make it easy to understand. These guidelines are:

[13] For more information about the system ontology please refer to APPENDIX B or to the files containing the ontology that are provided together with this master thesis.

- Do not use the same name for classes and slots. In fact, this condition was imposed by the editor since Protégé-2000 maintains a single name space for all its frames. Besides, Protégé-2000 is case sensitive.
- To use lower case for slot names.
- To capitalize class names
- When the name of a concept (class) contains more than one word (such Natural Category) write the words together and capitalize each word[14].
- When the name of an attribute (slot) contains more than one word (such Property Id) write the words together but in lower case (propertyid).
- Class names are written in singular
- Class are named using noun phrases
- Slots that stand for links are named using verbal phrases and are labelled in both directions (inverse slots).
- To avoid abbreviations in concept names
- Names of direct subclasses should include the name of the superclass

Preliminary hierarchy of classes

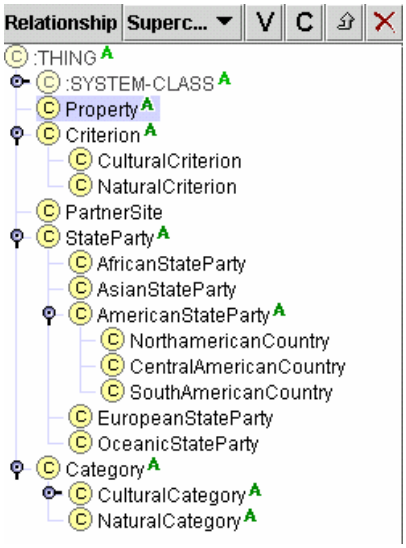


Figure 7.13 – Preliminary hierarchy (Protégé-2000 screenshot)

The most relevant concepts were sketched first, in a fast prototype of the knowledge model. This first hierarchy is shown in the figure on the left. Each concept is represented as a class (*Property*, *Criterion*, *PartnerSite*, *StateParty* and *Category*) and subdivided into subclasses if appropriated (*CulturalCriterion*, *NaturalCriterion* and so on). Some attributes were also added to each class.

A natural division of State Parties, grouped by the continent where they are located, was considered appropriated at the beginning. This subdivision was considered to be meaningfully useful in order to give the possibility of querying the ontology over regions and continents. Later on it was seen that this specialization of the countries does not contribute too much in the purpose of the ontology, that is not other than to drive an extraction process.

Classifications of categories

Only the hierarchy of categories will be presented and explained in this subsection. As many times repeated, the category (or categories) in which a WH site can be classified falls into two main groups: Natural categories and Cultural categories. These categories can be split into other subcategories. A preliminary classification was made at an early stage of the system analysis (see section 2.2.4). This taxonomy was used as a reference for a first approach to the design of the ontology; later on a thoughtful classification was built.

[14] Protégé-2000 allows spaces in concept names but it was not considered to use them to avoid incompatibilities with other systems that may interact with the application.

When dealing with the problem of building a hierarchy of categories for a WH site sometimes it is not easy to asses how much of cultural and how much of natural a property entails. It is not just the constructed heritage or the natural environment on their own; it is the common atmosphere where one can find the synthesis of the natural, artificial and anthropological (social relationships, religious beliefs etc) values. The boundaries among the areas of heritage protection have become scarce.
Another problem when classifying WH sites is the grade of specialization that can be reached with the information provided in the description field. Sometimes this information is too general, thus a site has to be classified in a medium level of specialization.

When building the ontology some concepts where found to belong to more than one category. There is no problem to handle those cases since Protégé-2000 supports multiple inheritance.

After building the ontology with all the subdivisions of the categories it was found that the taxonomy of categories can not be used for the purposes of the system so it will just remain in there for theoretical purposes.
The information that is going to be extracted in this stage of the prototype is the basic information concerning a Property (with some of its attributes) together with the information about States Parties and Criteria. Trying to identified the category of a World Heritage parsing the content of the web page and identifying keyword is completely out of the scope of this project. The character of the domain is too wide to allow such functionality in an initial study.

Due to the restrictions of GATE (see section 7.3.2), a second ontology had to be built, a reduced one without the categories hierarchy, to be able to extract information from it mapping its classes with annotations done over the documents. The second version of the World Heritage Ontology can be seen in appendix B2.

7.4.2 Data Storage Design

Storage of the extracted data is also one of the goals that this system aims to achieve. The system's architecture allows different strategies to make the data persistent. Several approaches to the problem will be studied and explained next, together with the reasons to be or not to be chosen.

Knowledge Base approach

The first approach to be considered was using the knowledge base (KB) provided by Protégé-2000 to store the results returned after the IE task performed by GATE. This is the approach that was the favourite from the beginning of this study, because it seemed the most normal to have the ontology and the data instances together. Also because after the survey done on both tools (GATE and Protégé) a good connection between them seemed quite straight forward. Far from that, this approach could not be followed because the API of GATE is not yet prepared to provide this easy connection to

populate a Protégé KB. What it is yet possible to do is to have a Protégé Project (an ontology) embedded in GATE with almost the same GUI that in the original tool.
Since this approach was not possible to implement further considerations like duplicates, synonyms, automatic consolidation of the KB and so on could not be further investigated.

However, it is known that this approach is feasible since the Arquetakt Project [E] has been successfully coupling GATE and Protégé, among other tools, to perform an automatic knowledge extraction and biography generator system ([4], [5] and [6]).

Database approach

This approach can be split again into two sub-approaches. The first one entails the use of the mechanisms that GATE offers to make the data persistent. The second one entails to have a separated database and make the connection between both environments manually through Java code. Both will be explained in detail below.

Gate's persistence approach

As GATE is the tool used to perform the information extraction (IE) task, it was further investigated to discover the support that it provides for data storage.

GATE is capable to assure persistence for its resources. These layers of persistence are various, including database persistence. Depending on the purposes a simple or complex level of persistence may be required. According to the user's guide [22], the types of persistent storage used for Language Resources (LR) are:

- Databases (like Oracle or PostgreSQL);
- Java serialization;
- XML serialization.

Only the first one will be discussed in this subsection.

GATE gives support to two different database data stores, Oracle (for Windows NT, Windows 2000 and Linux platforms) and PostgreSQL (only for Linux platform). At present GATE supports the following versions to be used as repository for GATE data: Oracle 8i, Oracle 9i, PostgreSQL 7.2 and 7.3

As the system prototype has to run under a Windows platform only the possibility of Oracle was investigated. Before being able to work coupled with GATE, these database servers have to be configured first (for further details refer to the manual at http://gate.ac.uk/gate/doc/persistence.pdf, written specially for this setup).
Oracle 9i Database Release 2 for Window NT/2000/XP was downloaded and installed. To stress that running an installation of Oracle is not for the faint-hearted, as it is warned in [22]. After several days spent in the installation and setup some conclusions were reached:

a) Certainly the user guide did not exaggerate
b) Oracle 9i is an excellent database server; however its use is out of the scope of this prototype. First because it requires too many resources (both in disk space and processor use), resources that are not available. Second because it is indeed

too powerful for the real needs of this application. And even a third reason of technical nature: since this tool is also a server, it interferes in the prototype's web server and makes it not work anymore. Probably this last issue could be solved with some more research, but considering the time resources for the thesis and the time required to manage this tool is not worth to try.

Obviously this approach was dismissed for the current prototype. It can be reconsidered in case of having to cope with future storage needs. The study carried out, during the system analysis phase, about a possible database schema could be used in this approach. Refer to appendix A3. for further detail.

Independent Database approach

This approach entails the use of an "external" database (external in the sense of not supported by GATE), like for instance MySQL. This popular relational database is open-source. The study about the schema for a relational database (done in appendix A3. can also be applied to this approach. This approach entails the programming of the necessary Java methods to dump the results from GATE's API to the MySQL database. It should be necessary to write some algorithms to collect the results of the annotation process, transform the data to records and make the necessary SQL request to the database.
However, this approach was also dismissed because, apart from the programming challenge, it does not add anything to the objectives of this master thesis. Therefore, no further investigation was made on this approach. The consequences of this database not being directly support for the GATE framework remain unknown.
Other choices of databases could have been done, like to chose an XML database (therefore supporting the storage of both structured and unstructured data).

Server side files approach

This approach uses one of the light levels of persistence provided by GATE, the XML serialization. This is the approach finally chosen due to the fact that the other two approaches were not possible to reach and also due to the lack of time resources for further research.

According to [22] XML persistence doesn't necessarily preserve all the objects belonging to the annotations, documents or corpora. Serializing these arbitrary data types in XML is not a simple task; GATE does the best it can, and supports native Java types such as Integers and Booleans, but where complex data types are used, information may be lost (the types will be converted into Strings).

GATE provides a full serialization of certain types of features such as collections, strings and numbers. It is possible to serialize only those collections containing strings or numbers. The rest of other features are serialized using their string representation and when read back, they will be all strings instead of being the original objects.
When GATE outputs an XML document it may do so in one of two ways:

- If the original document that was imported into GATE was an XML document, GATE can dump that document back into XML;

- For all document formats (including html), GATE can dump its internal representation of the document into XML.

In the former case, the XML output will be close to the original document. In the latter, the format is a GATE-specific one which can be read back by the system to recreate all the information that GATE held internally for the document. This second option will be the one used in this system to represent the information in an XML format.

How to access and make use of the XML serialization?

In the GUI the option of "saving as XML" saves all the annotations of a document together with their features (with the restrictions previously mentioned), using the *GateDocument.dtd*:

```
<!ELEMENT GateDocument (GateDocumentFeatures,TextWithNodes,
(AnnotationSet+))>
<!ELEMENT GateDocumentFeatures (Feature+)>
<!ELEMENT Feature (Name, Value)>
<!ELEMENT Name (\#PCDATA)>
<!ELEMENT Value (\#PCDATA)>
<!ELEMENT TextWithNodes (\#PCDATA | Node)*>
<!ELEMENT AnnotationSet (Annotation*)>
<!ATTLIST AnnotationSet Name CDATA \#IMPLIED>
<!ELEMENT Annotation (Feature*)>
<!ATTLIST Annotation Type CDATA \#REQUIRED
StartNode CDATA \#REQUIRED
EndNode CDATA \#REQUIRED>
<!ELEMENT Node EMPTY>
<!ATTLIST Node id CDATA \#REQUIRED>
```

Using GATE's API, this same option is available by calling *gate.Document's toXml()* method. This method returns a string which is the XML representation of the document on which the method was called. . If called with null as a parameter, then the method will attempt to restore only the original markup. This option makes possible to generate an XML document with tags surrounding the annotation's refereed text and feature saved as attributes.
This option of saving as XML works exactly the same for all GATE's documents so there is no particular observation to be made for the HTML formats that are the type of the documents managed by this system. When attempting to preserve the original markup formatting GATE will generate the document in XHTML. The HTML document will look the same in any browser after processed by GATE but it will be written in another syntax.

After knowing all this considerations about how GATE treats the files when converting to XML, a double strategy combining storage and presentation will be followed in this system.
For each of the files that match the user's query (after have been selected and filtered) two documents will be generated: an XHTML document and a XML document.
The documents created will be named with the unique number that identifies each WH site and stored in a specific folder of the application's web server. In the prototype's website those XTHML files will be the ones browsed and accessed by the users

(through hyperlinks with the name of the site linking to the folder in the web server were the files are kept), while the XML files will be kept as a data repository.

These files are quite small in size, so no problems of space are expected. Besides, some additional mechanism for periodic updating and/or deleting the files can be implemented if considered necessary.
This approach can be applied when the amount of structured or semistructured data to store and display is modest. The advantages and disadvantages of working directly with a presentation format are pretty obvious. It is very handy that the "database" is a self-contained package that can be updated using any text editor and can be directly served by a web server. But on the other hand the information is more difficult to queried and maintain.

Probably this is not the most practical way of managing a collection of semistructured data but it is still fine for this system's initial prototype and thesis purposes and considering that the other approaches were not possible.

Chapter 8: System Implementation

This chapter presents some details and decisions taken during the system implementation phase and a summary of the software tools that were used.

The World Heritage Query System uses many different technologies. Some of these are well proven technologies like Java and some other are still under development. More details about them will be given along the following chapters.

8.1 Programming languages

8.1.1 Java

Java is as much a platform as it is a programming language. Java is indeed a very appropriate platform for the Internet; therefore it suits very well for this system (which covers the development of a web application).
The Java language is platform independent, event driven and object-oriented and that makes it be a great tool to achieve portability, flexibility and reusability (some of the requirements for this system). It is also internationalized by design and uses UNICODE as standard to represent information. This is also very convenient if the system were dealing with other languages rather than English, languages with non standard characters (such as Greek, Chinese or Japanese).

The advantages mentioned before plus the fact that the main software tools used in the development of the application are written in this language (both Gate and Protégé have Java APIs) make Java the best choice of programming language.

The main disadvantage of Java is the speed because this language is interpreted (it is first compiled into byte code and then run for an interpreter called a Java Virtual Machine, specifically designed for a computer architecture). However, since this system implements a prototype, considerations about speed are not so important.

The Java development environment used for building this application was *Java(TM) 2 SDK, Standard Edition Version 1.4.2*. SDK stands for software development kit and therefore contains development tools (such as compilers or debuggers).

The Java runtime environment for executing this application was *Java(TM) 2 Runtime Environment, Standard Edition Version 1.4.2*. A Java runtime Environment contains the Java virtual machine, runtime class libraries, and Java application launcher that are necessary to run programs written in Java. It is not a development environment. For developing tasks Java 2 SDK Standard Edition was used, as mentioned before.

The version of Java that the Java Virtual Machine is running can be seen typing the command "java –version":

```
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\s021403>java -version
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2-b28)
Java HotSpot(TM) Client VM (build 1.4.2-b28, mixed mode)
```

Java Servlet and Java JSP (Java Server Pages) technologies were also used on the server side.

8.1.2 HTML and JavaScript

HTML and JavaScript languages have been used in the implementation of the user interface. The first one to design the appearance of the static web pages and the second one to add some validation control to them.

In the implementation of the HTML static pages Microsoft® FrontPage® 2002 was used. It was decided to use frames for the web application, to provide some continuity in the appearance of the web site. However, this practice is unevenly supported by web browsers. That is why the testing phase of the system will be made under the two main web browsers available: Internet Explorer and Netscape Navigator.

JavaScript form validation tries to ensure that the values entered into a form are correct before they are submitted to the server. While one can never ensure that all information is perfectly correct, at least some obvious errors made by people can be caught.
The main drawback of using JavaScript is that browsers have to enable its use. Some validation on the server side is therefore required. But still, for those users with JavaScript enabled browsers, JavaScript validation can save the server from having to "waste resources" returning error messages itself, as well as save the user time waiting for the server to return those error messages.

8.2 Software tools chosen

Many tools and very different tools have been used to carry out this work. Some of them are open source tools while some others are commercial. Free software was attempted to be used along the thesis but sometimes this is neither possible not practical.

In this section only the most relevant tools, those that are really important to the system, will be covered. The tools considered relevant to the application are: Protégé-2000 [C], Gate [D], Google Web APIs [J] and Jakarta Tomcat Server [H]. A brief explanation will be given for each of them. All the tools presented in this section are open source tools.

8.2.1 Protégé-2000

Figure 8.1 – Protégé splash

Protégé (from Stanford University) is a Java open-source development environment for ontologies and knowledge-based systems. It was chosen for this application because of its popularity and good press and because of the many plugins it has that extend its functionality.
Another reason to be chosen as the ontology editor for this application is because it can be used embedded in GATE. This popular knowledge base and ontology editor was integrated as a visual resource in GATE in May 2002 (http://gate.ac.uk/news.html).
A study on this tool was previously done during the research phase (see section 4.4) so nothing more about it will be added in this section. Only the plugins used in the design and implementation of the system and the reasons for choosing them will be explained below.

DAML+OIL Plug-in[15]

This plug-in is an extension of Protégé-2000 with support for the ontology language DAML+OIL. It allows users to read, edit, and generate DAML+OIL ontologies in the Protégé-2000 environment. The system ontology is build using this plugin. It generates three files that are the following:

- A Protégé-2000 project file (.pprj).
- A DAML+OIL file (.daml) which contains the generated DAML+OIL code.
- An instances file (.daml).

This plugin has currently some restrictions that should be kept in mind:

[15] This plugin only works with versions 1.8 and 1.9 of Protégé-2000; that is why the latest version of Protégé 2.0.1 (release 12 February 2004) could not be used for this application.

- It only supports the XML Schema Datatypes (user defined datatypes and extensions to existing datatypes are not supported).
- Due to restrictions in the Protégé system, the maximum cardinality of each slot (property) has been set to 1000.
- It is recommended that all Namespaces begin with http://, since URI references are absolutes. It is also recommended to end a namespace with a hash mark '#'.
- <imports> tags which are used to reference another DAML+OIL ontology containing definitions that apply to the current DAML+OIL resource are not supported.
- In the current release of the plugin multiple domain and range specifications are not properly treated. Multiple Range or Domain specifications in Protégé are understood as a union, whereas a DAML+OIL property with multiple domain or range specifications is understood to have the intersection of all specified elements as its range.
- The plug-in doesn't deal with facets stated at class level. Only changing the slot at the top level has the proper effect.
- User defined metaclasses are not currently supported.

As it can be seen there are a few tasks that this plugin still cannot perform. Remember that these tools are in constant development.

Protégé Web Browser Plug-in

This plugin has been used as part of the web prototype user interface. During the design phase it was mentioned that this plugin was considered to be an external functionality of the GUI. It was decided to include this plugin in the prototype to give the user the opportunity to see and to browse the WH ontology and also due to academic purposes. This plugin was slightly modified to adapt it to the prototype's user interface. Some of its functionality was limited too, like for instance the possibility of uploading an ontology to the server.

The Protégé browser is a Java based web application that allows users to share their protégé ontologies over the Internet. The application is deployed using an application server (like Tomcat). It provides the essential functionality needed to browse a protégé knowledge base. It also provides functionality to carry out text-based searches through the knowledgebase.

Please note that this plugin is only valid to browse Protégé ontologies and knowledge bases (in a web browser) and not for having a real server of Protégé ontologies to connect to. For the latest there is another plugin called Protégé Server.
As mentioned during the design phase the ontology does not have to be on the same server as the "business logic" of the application. So this plugin could be used to make Protégé worked as a server and therefore have the WH ontology in a different machine. No further investigation was made on this Protégé Server plugin since it was not required for the implementation of the prototype. Just to mention briefly that this plugin is CORBA-Based and that it enables client applications to interact with a Protégé KB server using the Common Object Request Broker Architecture, a mechanism for distributed object communication.

More information about not only the plugins mentioned in this section but all Protégé plugins can be found at http://protege.stanford.edu/plugins.html. Protégé provides its users with several discussion lists (protege-users, protege-discussion, protege-owl and protege-beta); those lists can be found at http://protege.stanford.edu/lists.html.

8.2.2 GATE

GATE (standing for General Architecture for Text Engineering) is an infrastructure for developing software components that process human language. It has been in development at the NLP (Natural Language Processing) Group in Sheffield University since 1995 (initial widespread release in late 1996) and has been used in a wide variety of projects (in particular for Information Extraction tasks) [21]. The system supports the full lifecycle of language processing (LP) components, from corpus collection and annotation through system evaluation.

Figure 8.2 – GATE splash

The underlying motivation in GATE was to provide human language processing researches with a software architecture that makes the arts and crafts for them (storing data on disk, displaying and passing data between processes etc.). Just like a professional mathematician probably regards a tool like *Mathematica* as necessary infrastructure for his work, language engineers and computational linguists should also have some support infrastructure. And there is where GATE came up some years ago, representing an attempting to fill this gap. But, what does infrastructure mean for Natural Language Processing? What sorts of tasks should be delegated to a general tool, and which should be left to individual projects? The position taken in designing GATE was to focus on the common elements of NLP systems.

As a result of this effort GATE comprises an architecture, framework and graphical development environment. The architecture is a macro-level organizational pattern for the components and data resources that make a LP system; the framework is an object-oriented class library implementing the architecture; and the development environment adds graphical tools to access the services provided by the architecture.
The framework and development environment are written in Java and are available as open-source free software from the GATE web site [D].

GATE also provides with some free components and wrappers for other people's components. Protégé support is available in GATE but one has to ask for an individual license.
GATE has a very active and useful discussion list at *gate-discuss@dcs.shef.ac.uk*. To subscribe to this list go to http://gate.ac.uk/mail/.

8.2.3 Google Web APIs

The system uses this tool (as part of the Information Retrieval component) in order to find relevant web pages to extract information from. It was chosen because from all the search engines this was the one to combine powerful search and a free API available. It was also chosen because it provides with documentation and FAQs online as well as technical support via email. An extra and more subjective reason for this tool to be chosen was its popularity in the Internet: the name of Google[16] is nowadays a synonym of simplicity, speed and accuracy. It seems fair to say that Google has itself transformed from a simple search engine into an online cultural icon.

The Google Web APIs service is a beta web program, so it has some limitations. One can not retrieve more than 10 results per query, and cannot make more than 1.000 queries per day. Another restriction is that it can only be used to search Google's main index (of more than 4 billion web pages constantly refreshed), but not to search for images, directories or groups. Despite these limitations it was found very suitable to be used as a resource in a prototype development.

To get access to the Google Web APIs service, one must first create a Google Account and obtain a license key. Every time that a call to the APIs service is made the license key must be included. Having a Google Account and a license key entitles to 1.000 automated queries per day.

To conclude talking about this tool please note that it supports requests and responses in UTF-8, therefore allowing one to make queries in all the languages of the world.

8.2.4 Tomcat server 4

In order to build the three-tier client/server architecture it was necessary to have a web server running on the development environment. Tomcat server was found the more suitable tool since not only it can work as a commercial Web server (for deployment) but also as a standalone server (for development) on the desktop. The version of the servlet container used for this system is Tomcat 4.

Tomcat (sometimes referred to as "*Jakarta* Tomcat") itself is part of the Jakarta Project, which is a suite of Java development tools developed through the Apache foundation [H].

This tool is a Java based web application container created to run servlets and JavaServer Pages (JSP) in Web applications. It has nearly become the industry accepted standard reference implementation for both the Servlets and JSP API. Version 4 of Tomcat container supports servlets 2.3 and JSP 1.2.

Installing Tomcat itself is relatively easy; the only prerequisite is to have the Java SDK. After having downloaded and executed the corresponding installer (Windows platform for this system) from the Tomcat pages at Apache [H], there is a need to set

[16] Google – http://www.google.com

the JAVA_HOME environment variable to point to the Java installation folder and configure the directory structure of the development environment. A really good tutorial about how to configure and use Tomcat server 4 can be found at http://www.moreservlets.com/Using-Tomcat-4.html.

Integrating Tomcat as a plugin within the regular Apache server (or other commercial Web server) to construct a deployment scenario is a little bit more complicated so it will not be described here (see http://jakarta.apache.org/tomcat/tomcat-4.0-doc/ for further detail).

Concluding the explanation of this tool, it should be noted that in the implementation of the prototype Tomcat 4 has been used as a standalone server. Therefore, no testing on concurrency can be applied to this system.

The next figure shows the structure of the prototype (called **whprototype**) in the Tomcat Server directories tree.

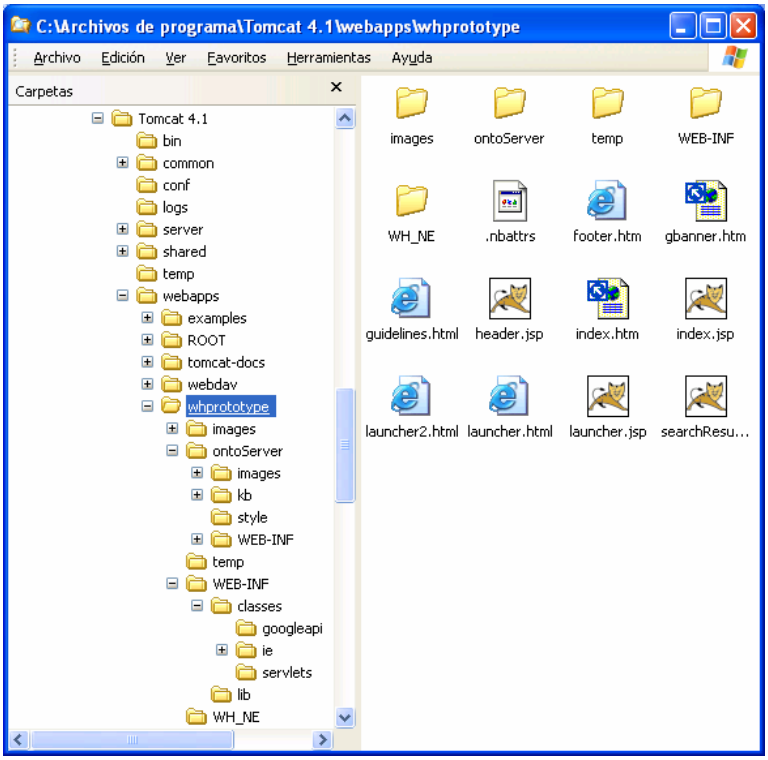


Figure 8.3 – Tree structure of the system on the Web Server

8.2.5 NetBeans IDE 3.5.1

NetBeans is the programming IDE (Integrated Developed Environment) used to edit, compile and debug the Java source code of the application. It is one of the most popular Java technology development environments.

Version 3.5.1 was used along the system implementation but at the time this document is being written NetBeans IDE 3.6 Platform is already available, supporting applications from mobile devices to multi-tier enterprise systems.

NetBeans is another open source project, and its designers define it as follow in the projects home site [G]: *"The NetBeans IDE is a development environment - a tool for programmers to write, compile, debug and deploy programs. It is written in Java - but can support any programming language. It is a free product with no restrictions on how it can be used."*

8.3 Notes about this chapter

This chapter serves as an introduction to the system implementation strategy and the chosen software tools. Code specifications and explanations about libraries, classes methods etc. have not been given for being out of the scope of this section. The reader can refer to the separate document (called *SourceCode.doc*) that contains this information.

After implementing the system prototype some tests are performed. The testing process is presented in the next chapter.

Part IV

SYSTEM TEST & CONCLUSION

Chapter 9: System Testing

This chapter presents the test cases designed for this system and the conclusions obtained from executing these tests. Part of the testing of the system was performed during the implementation phase using the GUIs of the development tools themselves.

9.1 Testing Approach

Testing is a practice where a software system (and its components) is evaluated by means of manual or automatic procedures. The objective of testing is to verify that the system meets its specifications and to detect differences between the expected results and the obtained ones.

Some of the testing objectives are:

- To test what the system should do.
- To test what the system should not do.
- Try to find errors; the aim of the testing procedures is actually this.
- A test is successful if it finds errors. The lack of errors can be due to inadequate testing cases or to a good quality of software (usually the first one).

There are two very common techniques used in software testing: black box testing and white box testing.

- **Black box tests**. These tests are designed without knowing the internal program structure, but knowing the task that the application should accomplish. The objective is to verify that the program really meets its required specifications. "Verify that the system does what is supposed to do".
- **White box tests.** These tests are designed knowing the internal structure of the program. Their objective is to verify that each of the logic sequences in the program is 100% correct. "Verify how good this solution performs it functionality".

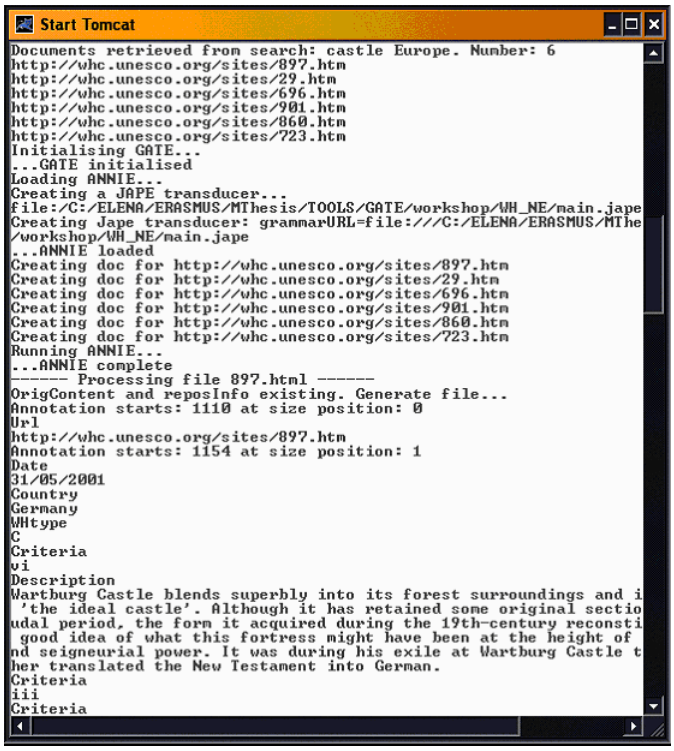


Figure 9.1 – Trace in Tomcat Web Server (negative image)

In the execution of the **white box tests** some of the development tools were very helpful, like the debugger provided with *NetBeans IDE* environment or the trace window in the *Tomcat Server* (see figure 9.1). Some of the design tools contributed as well in these testing tasks, like the editor of web pages *FrontPage*.

To be able to keep track of the correctness of the internal logic of every method and every class, some "extra" lines where introduced inside the Java code executed on the server side. This code remains in the original source files and can be commented or uncommented to be aware of the processes being running at all times. Before having the code running in the development server some of the Java routines were first executed and tested as standalone programs. This way the application code is split in different components and tested separately. Later on, all the different parts were integrated and tested as a whole block.

The errors found were weighed up and allocated into three groups: *serious bugs*, *bugs for a second review* and a third group that are considered not as errors but as *improvements* to the system. The serious bugs were corrected; some of the bugs for a second review where also fixed but not all of them. Not all were fixed due to a lack of time and due to the character of this project, being a prototype only the basic functionality is required. For the same reason potential improvements found while testing were not implemented.

No more details about the **white box test** will be given in this chapter. In the next section the test cases designed for this system will be presented. Only **black box tests** cases have been designed.

Before moving on to the design of the test suite to mention that these tests will be executed over a system with the same features:

CPU: mobile AMD Athlon™ XP, 1600+ MHz
Platform: Microsoft Windows XP, Version 2002
Memory: 496 MB RAM

9.2 Test Suite Design

Since the system is made up of several different components (information retrieval, information extraction, web site presentation and so on) to prepare case tests for the whole application is a laborious task. That is why the testing will be divided into two main blocks: one concerns web site evaluation (Block A) and the other one concerns knowledge extraction and data storage evaluation (Block B). No further explanation seems required.

9.2.1 Web site Evaluation (Block A)

TEST No A1

Description:
Test the internal and external hyperlinks of the prototype's website.

Expected behaviour:
All hyperlinks should be working properly (leading where they are supposed to lead). In addition, internal hyperlinks should open in the main frame of the frameset while external links should open in a separate browser window. There is an exception to this behaviour, those pages that show information about a WH site should be presented to the user in a separate window as well.

TEST No A2

Description:
Test that images that are hyperlinks provide the user with a tip about their destination.

Expected behaviour:
All images that work as hyperlinks (usually logos in this prototype) should provide with a short descriptive tip when the mouse pointer is moved over the image.

TEST No A3

Description:
Test all the aspects concerning the usability of the website.

Expected behaviour:
The website should have a visual continuity, should be easy to use and understand (friendly user interface) and should provide help to the user when necessary.

TEST No A4

Description:
Test that the website allows the user to perform requests.

Expected behaviour:
The website should provide the user with a web page where he can type his request to the system in the shape of keywords. This web page should be enough explanatory for itself

TEST No A5

Description:

Test that some kind of input validation is made on the client side prior to submit a query.

Expected behaviour:
The search launcher page of the website should control that the input text box has a value before submitting the query to the application server. If the text box is empty the system should inform the user about this fact and place the cursor in the input text box.

TEST No A6

Description:
Test that the search engine is working on the client side.

Expected behaviour:
After the user has typed the query and clicked the search button, he should be redirected to another page showing the results of his request. The results should be presented in a list containing the names of the WH sites in the shape of hyperlinks that lead the user to the information about each.

TEST No A7

Description:
Test that the user gets some kind of feedback from the web site.

Expected behaviour:
When the search results page is presented to the user some feedback information should be given to the user along with the results. The query string that he entered should be given back to him along with the number of matches found.

TEST No A8

Description:
Test that there is a certain level of error control.

Expected behaviour:
If an error occurs in the server side, the application should detect it and inform the user in an “understandable way”. The user should never get cryptic error windows.

TEST No A9

Description:
Test the response time

Expected behaviour:
The response time that the user “can feel” should be as small as possible.

The case tests designed for the web site were executed over two platforms: Internet Explorer 6 and Netscape Navigator 7.1.

NOTE1: The external functionality of the Ontology Server was not considered for this test.
NOTE2: As it was mentioned in previous chapters a test on the concurrency can not be applied to this system since it is running locally in a single machine.

9.2.2 Knowledge Extraction & Data Storage Evaluation (Block B)

TEST No B1

Description:
Test the connection with the Google Web APIs.

Expected behaviour:
This connection should be working properly unless there is an external failure in the internet connexion or the number of queries per day has been exceeded. In that case the IR system should control the exceptions and generate some sort of informative message.

TEST No B2

Description:
Test that the IR system harvests relevant documents from the working domain.

Expected behaviour:
The documents retrieved by the IR system should be inside the domain of the World Heritage Centre website. Moreover, they should match the query of the user with a high degree of accuracy.

TEST No B3

Description:
Test that the IE system is able to properly annotate the web pages received as input from the IR system.

Expected behaviour:
The IE system should create a corpus with the input files and execute the necessary processing resources (PR) and the JAPE grammar over them. The fragments of extracted information should be: name of a site, country, year or inscription, criteria for selection, description and original URL.

TEST No B4

Description:
Test that the system generates XML documents for each of the files fed to the IR component, containing the annotations done.

Expected behaviour:
For each of the files that are automatically fed by the IR component to the IE component, the system should generate a XML file containing the information plus the annotations made over them.

TEST No B5

Description:
Test that the system generates a XHTML document for each of the files fed to the IE component, containing the annotation highlighted.

Expected behaviour:

For each of the files that are automatically fed to the IE component, the system should generate a XHTML file containing the original text and some extra tags to highlight the annotations done in the page.

TEST No B6

Description:
Test that the system stores correctly both XML and XHTML documents generated for each of the files fed to the IE.

Expected behaviour:
The system should store the two files generated for every input document in the folder named *Temp* inside the web server structure. It should check the correctness of the names of the files.

TEST No B7

Description:
Check the semantic correctness of the annotations done over the web pages.

Expected behaviour:
The annotation should correspond with concepts of the ontology that defines the domain.

TEST No B8

Description:
Test that the annotations perform by the annotation tool are the same as the annotations in the XHTML documents.

Expected behaviour:
The annotation that GATE performs over the corpus of document should match the ones that the user get as a final result in the browser.

TEST No B9

Description:
Test all the possible operators (-, OR, “”, ~) in the input query strings. Check the same queries (examples mentioned in section 7.2.3) contrasting the results with the ones obtained in the real Google search engine.

Expected behaviour:
The results of the Information Retrieval process perform with these queries should be as expected from the standard behaviour of these operators.

TEST No B10

Description:
Test the answering times for the processes of extraction and output generation.

Expected behaviour:
The answering time should be as small as possible.

Some of these tests were executed using both GATE's GUI and GATE's API. For the testing in the GUI sets of documents locally stored and sets of online documents were used as input. For the testing using the API the requests executed against the system were the ones sketched in section 2.2.2 plus some other queries that have been mentioned along this document.

9.3 Test Evaluation

From the test cases presented in the previous sections the following conclusions can be drawn:

The application has the basic desirable behaviour but, in general, it works a little bit slowly in the testing environment and in some cases is not as accurate as desirable while recognising fragments of the text. It provides with a basic control of errors (see the following figure), but still there are some errors that could be controlled.

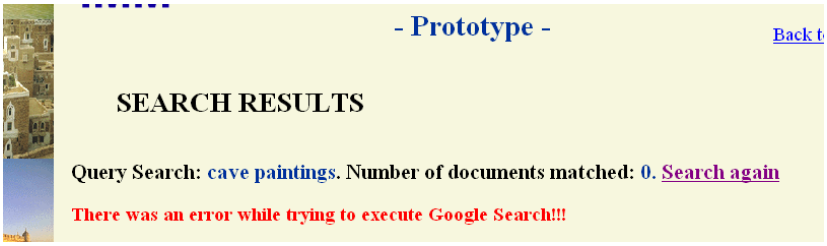


Figure 9.2 – Handling error message (Prototype screenshot)

Concerning the test done over the web site the main problem found was that as the user does not get any feedback until the search result page appears (this could be considered as an improvement in a second stage of the prototype) he could easily be tempted to move to another page. If the user does that the results will be lost and to see them again he will have to make a new search.

Concerning the extraction of knowledge some facts came out after the testing:

- Sometimes the annotations done via the GATE GUI are not the same as the ones done via the API (and so the user cannot perceive some of the annotations although they were successfully perform by the ANNIE system and grammar). This is due to the problem of crossover tags that appears in GATE when using the option of preserve the format of a document (this option is used by this system through the API). This option is implemented in GATE with the cost of losing certain annotations because what it does is trying to restore the original markup and where possible add the annotations produced by GATE. If is not possible it just overrides the annotations in favour of the old ones.
- There are some words that used as keywords cause a lot of ambiguity, for instance the word "place". It can be used in many senses but probably if the user types this word as part of his search what he is trying to find is WH sites with "an open square lined with houses of a similar type in a city or town". Instead what he gets is a set of WH sites of different kind containing sentences like *"...being the only place on earth where rocks...", "... still today a place of*

pilmigrage and devotion" or *"the construction of this Gothic masterpiece took place in several stages…"*. Far from the purpose of his initial request. Therefore, this kind of multi-meaning words should be controlled somehow.

- The "tower problem" is still present to some extent. Further researched should be done in the field of word sense disambiguation to distinguish nouns and verbs. It is suggested at this point the use of a general-purpose lexical database such as *WordNet* (www.cogsci.princeton.edu/~wn) coupled with GATE and the ontology.
- The JAPE grammar is not as accurate as desirable in some of test cases, although in general terms it works fine.

From this testing phase it can be concluded that the system works fine in general but that for certain specific cases (ambiguity, general meaning words…) needs some improvement and further research to include other software tools and extend the scope of the system.

Chapter 10: Conclusions

Finally comes the moment to draw conclusions from the development of this master dissertation. In the first place some general conclusions reached from the theoretical and practical issues discussed along this study will be presented. Then the personal achievements will be mentioned and finally some ideas for future work that can be started from the point where this master thesis has stopped.

This master thesis project ends with this chapter. Some extra information can be found in the appendices part (Part IV). But before that some other useful information is included, like the references, glossary and list of acronyms sections.

10.1 General conclusions

This master thesis tried to deal with the problem of ontology-driven data extraction from online html documents. During this study the basic architecture of the system and the initial prototype have been designed and described. This initial prototype, in this first stage of development, is incomplete and should be extended with the rest of the components identified in the architecture. Also some improvements and optimizations to the system could be suggested.

This study shows that it is possible to exploit and use automatically the data presented in some web pages. However, some manual preparation is first required. This manual step is always necessary if one's intention is to extract reliable data.
During the several months of this project the website that has been used as a working domain did not change their structure or contents. The pages that have been used along this work are static. However, just a week before the expiration of this thesis's deadline, we have been informed that the entire content and structure of the *World Heritage Centre web site* is expected to change since now this organization wants to follow a dynamic database-driven design for the website. This confirms the fact that very few pages remain static in the WWW.

How this new website is going to be remains unknown, therefore how the changes will affect this system is a question that can not be evaluated right now. However it is probable that with some slight changes to the grammar and some modifications in the Google search options the system can be easily adapted to the new situation. Despite this last-minute news about the future changes in the working domain, it is still believed that the approach followed by this system (using an ontology to drive the extraction process coupled with entity recognition and pattern matching techniques) is appropriate for extracting data from the Web.

10.2 Personal achievements and conclusions

The main personal achievement gained with the development of this master thesis has been the research experience, which I consider very positive for my general education. Until now I used to see myself as a developer more than a researcher, and with this work I have learned how to do research in new concepts. Particularly, I have found very appealing the concepts of semantic web and IE techniques as well as everything concerning the ontology's world. These concepts were totally unknown to me.
Another personal achievement has been to learn Java and servlets technologies, which were also totally new to me. And yet I feel that I still have a lot to learn about them.

During the research process I have found very interesting and incredibly useful the concept of open source development and research communities. There are plenty of discussion lists where one can ask for advice to people that are experts in a field where you are just a newbie. Your doubts are seriously taken into account by people that may be direct part of the team developing the technology.

Concerning the final result obtained in this project I must say that I would have liked to successfully achieve a better strategy for storing the information extracted (populating a Protégé knowledge base as it was the desired aim) but that was not possible due to the lacks of the tools used as well as the lack of time. Therefore, some of the requirements could not be met.

In retrospect this project was probably too ambitious to be carried out in just six months. Only to acquire all the background necessary and make the problem analysis took almost half of the time scheduled. Not to mention the resources spent in getting to know the techniques and tools, dealing with technical problems and the additional difficulty of making it work in the WWW.
However, the system developed is able to get a request from a user and (with only that input) launch all the processing in the server (retrieving and filtering relevant web pages, annotating fragments from them by means of an ontology and a grammar, extracting the critical information… and so on) until presenting the results in the web browser. It also stores the knowledge acquired in the shape of server side annotated web pages. The system prototype may lack of some functionality and may need some improvement but at least completes the cycle from the user input to the system output. Therefore, showing that is possible to develop a basic system that extracts information from online documents using an underlying ontology and combining new generation

software tools and technologies. In that sense, it can be considered that the goals were achieved.

About the experience of working with tools to develop language processing (LP) systems, I believe that at this stage in the progress of this field, no one should really have to write code to, for example, view the output of a syntax analyzer or split sentences of a text, or even have to do significant work to get an existing viewing tool to process their data. Many common language processing tasks have been solved to an acceptable degree by previous work and should be reused. Such reuse is still less common than it should be, often because of installation and integration problems that have to be solved again in each case.
These tools are here to help us by delegating common tasks to them and providing with a store of reusable resources; trying to start from scratch would be, in my opinion, a step back and a waste of time. Moreover, we have the chance to contribute with this development since most of these tools are open-source.

To conclude, simply state that I learnt from experience that building a system using relatively new technologies has some pros and cons. On one hand, it is very interesting and exciting to be in the "state of the art" of new concepts and technologies. On the other hand, it is a little bit risky (and more in commercial system) to use tools that still are under constant development because that increases the probability of failure. But I believe that once the specifications become definitive a lot of useful applications will probably emerge from them.

10.3 Future work

This section points out some of the improvements that can be done to the current system as well as some of the future research work that can be accomplished having this thesis as an inspiration or starting point.

Within the first group of improvements it can be found:

- To improve the current UI with some extra functionality like for instance spelling suggestions (if the user types a keyword with mistakes) or a more detailed help.
- To provide the user with a more complex interface for the search launcher with which he could add more options to his request.
- To allow the search in French. This is the only language with which the system can be extended because the working domain of the *World Heritage Centre* [A] is only available in English and French. However, some drastic adjustments will be required in the IE subsystem.
- To weigh up the possibility of extracting more concepts within the description field of a WH property. Together with this would come some changes in the ontology and also a possible new task: to cope with the extraction of

relationships among concepts of the ontology. This is however not seen as a necessary task because of the nature of the domain.

- To make the IR subsystem more accurate. By using other methods rather than the beta Google Web APIs software to retrieve documents. But specially by applying some techniques to add "semantics" also to the search and therefore solve issues like the "tower problem" (among others). This last issue would require a further investigation, which could easily be itself the topic for a new master dissertation.

This last point closely links with the future research work that can be done from this study and on. Some suggestions of new fields to research are:

- The automatic extraction of images. This issue will require specific techniques and treatment and would mean an extra burden to the IE system.
- A different way of attacking the same problem could have been considered, for instance a wrapper for the WH domain could have been built. The hand-code wrapper technique is quite reliable but it has several disadvantages: it is time consuming and prone to error, and if the site changes the wrapper has to be rewritten. Nearly all wrappers today are constructed by hand. A new research field to solve the problems of classical hand-code wrappers would be to automatically construct a wrapper (automatic programming is underlying).
- Some research in the field of automated text summarization can also be applied to this system. To summarize text means to render it in a readily comprehensible format for humans (whereas the output of an IE system is usually in a machine readable form to be entered in a database for future access or analysis).

  Actually the information provided in the official web site is quite limited, gathering and merging information from more sources would consolidated a very rich World Heritage repository. This would mean to work with several sites and combine several techniques like information extraction and natural language generation [14] to support user-directed multi document summarization. Very little research has been made to explore the potential of merging summarization and IE techniques. A point to start could be at http://www.summarization.com.

As seen before, there are several issues in this thesis that could be a subject for further work. Some cases have not been considered or prioritized during the development for being beyond the scope of this work and the time given.

Taking a look at the horizon, it can be forecasted that information on the Web will turn into one huge knowledge base: the Semantic Web. It is now the right time to get involved in this process.

References

[1] Tim Berners-Lee (August 1996). "The World Wide Web: Past, Present and Future". http://www.w3.org/People/Berners-Lee/1996/ppf.html

[2] Tim Berners-Lee, James Hendler, Ora Lassila. "The Semantic Web", *Scientific American,* May 2001. http://www.w3.org/2001/sw/

[3] Robert Gaizauskas, Alexander M. Robertson (1997). "Coupling Information Retrieval and Information Extraction: A New Text Technology fro Gathering Information from the Web*". Department of Computer Science, University of Sheffield, UK*.

[4] Alani, H., Kim, S., Millard, D., Weal, M., Hall, W., Lewis, P. and Shadbolt, N. (2003) "Automatic Ontology-Based Knowledge Extraction and Tailored Biography Generation from the Web". *Intelligence, Agents, Multimedia Group University of Southampton, UK*.

[5] Alani, H., Kim, S., Millard, D., Weal, M., Hall, W., Lewis, P. and Shadbolt, N. (2003) "Automatic Ontology-Based Knowledge Extraction from Web Documents". *University of Southampton*.

[6] Alani, H., Kim, S., Millard, D., Weal, M., Lewis, P., Hall, W. and Shadbolt, N. (2003) "Automatic Extraction of Knowledge from Web Documents*". I.A.M. Group, ECS Dept. University of Southampton, UK*.

[7] Xiaoying Gao and Leon Sterling, "Semi-Structured Data Extraction from Heterogeneous Sources*", Intelligent Agent Laboratory Department of Computer Science and Software Engineering. The University of Melbourne, Australia.*

[8] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, R. D. Smith, "Conceptual-model-based data extraction from multiple-record Web pages". *Data Extraction Group, Brigham Young University, Provo, Utah, USA*.

[9] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira, "A Brief Survey of Web Data Extraction Tools". *Department of Computer Science Federal University of Minas Gerais Belo Horizonte MG Brazil.*

[10] Natalya F. Noy and Deborah L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford University, Stanford.

[11] Kalina Bontcheva, Hamish Cunningham, Valentin Tablan, Diana Maynard, Oana Hamza. "Using GATE as an Enviroment for Teaching NLP". *Department of Computer Science, University of Sheffield*.

[12] Vargas-Vera, M., E. Motta, J. Domingue, M. Lanzoni, A. Stutt and F. Ciravegna (2002). "MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup". *13th Int. Conf on Knowledge Engineering and Management (EKAW 02), Spain*.

[13] Handschuh, S., Staab, S., and Ciravegna (2002), F. "S-CREAM – Semi Automatic Creation of Metadata". *Semantic Authoring, Annotation and Markup Workshop, 15th European Conf. on Artificial Intelligence, Lyon, France*.

[14] Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff (2001). "Multidocument Summarization via Information Extraction*". CoGen Tex, Inc & Department of Computer Science, Cornell University, Ithaca, NY.*

[15] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein. "Annotated DAML+OIL Ontology Markup". *W3C Note 18 December 2001*, http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/

[16] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein. "DAML+OIL (March 2001) Reference Description". *W3C Note 18 December 2001*, http://www.w3.org/TR/daml+oil-reference.

[17] Nicola Guarino (1998). "Formal Ontology and Information Systems". *N*ationa*l Research Council, LADSEB-CNR, Padova, Italy* (pages 7-11)

[18] Mark Dutra. "Ontologies for Web Services". Sandpiper Software, Inc.

[19] Michael Denny (2002). "Ontology Building: A Survey of Editing Tools". Published on XML.com, http://www.xml.com/pub/a/2002/11/06/ontologies.html

[20] Holger Knublauch (20 June 2003). "An AI tool for the real world. Knowledge modelling with Protege". *Article published in JavaWorld.com web site*, http://www.javaworld.com/javaworld/jw-06-2003/jw-0620-protege.html.

[21] H. Cunningham, K. Bontcheva, D. Maynard, V. Tablan. "GATE - A New Release". *ELSNews*, 11(1), 2002. (pages 3-4)

[22] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu, Marin Dimitrov (2001-2002). "Developing Language Processing Components with GATE (a User Guide)". ©*TheUniversity of Sheffield.*

[23] Jon Udell (9 July 2003). "The Document is the Database". Published on XML.com. at http://www.xml.com/pub/a/2003/07/09/udell.html

Other resources:

[A] **UNESCO World Heritage Centre.** http://whc.unesco.org/

[B] **World Heritage Explorer Prototype.** http://www.vrheritage.org/engine/explorer

[C] **Protégé 2000.** http://protege.stanford.edu/

[D] **GATE (General Architecture for Text Engineering).** http://gate.ac.uk

[E] **The Artequakt Project.** http://www.artequakt.ecs.soton.ac.uk

[F] **Java Technology.** http://java.sun.com

[G] **NetBeans Project.** http://www.netbeans.org

[H] **The Apache Jakarta Project.** http://jakarta.apache.org/

[I] **World Wide Web Consortium.** http://www.w3.org

[J] **Google Web APIs home (beta).** http://www.google.com/apis/

[K] **Webopedia.** http://www.webopedia.com

[L] **Whatis?com.** http://whatis.techtarget.com/

[M] **FOLDOC**. http://wombat.doc.ic.ac.uk/foldoc/index.html

[N] **Die.net online dictionary.** http://dict.die.net/

[O] **SmartDraw:** http://www.smartdraw.com

Glossary

Agent On the Internet, an agent (also called an intelligent agent) is a program that gathers information or performs some other service without your immediate presence and on some regular schedule. Typically, an agent program, using parameters you have provided, searches all or some part of the Internet, gathers information you're interested in, and presents it to you on a daily or other periodic basis. An agent is sometimes called a *bot* (short for robot).

Browsing Finding your way around Internet by navigating hypertext documents. Browsing is often used to mean the same as surfing.

Business logic "Business logic" is just a fancy way of saying "code." More precisely, in a 3-tier architecture, business logic is any code that is not specifically related to storing and retrieving data (that's "data storage code"), or to formatting data for display to the user (that's "presentation logic"). It makes sense, for many reasons, to store this business logic in separate objects; the middle tier comprises these objects. However, the divisions between the three layers are often blurry, and business logic is more of an ideal than a reality in most programs. The main point of the term is, you want somewhere to store the logic and "business rules" of your application, while keeping the division between tiers clear and clean.

Cardinality In an ER diagram specifies how many instances of an entity relate to one instance of another entity. See definition of *Ordinality* in this glossary.

Case sensitive Describes the ability to distinguish between uppercase (capital) and lowercase (small) letters.

CORBA Short for *C*ommon *O*bject *R*equest *B*roker *A*rchitecture, an architecture that enables pieces of programs, called objects, to

communicate with one another regardless of what programming language they were written in or what operating system they're running on. CORBA was developed by an industry consortium known as the Object Management Group (OMG).

Corpus All the documents in the domain of interest.

Flag A variable or quantity that can take on one of two values; a bit, particularly one that is used to indicate one of two outcomes or is used to control which of two things is to be done.

Flexibility The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.

Frames A feature supported by most modern *Web browsers* than enables the Web author to divide the browser display area into two or more sections (frames). The contents of each frame are taken from a different Web page. Frames provide great flexibility in designing Web pages.

Gazetteer An alphabetical descriptive list of anything, usually words.

GUI Acronym for Graphical User Interface. Allows users to navigate and interact with information on their computer screen by using a mouse, instead of typing in words. The WWW is an example of a GUI designed to enhance navigation of the Internet, once done exclusively via terminal-based (typed command line) functions

HTML Short for *H*yper*T*ext *M*arkup *L*anguage, the authoring language used to create documents on the WWW.
HTML defines the structure and layout of a Web document by using a variety of tags and attributes.
There are hundreds of tags used to format and layout the information in a Web page. Tags are also used to specify hypertext links.

Jsp Short for *J*ava *S*erver *P*age. A server-side technology, Java Server Pages are an extension to the Java servlet technology that was developed by Sun.
JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements -- the actual design and display of the page -- to help make the HTML more functional.

Knowledge base In general, a knowledge base is a centralized repository for information. In relation to Information technology (IT), a knowledge base is a store of knowledge about a particular domain represented in machine-processable form, which may be rules, facts

or other representations.

Ontology The word ontology refers to two things:
A study of the subject of the categories of things that exist or may exist in some domain. Thus ontology is the study of categories.
The product of such a study is called an ontology.
The product of an ontological study will as a minimum come up with a type hierarchy. It may also come up with a relation hierarchy, as is the case in conceptual graph-theory. These two combined will be called an ontology.

Ordinality Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships. When the minimum number is zero, the relationship is usually called optional and when the minimum number is one or more, the relationship is usually called mandatory.

Pipeline A sequence of functional units which performs a task in several steps. Each functional unit takes inputs and produces outputs which are stored in its output buffer. One stage's output buffer is the next stage's input buffer. This arrangement allows all the stages to work in parallel. Pipelines may be synchronous or asynchronous.

Portability The ease with which a system or component can run or be transferred from one environment to another

Precision The percentage correct of instances reported as positive. See also *recall*.

Query A user's (or agent's) request for information, generally as a formal request to a database or search engine. SQL is the most common database query language.

Recall The percentage of positive instances that are identified by the system. See also *precision*.

Regular expression A regular expression (sometimes abbreviated to "regex") is a way for a computer user or programmer to express how a computer program should look for a specified pattern in a text and then what the program is to do when each pattern match is found.

Scalability The ease with which a system or component can be modified to fit the problem area.

Semantic Web "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling*

computers and people to work in cooperation" (article "The Semantic Web", Berners-Lee et al.).

Semistructured Data Is data that has some structure, but it may be irregular and incomplete and does not necessarily conform to a fixed schema.

Serialization The conversion of an object instance to a data stream of byte values in order to prepare it for transmission.

Servlet (By analogy with "applet") A Java program that runs as part of a network service, typically an HTTP server and responds to requests from clients.

The most common use for a servlet is to extend a web server by generating web content dynamically. For example, a client may need information from a database; a servlet can be written that receives the request, gets and processes the data as needed by the client and then returns the result to the client.

Splash image It is the first image that appears in the screen on the first and subsequent launches of an application. Splash images are used to promote a product and usually are only visible for a few seconds while a program is loading.

SQL Abbreviation of structured query language. SQL is a standardized query language for requesting information from a database. The original version called SEQUEL (structured English query language) was designed by an IBM research centre in 1974 and 1975. SQL was first introduced as a commercial database system in 1979 by Oracle Corporation.

Surfing To move from place to place on the Internet searching for topics of interest. The term surfing is generally used to describe a rather undirected type of Web browsing in which the user jumps from page to page rather whimsically, as opposed to specifically searching for specific information.

Text summarization Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user and task (or tasks).

Unicode Officially called the Unicode Worldwide Character Standard, is a standard for representing characters as integers. Unlike ASCII, which uses 7 bits for each character, Unicode uses 16 bits, which means that it can represent more than 65,000 unique characters. This is a bit of overkill for English and Western-European languages, but it is necessary for some other languages, such as Greek, Chinese and Japanese.

Unicode it is a system for "the interchange, processing, and display of the written texts of the diverse languages of the modern world." Some analysts believe that as the software industry becomes increasingly global, Unicode will eventually supplant ASCII as the standard character coding format.

URL Abbreviation of *U*niform *R*esource *L*ocator, the global address of documents and other resources on the World Wide Web.
The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located.

UTF-8 The UTF-8 encoding of Unicode and UCS avoids the problems of fixed-length Unicode encodings because an ASCII file encoded in UTF is exactly same as the original ASCII file and all non-ASCII characters are guaranteed to have the most significant bit set (bit 0x80). This means that normal tools for text searching etc. work as expected. UTF-8 is defined in RFC 2279.

Web browser Client software application that is used to locate and display Web pages.

Web site Collection of network services, primarily HTML documents, that are linked together and that exist on the Web at a particular server. Exploring a website usually begins with the home page, which may lead you to more information about that site. Each site is owned and managed by an individual, company or organization.

WWW (Web) World Wide Web (or simply Web for short) is a term frequently used when referring to "The Internet". WWW has two major meanings:
First, loosely used: the whole constellation of resources that can be accessed using Gopher, FTP, HTTP, telnet, USENET, WAIS and some other tools.
Second, the universe of hypertext servers (HTTP servers), more commonly called "web servers", which are the servers that serve web pages to web browsers.

XML Short for E**x**tensible **M**arkup **L**anguage, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XHTML Short for E**x**tensible **H**ypertext **M**arkup **L**anguage, a hybrid between HTML and XML specifically designed for Net device displays.
XHTML is a markup language written in XML; therefore, it is an

XML application. XHTML uses three XML namespaces, which correspond to three HTML 4.0 DTDs: Strict, Transitional, and Frameset. XHTML markup must conform to the markup standards defined in a HTML DTD.
When applied to Net devices, XHTML must go through a modularization process. This enables XHTML pages to be read by many different platforms.

List of Abbreviations

AI	Artificial Intelligence
ANNIE	A Nearly-New Information Extraction
API	Applications Programmers' Interface
CREOLE	a Collection of REusable Objects for Language Engineering
DARPA	Defense Advanced Research Projects Agency
ERD	Entity Relationship Diagram
GATE	a General Architecture for Text Engineering
GUI	Graphical User Interface
IE	Information Extraction
IR	Information Retrieval
JAPE	Java Annotation Patterns Engine
KB	Knowledge base
LaSIE	the Large-Scale Information Extraction system
LP	Language Processing
LR	Language Resource
NE	Named Entity
NLP	Natural Language Processing
PR	Processing Resource
SALE	Software Architecture for Language Engineering
TIPSTER	not an acronym; the name of a US IE/IR research programme
UNESCO	United Nations Educational Scientific and Cultural Organization
WH	World Heritage
WHC	World Heritage Centre

Part V

APPENDICES

APPENDIX A. World Heritage ER Model & Design of Database Schema

A1. *ER Model: General Concepts*

The Entity-Relationship (ER) model was originally proposed by Peter Chen in 1976 as a way to unify the network and relational database views. Since then, Charles Bachman and James Martin have added some slight refinements to the basic ERD principles.

The basic elements of the ER model are entities, relationships, and attributes. Entities are concepts (real or abstract) about which information is collected. Relationships are associations between the entities and attributes are properties which describe those entities.

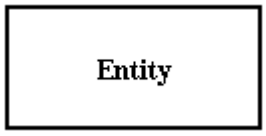
A brief explanation about ER main components is offered below, together with their notation.

A1.1 Entities

Entities are the principal data object about which information is to be collected. They are usually recognizable concepts of the real world, either concrete or abstract.
Entities are classified as independent or dependent (in some methodologies, the terms used are strong and weak respectively). An independent entity is one that does not rely on another for identification. A dependent entity behaves just the opposite; it relies on another one for identification.

The set of all entities of the same type is called an *entity set*. An *entity occurrence* (also known as instance) is an individual occurrence of an entity.

Notation:
Entities are represented by labelled rectangles. The label is the name of the entity. Entity names should be singular nouns.



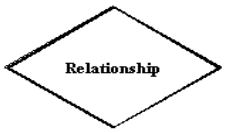
A1.2 Relationships

Relationships represent associations between two or more entities. They are classified by their degree, connectivity, cardinality, direction, type, and existence.

There are many notation styles that express cardinality but basically all of them with the same underlying concepts (Chen, Bachman, Martin etc).

Notation:

Relationships are represented by labelled diamonds.



Lines are used to link entity sets with relationships and entity sets with their attributes. It is possible to use arrows together with the lines to point the direction of the relationship.

A1.3 Attributes

Attributes describe the entity of which they are associated. A particular instance of an attribute is a *value*. The domain of an attribute is the collection of all possible values an attribute can have.

Attributes can be classified as identifiers or descriptors. Identifiers, more commonly called keys, uniquely identify an instance of an entity. A descriptor describes a non-unique characteristic of an entity instance.

Notation:

Attributes are represented by labelled ellipses. Those which are identifiers are underlined. Those which can have more than one value (multivalued attributes) have a double ellipse. Attribute names should be singular nouns.

There is another important construct of the ER modelling that it should be briefly introduced for a subsequent better understanding of the ERD of WH. This construct is the generalization hierarchy.

A1.4 Generalization Hierarchies

A generalization hierarchy is a form of abstraction that specifies that two or more entity sets that share common attributes can be generalized into a higher level entity set called a *supertype* or generic entity. The lower-level entity sets become the *subtypes*, which are dependent entities. A lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

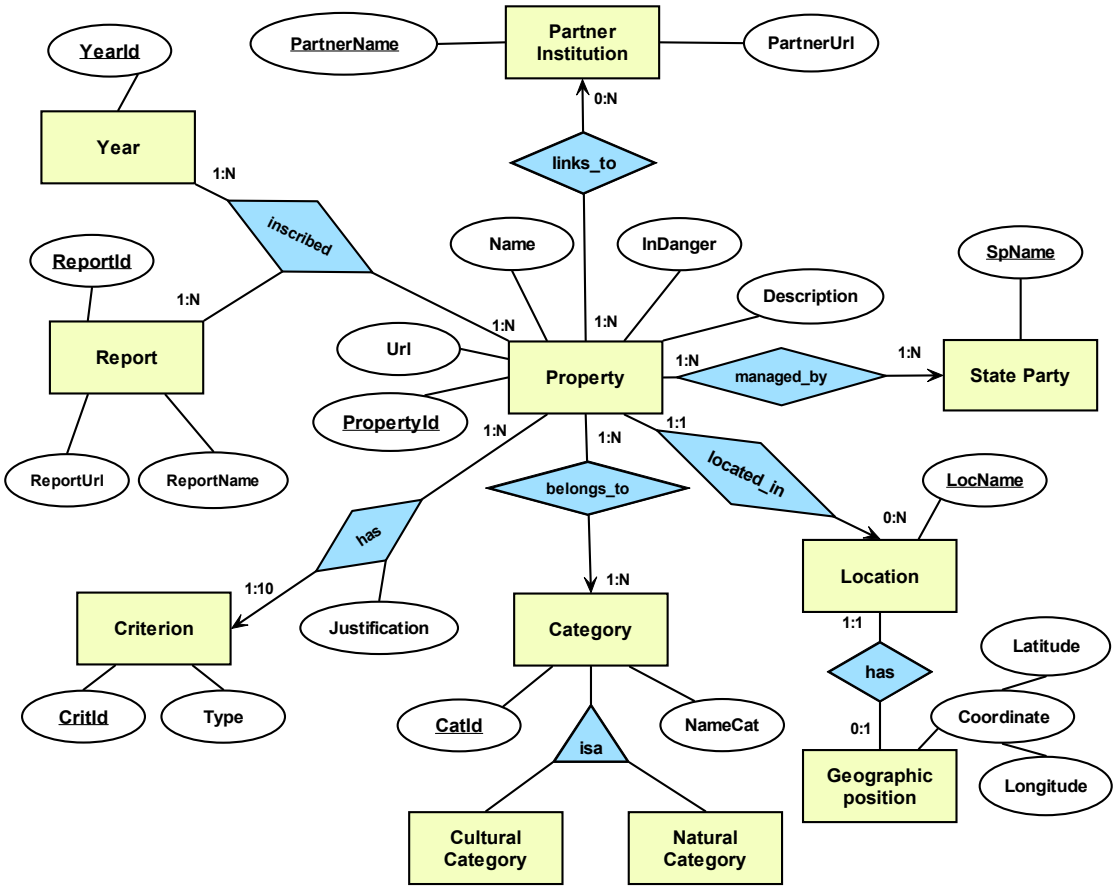
Generalization occurs when two or more entity sets represent categories of the same real-world object. Subtypes can be either mutually exclusive (disjoint) or overlapping (inclusive). A mutually exclusive category is when an entity instance can be in only one category. An overlapping category is when an entity instance may be in two or more subtypes. The completeness constraint requires that all instances of the subtype be represented in the *supertype*.

Generalization hierarchies can be nested. That is, a subtype of one hierarchy can be a supertype of another. The level of nesting is limited only by the constraint of simplicity.

Notation:
This type of hierarchy is depicted in an ERD by a triangle labeled ISA.

A2. *Complete ERD for World Heritage*

Below is the complete Entity Relationship Diagram (ERD) for the World Heritage domain. Chen notation is followed to model cardinalities.



The following section builds a relational database schema based on this ERD.

A3. *Design of the Relational Database Schema*

In the following sections an approach to the relational database schema built from the World Heritage ERD is given. Since this is a technical study further explanations are considered not required.

A3.1 Relation between Entities and Tables

The following table shows the relation between the entities of the ER model and the tables of a relational database.

Entity / Relation	Table
Property State Party	Properties
Year Inscribed	Inscriptions
Category Cultural Category Natural Category	Categories
Belongs_to	BelongsToCat
Criterion	Criteria
Has	HasCriteria
Location Geographic position Located_in	Locations
State Party	State Parties
Managed_by	ManagedBySp
Partner Institution	Partner Institutions
Links_to	LinkToPart

A3.2 Unnormalizations

Entity "Year"

Most of the Properties are inscribed just once. In order to improve the queries, some new attributes will be added to the table "Properties".

- *InscripYear*
  The year in which the site was inscribed for the first time in the World Heritage List.
- *UrlReport*
  Link to the report that justifies the inscription.
- *MultipleIns*
  Boolean flag that indicates whether the site has been inscribed more than once. This attribute should have a default value of FALSE. There will be no need to access the table "Inscriptions" unless the value of this attribute is TRUE.

Entity "Category"

The entities "Category", "Cultural Category" and "Natural Category" are merged into a single table "Categories". This table has two attributes that indicate the type of the

category. Both flags can have a TRUE value at the same time, just like in the "cave" example.
Two flags are also added to the table "Properties":

- *NaturalValue*
  Indicates that the site has a natural value.
- *CulturalValue*
  Indicates that the site has a cultural value.

Both flags can be raised at the same time. They are included in order to improve potential queries in which the type of value of the site is a criterion for the search.

Entity "Criterion"

A flag for each criterion is added in the table "Properties".

- *CulturalCriti, CulturalCritii, CulturalCritiii,…*
  Flags for the cultural criteria.
- *NaturalCriti, NaturalCritii, NaturalCritiii,…*
  Flags for the natural criteria.

Entity "Location"

The entities "Location" and "Geographic Position" are merged into a single table. The attribute "Coordinate" is split into "Longitude" and "Latitude". Both of them are optional.

Most of the Properties have just one location. Some new attributes are added to the table "Properties".

- *MultipleLoc*
  Boolean flag that indicates whether the site has multiple locations.
- *LocName, Latitude, Longitude*
  First location and coordinates of the site. They are optional.

Entity "State Party"

Most of the properties are managed by just one State Party or country. The name of the first country is included in the table "Properties", as well as a flag "MultipleSp" that indicates whether it is related to more than one country. If the flag is raised, it will be necessary to access the table "State Parties".

A3.3 Tables

The following diagram shows the tables with their primary keys (notation @) and foreign keys (notation underlined) together with the relationships among them. After that each table is presented.

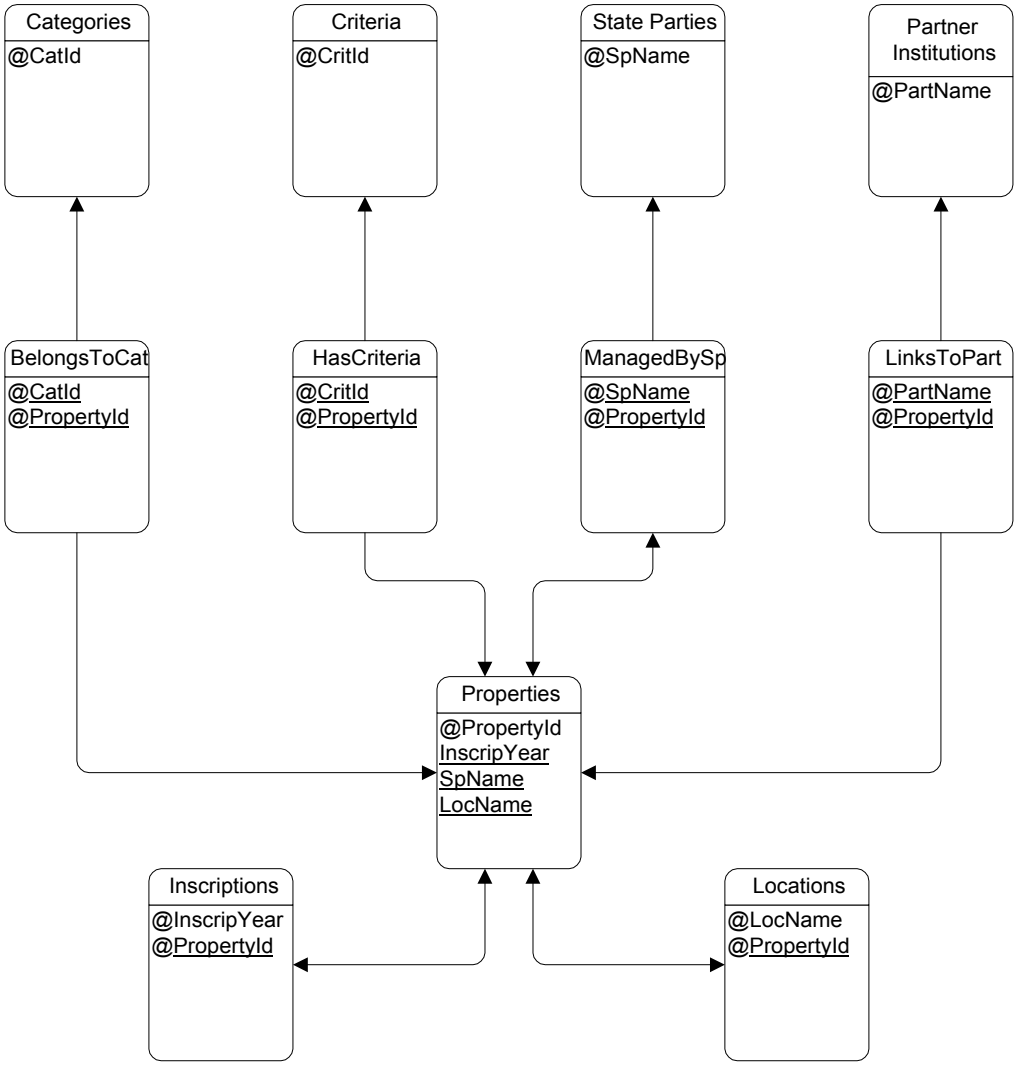


Table "Properties"

This is the main table. It contains the information of the World Heritage properties.

Field	Type	Opt	Description
@PropertyId	Integer		Number given to a property when it is inscribed in the *World Heritage List*.
Name	Character (100)		Name of the site.
Description	Character (500)		A short text describing its outstanding universal value.
Url	Character (100)		URL address of its web page.
InDanger	Boolean		It indicates whether the site is in danger or not.
InscripYear	Integer		Year in which the site was inscribed for the first time.
UrlReport	Character (100)		Link to the report that justifies the first inscription.
MultipleIns	Boolean		Flag that indicates whether the site has

			been inscribed more than once.
NaturalValue	Boolean		Flag that indicates that the site has a natural value.
CulturalValue	Boolean		Flag that indicates that the site has a cultural value.
CulturalCriti	Boolean		Flag that indicates that the site meets the cultural criterion i.
CulturalCritii	Boolean		Flag that indicates that the site meets the cultural criterion ii.
CulturalCritiii	Boolean		Flag that indicates that the site meets the cultural criterion iii.
CulturalCritiiv	Boolean		Flag that indicates that the site meets the cultural criterion iv.
CulturalCritv	Boolean		Flag that indicates that the site meets the cultural criterion v.
CulturalCritvi	Boolean		Flag that indicates that the site meets the cultural criterion vi.
NaturalCriti	Boolean		Flag that indicates that the site meets the natural criterion i.
NaturalCritii	Boolean		Flag that indicates that the site meets the natural criterion ii.
NaturalCritiii	Boolean		Flag that indicates that the site meets the natural criterion iii.
NaturalCritiv	Boolean		Flag that indicates that the site meets the natural criterion iv.
MultipleLoc	Boolean		Flag that indicates whether the site has multiple locations.
LocName	Character (100)	√	First location of the site.
Latitude	Character (30)	√	Coordinates of the site. Latitude.
Longitude	Character (30)	√	Coordinates of the site. Longitude.
SpName	Character (50)		Name of the first country that manages the site.
MultipleSp	Boolean		Flag that indicates whether the site is managed by more than one country.

Table “Inscriptions”

Table with information about the year and justification of each inscription of a Property in the World Heritage list.

Field	Type	Opt	Description
@InscripYear	Integer		Year of inscription.
@PropertyId	Integer		Property inscribed.
UrlReport	Character (100)		Link to the report that justifies the inscription.

Table "Categories"

Different categories (cultural or natural) used to classify the properties.

Field	Type	Opt	Description
@CatId	Integer		Internal code for the category.
CatName	Character (50)		Category name.
TypeCultural	Boolean		Indicates whether the category is cultural or not.
TypeNatural	Boolean		Indicates whether the category is natural or not.

Table "BelongsToCat"

Categories assigned to each Property.

Field	Type	Opt	Description
@CatId	Integer		Category.
@PropertyId	Integer		Property.

Table "Criteria"

Criteria for inscription in the World Heritage List.

Field	Type	Opt	Description
@CritId	Integer		Internal code for the criterion.
Number	Character (3)		Values: i, ii, iii, iv, v, vi
Type	Boolean		Type of the criterion. Values: True = Cultural criterion. False = Natural criterion.

Table "HasCriteria"

Criteria assigned to each property.

Field	Type	Opt	Description
@CritId	Integer		Criterion.
@PropertyId	Integer		Property.
Justification	Character (500)	√	Explanation for the inscription.

Table "Locations"

General locations of the World Heritage sites.

Field	Type	Opt	Description
@LocName	Character (100)		Location name of the site.
@PropertyId	Integer		Property.
Latitude	Character (30)	√	Coordinates of the site. Latitude.
Longitude	Character (30)	√	Coordinates of the site. Longitude.

Table "State Party"

Countries that managed one or more properties.

Field	Type	Opt	Description
@SpName	Character (50)		Name of the country.

Table "ManagedBySp"

Relationships between the properties and the State Parties.

Field	Type	Opt	Description
@SpName	Character (50)		Name of the country.
@PropertyId	Integer		Property.

Table "Partner Institutions"

Hypertext links to institutions that hold information about properties.

Field	Type	Opt	Description
@PartName	Character (100)		Name of the institution.
PartnerUrl	Character (100)		Web page URL of the institution.

Table "LinksToPart"

Relationships between the properties and the partner institutions.

Field	Type	Opt	Description
@PartName	Integer		Partner institution.
@PropertyId	Integer		Property.

APPENDIX B. World Heritage Ontology

B1. *DAML+OIL WH Ontology (complete version)*

Below is the complete World Heritage ontology created by Protégé-2000 v1.9 in DAML+OIL.

```
<rdf:RDF
  xmlns:XMLSchema ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml_oil ="http://www.daml.org/2001/03/daml+oil#"
  xmlns:kb ="http://localhost/whprototype/ontoServer/kb#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns ="http://localhost/whprototype/ontoServer/kb#"
  >

<daml_oil:Class rdf:ID="ReligiousConstruction">
  <rdfs:subClassOf rdf:resource="#Construction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Mine">
  <rdfs:subClassOf rdf:resource="#IndustryConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="CentralAmericanCountry">
  <rdfs:subClassOf rdf:resource="#AmericanStateParty"/>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="parnership_with">
  <rdfs:comment>Links with partner institutions. It is
optional.</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="location">
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="managed_by">
  <rdfs:comment>Country or countries that manages a World Heritage
Property.</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Garden">
  <rdfs:comment>Garden subcategory. Belongs to cultural category,
Cultural Landscape  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#ArtificialLandscape"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="ArtificialLandscape">
  <rdfs:comment>Landscape designed and created intentionally by
man</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#CulturalLandscape"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="latitude">
```

```
  <rdfs:comment>Latitude for the location of a
Property</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="TownHall">
  <rdfs:subClassOf rdf:resource="#PublicConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Church">
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
</daml_oil:Class>
 <daml_oil:Class rdf:ID="AmericanStateParty">
  <rdfs:subClassOf rdf:resource="#StateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Mansion">
  <rdfs:subClassOf rdf:resource="#ResidentialConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Saltwork">
  <rdfs:subClassOf rdf:resource="#IndustryConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Category">
</daml_oil:Class>

<daml_oil:Class rdf:ID="Synagogue">
  <rdfs:subClassOf rdf:resource="#JudaismConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="description">
  <rdfs:comment>Brief description about a WH Property</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="AfricanStateParty">
  <rdfs:comment>A State Party located in the african
continent.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#StateParty"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="longitude">
  <rdfs:comment>Longitude for the location of a
Property</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Tomb">
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
  <rdfs:subClassOf rdf:resource="#MuslimConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Spa">
```

```
  <rdfs:subClassOf rdf:resource="#HealthConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Minaret">
  <rdfs:subClassOf rdf:resource="#MuslimConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="MuslimConstruction">
  <rdfs:subClassOf rdf:resource="#ReligiousConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="propertyid">
  <rdfs:comment>Unique identifier of a WH site</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="NaturalCategory">
  <rdfs:comment>Natural categories</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Category"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="danger">
  <rdfs:comment>Flag that tells if a Property is also inscribed in
the list of World Heritage sites in danger. It is optional, by
default it is NO.  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Canal">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Chapel">
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="WHontoDAMLOIL_00195">
</daml_oil:ObjectProperty>

<daml_oil:DatatypeProperty rdf:ID="name">
  <rdfs:comment>String containing the name</rdfs:comment>
  <daml_oil:domain rdf:resource="#Category"/>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:domain rdf:resource="#StateParty"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Mill">
  <rdfs:subClassOf rdf:resource="#IndustryConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="criteriaid">
  <rdfs:comment>The identifier of a criterion can only be one of the
following: i, ii, iii, iv, v and vi.  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Criterion"/>
</daml_oil:DatatypeProperty>
```

```
<daml_oil:Class rdf:ID="Abbey">
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="EuropeanStateParty">
  <rdfs:comment>A State Party located in the european
continent.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#StateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="CulturalCategory">
  <rdfs:comment>Cultural categories</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Category"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Fortress">
  <rdfs:subClassOf rdf:resource="#MilitarConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="PublicConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="justification">
  <daml_oil:domain rdf:resource="#Criterion"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Deposit">
  <rdfs:subClassOf rdf:resource="#IndustryConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Mosque">
  <rdfs:subClassOf rdf:resource="#MuslimConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Theater">
  <rdfs:subClassOf rdf:resource="#PublicConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="OceanicStateParty">
  <rdfs:comment>A State Party located in the oceanic
continent.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#StateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="MilitarConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="NorthamericanCountry">
  <rdfs:subClassOf rdf:resource="#AmericanStateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Library">
  <rdfs:subClassOf rdf:resource="#PublicConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Cathedral">
```

```
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Tower">
  <rdfs:subClassOf rdf:resource="#MilitarConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="AsianStateParty">
  <rdfs:comment>A State Party located in Asia</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#StateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Ramp">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="ChristianConstruction">
  <rdfs:subClassOf rdf:resource="#ReligiousConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="SouthAmericanCountry">
  <rdfs:subClassOf rdf:resource="#AmericanStateParty"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Criterion">
  <rdfs:comment>Criterion to be subscribed in the World Heritage
List</rdfs:comment>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="includes_sites">
  <rdfs:comment>Inverse slot for the belongs_to slot in class
Property. States which properties are included in a certain category
</rdfs:comment>
  <daml_oil:domain rdf:resource="#Category"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="criterion_of">
  <rdfs:comment>Inverse of slot has_criteria in class
Property</rdfs:comment>
  <daml_oil:domain rdf:resource="#Criterion"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Bridge">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Hospital">
  <rdfs:subClassOf rdf:resource="#HealthConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="AssociativeLandscape">
  <rdfs:comment>Landscapes with powerful associations (religious,
artistic or cultural) to the natural element rather than cultural
evidence ( which may be insignificant or even absent).
</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#CulturalLandscape"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Sanctuary">
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
```

```
  <rdfs:subClassOf rdf:resource="#HinduConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Harbour">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="SecularConstruction">
  <rdfs:subClassOf rdf:resource="#Construction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="ResidentialConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Park">
  <rdfs:comment>Park subcategory. Belongs to cultural category,
Cultural Landscape  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#ArtificialLandscape"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Aqueduct">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="HinduConstruction">
  <rdfs:subClassOf rdf:resource="#ReligiousConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="PartnerSite">
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="continent">
  <rdfs:comment>Continent which a country belongs to.</rdfs:comment>
  <daml_oil:domain rdf:resource="#StateParty"/>
</daml_oil:DatatypeProperty>

<daml_oil:ObjectProperty rdf:ID="has_criteria">
  <rdfs:comment>Criteria that explain why the WH site is included in
the list. Minimum one  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="Temple">
  <rdfs:subClassOf rdf:resource="#BuddhismConstruction"/>
  <rdfs:subClassOf rdf:resource="#HinduConstruction"/>
  <rdfs:subClassOf rdf:resource="#JudaismConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="University">
  <rdfs:subClassOf rdf:resource="#PublicConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="url">
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="StateParty">
```

```
</daml_oil:Class>

<daml_oil:Class rdf:ID="Railway">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Palace">
  <rdfs:subClassOf rdf:resource="#ResidentialConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="JudaismConstruction">
  <rdfs:subClassOf rdf:resource="#ReligiousConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Residence">
  <rdfs:subClassOf rdf:resource="#ResidentialConstruction"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="categorytype">
  <rdfs:comment>Category type. Could only be or Natural (N) or
Cultural (C).</rdfs:comment>
  <daml_oil:domain rdf:resource="#Category"/>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Wall">
  <rdfs:subClassOf rdf:resource="#MilitarConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="CulturalCriterion">
  <rdfs:comment>Cultural criterions to be subscribed for in the World
Heritage List  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Criterion"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="IndustryConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="WaterMill">
  <rdfs:subClassOf rdf:resource="#Mill"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="BuddhismConstruction">
  <rdfs:subClassOf rdf:resource="#ReligiousConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Construction">
  <rdfs:subClassOf rdf:resource="#CulturalCategory"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Castle">
  <rdfs:subClassOf rdf:resource="#MilitarConstruction"/>
  <rdfs:subClassOf rdf:resource="#ResidentialConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="CulturalLandscape">
  <rdfs:comment>Cultural Landscape (Cultural Category)</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#CulturalCategory"/>
</daml_oil:Class>
```

```
<daml_oil:Class rdf:ID="WindMill">
  <rdfs:subClassOf rdf:resource="#Mill"/>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="belongs_to">
  <rdfs:comment>Relationship that link the Properties with the
Categories they belong to. Multiple  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Property">
  <rdfs:comment>This is an abstract superclass of World Heritage
properties, containing the information common to all of them.
</rdfs:comment>
</daml_oil:Class>

<daml_oil:DatatypeProperty rdf:ID="inscription">
  <rdfs:comment>Year in which the site was incribed in the
list</rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</daml_oil:DatatypeProperty>

<daml_oil:Class rdf:ID="Tunnel">
  <rdfs:subClassOf rdf:resource="#TechnicalConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="HealthConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="NaturalCriterion">
  <rdfs:comment>Natural criterions to be subscribed for in the World
Heritage List  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Criterion"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Monastery">
  <rdfs:subClassOf rdf:resource="#BuddhismConstruction"/>
  <rdfs:subClassOf rdf:resource="#ChristianConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Bath">
  <rdfs:subClassOf rdf:resource="#HealthConstruction"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Ironwork">
  <rdfs:subClassOf rdf:resource="#IndustryConstruction"/>
</daml_oil:Class>

<daml_oil:Ontology rdf:ID="">
</daml_oil:Ontology>
<daml_oil:Class rdf:ID="TechnicalConstruction">
  <rdfs:subClassOf rdf:resource="#SecularConstruction"/>
</daml_oil:Class>

</rdf:RDF>
```

B2. *DAML+OIL WH Ontology (reduced version)*

Below is the reduced version of the World Heritage ontology created to use in GATE.

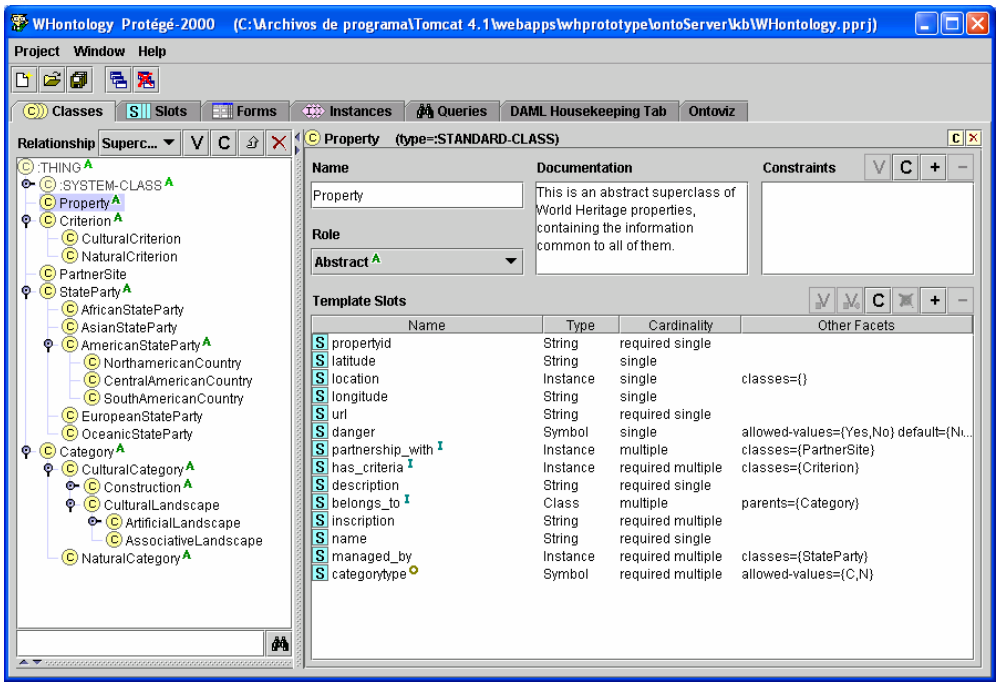
```
<rdf:RDF
  xmlns:kb ="http://localhost/whprototype/ontoServer/kb/"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml_oil ="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns ="http://localhost/whprototype/ontoServer/kb/"
  >

<daml_oil:Class rdf:ID="Criteria">
  <rdfs:comment>Criteria por selection (small version of the
ontology). No attributes possible.  </rdfs:comment>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Description">
  <rdfs:comment>Description of a WH site (small version of the
ontology). No attributes possible  </rdfs:comment>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Location">
  <rdfs:comment>Location of a Property (small version of the
ontology). No attributes possible  </rdfs:comment>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Property">
  <rdfs:comment>World Heritage Property (small version of the
ontology). No attributes possible  </rdfs:comment>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="has_criteria">
  <rdfs:comment>Relationship between a Property and the criteria that
meet</rdfs:comment>
  <daml_oil:range rdf:resource="#Criteria"/>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:ObjectProperty rdf:ID="located_in">
  <rdfs:comment>Relationship bewteen a Property and the geographic
place where is located  </rdfs:comment>
  <daml_oil:range rdf:resource="#Location"/>
  <daml_oil:domain rdf:resource="#Property"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="NaturalCriteria">
  <rdfs:comment>Natural Criteria
(small version of the ontology). No attributes possible
</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Criteria"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Url">
  <rdfs:comment>Url address of the original source file (small
version of the ontology). No attributes possible  </rdfs:comment>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="inscribed_in">
```

```
  <rdfs:comment>Relationship that links a Property with the Year (or
years) in which it was inscribed on the list  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range rdf:resource="#Year"/>
</daml_oil:ObjectProperty>

<daml_oil:Class rdf:ID="CulturalCriteria">
  <rdfs:comment>Cultural Criteria (small version of the ontology). No
attributes possible  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Criteria"/>
</daml_oil:Class>

<daml_oil:Class rdf:ID="Year">
  <rdfs:comment>Year of inscription in the World Heritage List.
(small version of the ontology). No attributes possible
</rdfs:comment>
</daml_oil:Class>

<daml_oil:ObjectProperty rdf:ID="managed_by">
  <rdfs:comment>Relatioship that links a Property with the State
Party (or States Parties) that manage it  </rdfs:comment>
  <daml_oil:domain rdf:resource="#Property"/>
  <daml_oil:range rdf:resource="#StateParty"/>
  <rdf:type
rdf:resource="http://www.daml.org/2001/03/daml+oil#UnambiguousPropert
y"/>
</daml_oil:ObjectProperty>

<daml_oil:Ontology rdf:ID="">
</daml_oil:Ontology>

<daml_oil:Class rdf:ID="StateParty">
  <rdfs:comment>Country or State Party (small version of the
ontology).</rdfs:comment>
</daml_oil:Class>

</rdf:RDF>
```

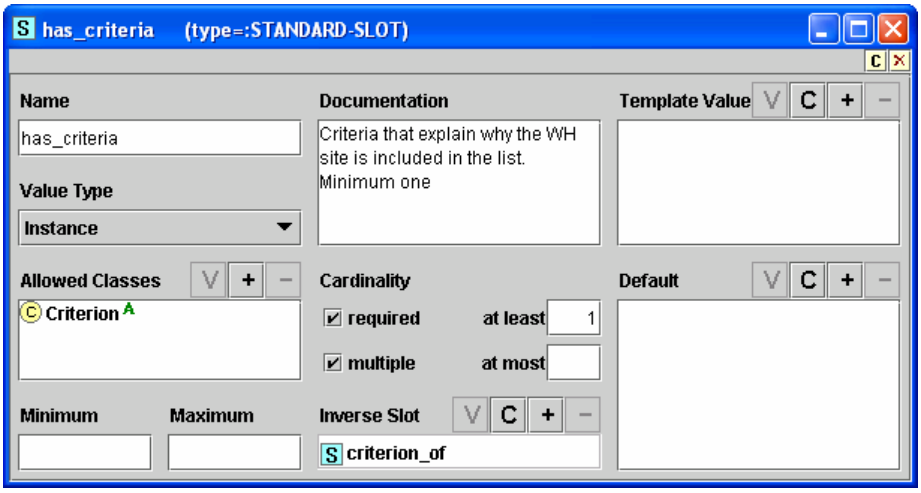
B3. *Protégé Ontology editor screenshots*

This section shows some of the screenshots taken from Protégé-2000 environment while the domain ontology was been built.

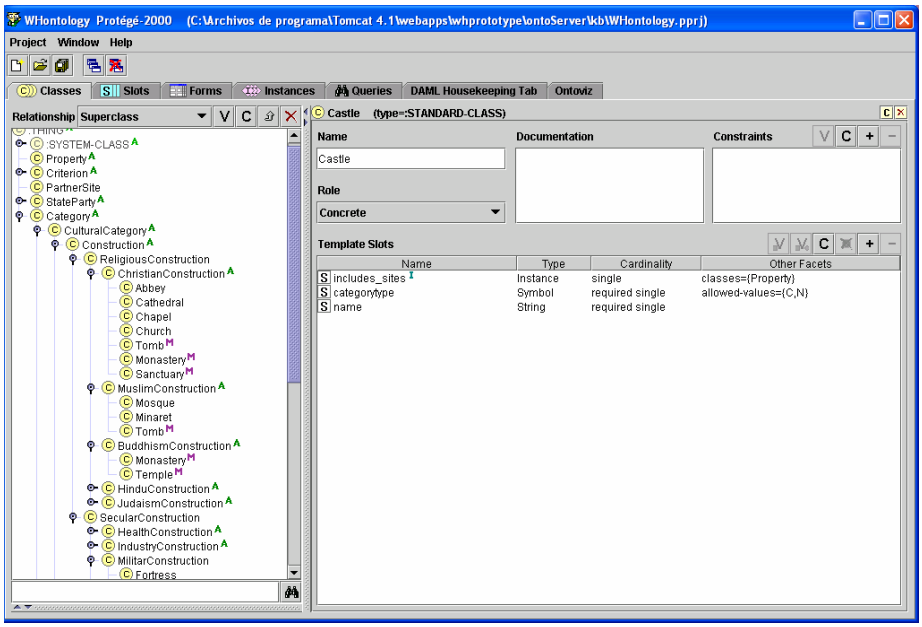
Screenshot showing one of the early stages of the process:



Screenshot showing a case of correspondence between inverse slots:



Screenshot showing some cases of multiple inheritances:


WHontology Protégé-2000 (C:\Archivos de programa\Tomcat 4.1\webapps\whprototype\ontoServer\kb\WHontology.pprj)
Project Window Help
Classes Slots Forms Instances Queries DAML Housekeeping Tab Ontoviz
Relationship Superclass
:SYSTEM-CLASS
Property
Criterion
PartnerSite
StateParty
Category
CulturalCategory
Construction
ReligiousConstruction
ChristianConstruction
Abbey
Cathedral
Chapel
Church
Tomb
Monastery
Sanctuary
MuslimConstruction
Mosque
Minaret
Tomb
BuddhismConstruction
Monastery
Temple
HinduConstruction
JudaismConstruction
SecularConstruction
HealthConstruction
IndustryConstruction
MilitarConstruction
Fortress
Castle (type=:STANDARD-CLASS)
Name
Castle
Documentation
Constraints
Role
Concrete
Template Slots
Name Type Cardinality Other Facets
includes_sites Instance single classes={Property}
categorytype Symbol required single allowed-values={C,N}
name String required single

APPENDIX C. System Requirements

Type	Id	Description
F	R1	The system has to be web-based
F	R2	The system must be accessible by means of a web browser
F	R3	The system domain is World Heritage Centre website
F	R4	Select relevant pages to extract information matching a user's query
F	R5	Be able to extract knowledge from html documents based on ontology
F	R6	The system should be able to store the knowledge extracted.
F	R7	The system should return the results of the search to the user
F	R8	Provided with a prototype of the user interface with basic functionality
F	R9	UI has to follow same guidelines of style and presentation for all pages
F	R10	The user interface must allow a user to enter a request to the system
F	R11	The user interface must present the output to the user
F	R12	Provide the user with some kind of help of instructions if necessary
F	R13	Documents managed must be html pages
F	R14	Documents managed have to be written in English
F	R15	Data managed will be within the system domain
F	R16	Information managed will be at least the one considered critical
F	R17	To have an ontology including relevant concepts of the system domain
F	R18	A knowledge base in the server should keep the information extracted
NF	R19	Admission to the system must be none restricted
NF	R20	Easy of use. System should be appropriate to the user's ability
NF	R21	Easy of learning, intuitive
NF	R22	System must try to achieve user's satisfaction
NF	R23	System must be suitable for the task
NF	R24	System should prevent the users from errors and allow error recovery
NF	R25	System has to be design to be the more automatic as possible
NF	R26	Response times for the extraction tasks have to be the less as possible
NF	R27	Response times for interaction with KB have to be the less as possible
NF	R28	Selection of relevant documents has to be as accurate as possible
NF	R29	Information extracted has to be as reliable as possible
NF	R30	System must be designed to hold future extensions
NF	R31	System must be implemented to allow future improvements
NF	R32	System must handle the processing of simultaneous users requests
NF	R33	The initial capacity of storage of the system is zero
NF	R34	KB has to store information of at least all the WH sites inscribed
NF	R35	The system has to be scalable to cope with new volumes of data
NF	R36	The system architecture has to be prepared to grow as the needs grow.
NF	R37	Run at least under Internet Explorer and Netscape Navigator
NF	R38	Expected to run under Windows and Linux platforms in the client side
NF	R39	The system architecture should provide significant flexibility.

APPENDIX D. A minimalist guide to “regex”

What are Regular Expressions?

A regular expression is a formula for matching strings that follow a certain pattern. Regular expressions are supported by many text editors although some differences can be found between different programs.

Regular expressions are made up of normal characters and metacharacters and are case sensitive. Normal characters include upper and lower case letters and also digits. Metacharacters have special meanings and are described in detail below.

Regular expressions are used in all kinds of text-manipulation tasks and we are used to see them everywhere (more than we could think at first…). If we have ever typed "cp *.txt" at the command prompt, or entered "data?" into a web-based search engine we have already used a simple regular expression. Searching and search-and-replace are among all their more common uses, but they can also be used for testing certain conditions in a text file.

Regular Expression Operators

Regular expressions can be combined with the following characters, called *regular expression operators* or *metacharacters*, to increase their power and versatility. To make really good use of regexs it is very important to understand metacharacters.
The following table lists the main metacharacters and offers a short explanation of their meaning.

.	Matches any single character, including the newline character.
\	Used to suppress the special meaning of a character when matching.
$	Matches the end of a string
^	Matches the beginning of a string.
?	Matches 0 or 1 occurrence of the character or regex immediately preceding. *
+	Matches one or more occurrences of the character or regex immediately preceding. *
*	Matches 0 or more occurrences of the character immediately preceding.
|	Alternation operator, used to specify alternatives. It applies to the largest possi regexs on either side. In other words, `|' has the lowest precedence of all the re expression operators. ☺
[]	Matches any one of the characters between the brackets.
()	Group together the expressions contained inside them. Can be used to concate regular expressions containing the alternation operator

* NOTE: not supported by all applications

In regular expressions, the *, +, and ? operators, as well as the braces { and }, have the highest precedence, followed by concatenation, and finally by |. As in arithmetic, parentheses can change how operators are grouped.

APPENDIX E. JAPE grammar for WH

E1. File main.jape

```
/*
*  main.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, March 2004
*
*  $Id: main.jape,v1 $
*/

MultiPhase: TestTheGrammars
Phases:
first
country
insyear
address
url_pre
url
whtype
criteria
descrip
loc_context
clean
```

E2. File first.jape

```
/*
*  first.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, March 2004
*
*  $Id: first.jape, v1 $
*/

Phase:      First
Input: Token Lookup Split
Options: control = appelt

// This has to be run first of all
// contains any macros etc needed only for standard grammars

////////////////////////////////////////////////////////////
Macro: SPACE
// space
// control
// space control
// control space

(
 ({SpaceToken.kind == space}
  ({SpaceToken.kind == control})?
```

```
  ({SpaceToken.kind == control})?
 )
|
 ({SpaceToken.kind == control}
  ({SpaceToken.kind == control})?
  ({SpaceToken.kind == space})?
 )
)

////////////////////////////////////////////////////////////

Rule: Silly
// We have to have a rule here, so we'll just write something silly
(
 {Token.string == "guachumeniguachumi"}
)
:silly
-->
 {}
```

E3. File country.jape

```
/*
*  country.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, 29 March 2004
*
*  $Id: country.jape,v 1 $
*/

Phase:  Country
Input:  Token Lookup
Options: control = appelt


// Country rules

Rule:CountryContext
Priority: 50
(
   {Lookup.minorType == country}
)
:countryWH
-->
  :countryWH.Country= {rule = "CountryContext"}
```

E4. File whtype.jape

```
/*
*  whtype.jape
*
*  Copyright (c) 2004, DTU
*  Elena Viñuela, March 2004
*
*  $Id: whtype.jape,v1 $
```

```
*/

Phase:      WHtype
Input:  Token Lookup SpaceToken
Options: control = appelt


// WHtype rules
// Type of a World Heritage Site: Natural or Cultural
// We want to match things C or N using the
// context of "Criteria: C" or "Criteria: N"

Rule:TypeContext
Priority: 50
({Token.string=="Criteria"}
 {Token.string==":"}
 SPACE
)
(
 (
  ({Token.string == "C"}|
   {Token.string == "N"})
  SPACE
 )
)
:typeCN
-->
  :typeCN.WHtype= {kind = "typeCN", rule = "TypeContext"}
```

E5. File criteria.jape

```
/*
*  criteria.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, March 2004
*
*  $Id: criteria.jape,v1 $
*/

Phase:      Criteria
Input:  Token Lookup SpaceToken
Options: control = appelt


// Criteria rules
// We want to match things like (i),(ii),(iii)... (iv)
// So there is a context that is that the criteria has
// to be between round brackets

Rule:CriteriaContext
Priority: 50
({Token.string=="("})
(
 (
  {Token.string == "i"}  |
  {Token.string == "ii"} |
  {Token.string == "iii"}|
```

```
  {Token.string == "iv"} |
  {Token.string == "v"}  |
  {Token.string == "vi"}
 )
)
:criterionWH
({Token.string==")"})
-->
  :criterionWH.Criteria= {kind = "criterionWH", rule =
"CriteriaContext"}
```

E6. File descrip.jape

```
/*
*  descrip.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, April 2004
*
*  $Id: descrip.jape,v1 $
*/


Phase:      Description
Input: Lookup Token Split
Options: control = appelt

////////////// Macros ///////////////
Macro: BRIEF
(
  {Token.string == "Brief"} |
  {Token.string == "brief"}
)

Macro: DESC
(
  {Token.string == "Description"} |
  {Token.string == "description"}
)


////////////// Rules ///////////////

Rule: DesContext
Priority:60

(
  (BRIEF
   DESC
   {Token.string == ":"}
   ({Split})*)
)
(
  ({Token})*
)
:desc
-->
 :desc.Description = {rule = "DesContext"}
```

E7. File url_pre.jape

```
/*
*  url_pre.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, 7 April 2004
*
*  $Id: url_pre.jape,v1 $
*/

Phase:      UrlPre
Input:  Token SpaceToken
Options: control = appelt

Rule: Urlpre

(     (({Token.string == "http"}    |
        {Token.string == "ftp"})
       {Token.string == ":"}
       {Token.string == "/"}
         {Token.string == "/"}
        )   |
      ({Token.string == "www"}
         {Token.string == "."}
        )
):urlpre
-->
:urlpre.UrlPre = {rule = "UrlPre"}
```

E8. File url.jape

```
/*
*  url.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela, 7 April 2004
*
*  $Id: url.jape,v1 $
*/

Phase:      Url
Input:  Lookup SpaceToken Token UrlPre
Options: control = appelt



// Url Rules

// http://www.amazon.com
// ftp://amazon.com
// www.amazon.com
```

```
Rule: Url1
Priority: 50

(
 {UrlPre}
 ({Token})*
):urlAddress
({SpaceToken})
-->
:urlAddress.Url = {kind = "urlAddress", rule = "Url1"}

Rule: UrlContext
Priority: 20

(
 {Token.string == "at"}
 {Token.string == ":"}
)
(
 ({Token.orth == lowercase}   |
        {Token.orth == upperInitial}      |
        {Token.kind == number}            |
        {Token.kind == punctuation} |
        {Token.kind == symbol}            |
      {Token.string == "."})+

        {Token.string == "."}

      ({Token.orth == lowercase}    |
        {Token.orth == upperInitial}      |
        {Token.kind == number}            |
        {Token.kind == punctuation} |
        {Token.kind == symbol}            |
      {Token.string == "/"}         |
        {Token.string == "."})*
)
:urlAddress
-->
 :urlAddress.Url = {kind = "urlAddress", rule = "UrlContext"}
```

E9. File clean.jape

```
/*
*  clean.jape
*
*  Copyright (c) 2004, DTU.
*  Elena Viñuela
*
*  $Id: clean.jape $
*/

Phase:      Clean
Input: TempLocation TempOrganization TempDate TempTime TempYear
TempZone UrlPre Ip TempIdentifier Lookup
Options: control = appelt

Rule:CleanTempAnnotations
```

```
(
 {TempLocation}|
 {TempOrganization}|
 {TempDate}|
 {TempTime}|
 {TempYear}|
 {TempZone}|
 {UrlPre}|
 {Ip}|
 {TempIdentifier}
):temp
-->
{
 gate.AnnotationSet temp = (gate.AnnotationSet)bindings.get("temp");
 annotations.removeAll(temp);
}

Rule:CleanLookupAnn
(
 {Lookup}
):toGo
-->
{
//removes all Lookup annotations
//Long startOffset = firstAnn.getStartNode().getOffset();
//Long endOffset = firstAnn.getEndNode().getOffset();
//gate.AnnotationSet toGo = (gate.AnnotationSet)inputAS.get("Lookup",
startOffset,endOffset);
//inputAS.removeAll(toGo);
 gate.AnnotationSet toGo = (gate.AnnotationSet)bindings.get("toGo");
 annotations.removeAll(toGo);

}
```