Ontology-based semantic querying of the Web with respect to food recipes

Leticia Gutiérrez Villarías

Kgs. Lyngby 2004
IMM-THESIS-2004-28

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

I Preface (Formalities)

Title: Ontology-based semantic querying of the WEB with respect to food recipes
Author: Leticia Gutiérrez Villarías
University: Denmark's technical university (DTU)
Institute: Informatics and Mathematical modelling (IMM)
Supervisors: Hans Bruun and Jørgen Fischer Nilsson
Period: From 1st October 2003 to 30th April 2004
Date: 30/04/2004
Points: 30 ECTS

II Abstract

The project consists of a study of the semantic web, and the new technologies to develop it making a comparison with the current web and showing the limitations of the last one.

Afterwards make an application to show the knowledge obtained during the previous research. This application will be an intelligent system able to understand the unstructured web pages posted on the WWW.

The user can make queries about the subject of the web page, and the system will resolve them with some intelligent system and show all the obtained results to him.

The main target of this project is to make a system able to answer the questions made based on the meaning and the semantics of the data, instead of the appearance.

The main goal is to develop a well structured application with a well defined meaning and capable to understand the semantics of the data, being part of the next web generation.

IV Introduction

1 Background

The semantic Web will provide a semantic meaning to the current Web, so it will be easier (for people and machines) to work with this data.
There are several ways to improve the Web by providing it with meaning.

One is to structure all the information available in some semantic-based form, providing the data along with its meaning. These can be done with some of the current semantic web languages, like XML, OWL, DAML, etc. A brief explanation of each one is provided in the next chapter.

But this is a slow task. We can pray for all the new people posting documents in the web would do it in a semantic-based form in order to achieve our goal, but besides this is very difficult, what happens with all the information already available on the net? Should we remove everything and re-write it in a structured way? The answer is very clear, of course not, this is a non sense.
The main strengthen of the WWW is that everybody can post everything on it, no matters what it is, no matters where it comes from, no matters how it is written.

But if we want to improve the information acquiring from the current documents all over the net, some solutions have to be found.
One solution is presented as this project's goal: to extract information from the current web and structure in other way in order to provide semantic meaning to it.

2 Problem description

This project should develop an Information Extraction process, which extracts relevant information from an unstructured set of HTML pages about the recipes' context. This information is processed in order to provide meaning to it; so the system can "understand" the texts, extract information from them, relate it and storage it.
So the user can make advanced queries based on the meaning instead of the semantics. All this process of providing meaning to the unstructured texts is guided by an Ontology.

3 Objectives

Find and extract the desired information within an input set of documents
Automatically relate and structure the extracted information
Automatically storage the information in a structured way

4 Project Motivations

I began thinking about this project when I attended the course "Advanced Databases" imparted by Hans Bruun last year (2003, Spring Semester) at DTU. I was very interested in XML utilities as a semi-structured database, as well as being a Web-oriented language. I began thinking about a possible project to exploit its potential on the Web. Afterwards I read an article written by Tim Berners-Lee [19]. It was then when I came into contact with the concept of Semantic Web. I was fascinated about this new concept, and all its unexplored utilities.

5 Methodology

In this section is described the methodology that has guided this project. A methodology is a set of principles that help the project manager to choose the methods that better fit this specific project.

The use of a methodology helps to produce a better quality product, focusing on the documentation standards, acceptability to the user, maintainability and consistency of software. It also plans the task to ensure that the project will be delivered in time.

Defining a methodology, the reader can easily have an idea of the structure of the project, its objectives, and how they will be reached.

This project differs from most projects because it purpose responds to a specific problem but without a specific solution; find new methods to handle some of the needs and lacks that appear nowadays in the WWW.

This project comes from a set of broad ideas that will be shaped during the project development. It is essential to discern the elements constituting the problem and how they should be improved.

The three main parts of this project are:

- **Gather information:**
  - Define the current lacks of the projects' domain.
- **Define what can be done:**
  - State the limits of the project scope.
  - Performing research to uncover methods that would have an interesting impact on the problem definition
- **Do it:**
  - Find the most suitable implementation for these new methods

This is mostly a research study. It focuses to find and discuss new methods to perform uncovered actions within the project scope, but this project has been also extended with the implementation of new approaches, becoming a theoretical and practical project at once.

5.1 Project planning

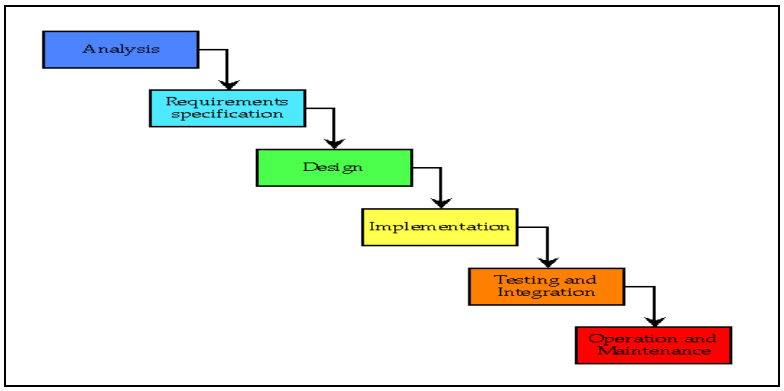


Figure 1 - Theoretical Waterfall Diagram

This project has followed the waterfall diagram schema along its development.
But the theoretical waterfall diagram [Figure1] is too rigid to be applied to an investigation project. This model divides the project in clearly separated development stages.

This particular project has had a lot of feed back from one stage to the others. When new discovers are reached, it is sometimes necessary to reconsider decisions made in previous stages. Due to this continuous feed-back a spiral model could be suitable to define the approach, but in the spiral Diagram a prototype is made each time a cycle is finished, which has not been done in this project.

The diagram which best models the way of doing of this project, is a real waterfall diagram [Figure 2]

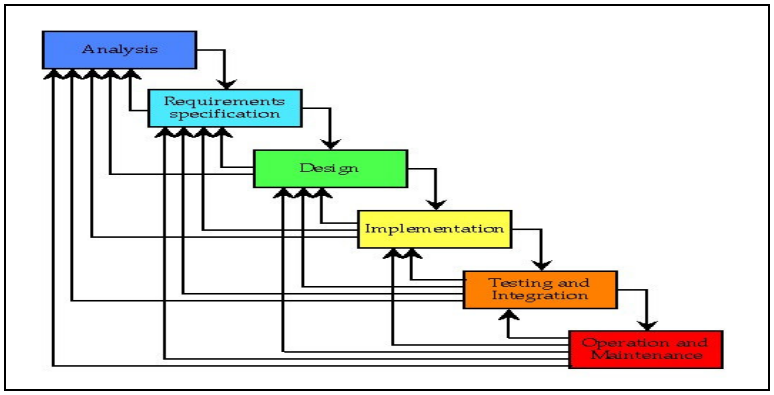


Figure 2 - Practical Waterfall Diagram

The Analysis-Requirements-Design stages where interleaved all the time in this project. As it is explained in future chapters, some problems and new discovers found in the implementation

phase made the project go backwards to the design phase, to remodel some features in a different way.

5.2 Time schedule

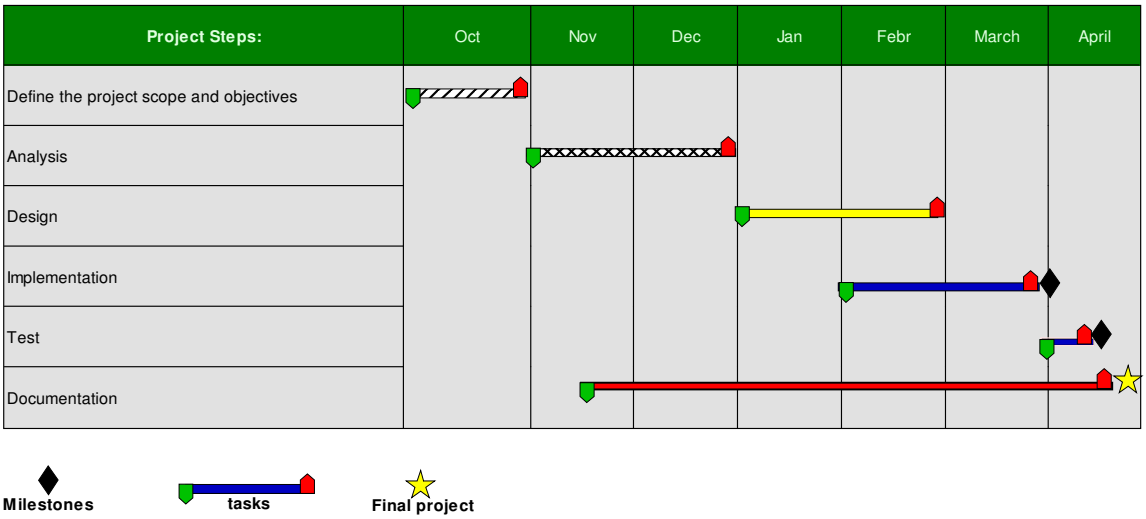


Figure 3 - Time schedule

This is the time schedule followed during the development of this master thesis. The first month was spent in defining the objectives and scope of the project. Afterwards the next two months were dedicated to read articles, analyze the state-of-the-art, find out the lacks of the current situation (concerning the project scope) and propose different possible solutions.
At the end of the third month a proposal of a possible solution was presented.
Then the implementation phase began. The next month was spent in finding which techniques and kind of design are needed to fulfill the objectives. Once made a design of the system the implementation phase begun, this phase is when all the ideas are codified. At the end of this phase a program that is capable to do all the desired features is given. Notice that the design and implementation phases are overlapped; some facts were reconsidered while implementing them, due to several reasons related in the implementation chapters. Finally the testing was performed. The documentation was made all along the project, since the very first months, so it reflects accurately all the project development process.

6 Document structure

The main chapters that compose this project are the following:

- World Wide Web Overview: This chapter is an introduction to the problematic of the current Web and future approaches.
- Problem Analysis: This chapter presents an overview of the specific topic that has been chosen to develop this project.
- Requirements Specification: This chapter specifies the limits of the project. Defines what exactly the functionality of the systems is.
- Domain Modelling: This chapter describes the theoretical models that represent the domain of the project. It is a formal conceptualization of the reality.
- System Design: This chapter will explain the design of this project; this is the choice of the technologies that fulfill the Information Extraction task basing on the selected approach.
- Implementation: This chapter explains the final realization of the selected approach. This is what has been codified and how the diverse tools used are run.

V World Wide Web Overview

7 Current Web Overview

At the beginning the web emerged as some computers interconnected in order to work together and share out the work (1989, Tim Berners-Lee). The web begun to grow and the intranets [see Glossary] and LANs [see Glossary] appeared. But the explosion of personal computers and major advances in the field of telecommunications were the triggers of the web as we know it today. The growth of the WWW has been impressive these last years.

In its first stage the web was thought as some exchange of documents and data and some kind of working collaboration. It was meant to be a big working place where the programs and databases could share their knowledge and work together.

But with the explosion of the media programs, video games, films, music, pictures, and so on, the web now is almost only used by the humans and not by the machines.

Its main problem is that appeared in the WWW is that the information is written only for human consumption in most of the cases. The machines can not understand what the meaning of what is online is. A lot of pictures, drawings, movies and natural language populate the actual web. This meaningless information is not useful at all for the machines, which can not operate with this data; they only show it to the user using a proper format.

7.1 Current Web Languages

A big amount of languages are used to publish data in the current Web. Some of them are: HTML, JSP, ASP, and some Media-oriented web languages: Flash …etc. But they have in common the lack of semantic meaning.

7.2 Current Web situation: information management in the current Web

The incredible growth of the web has as direct consequence a big explosion of all kind of on-line documents. The information storage and collection is like following: the information is stored in large databases kept in the servers. The programs running on the servers generate webs pages "on the fly", basing on this data.
The next picture attempts to briefly describe the information flow schema in the WWW.

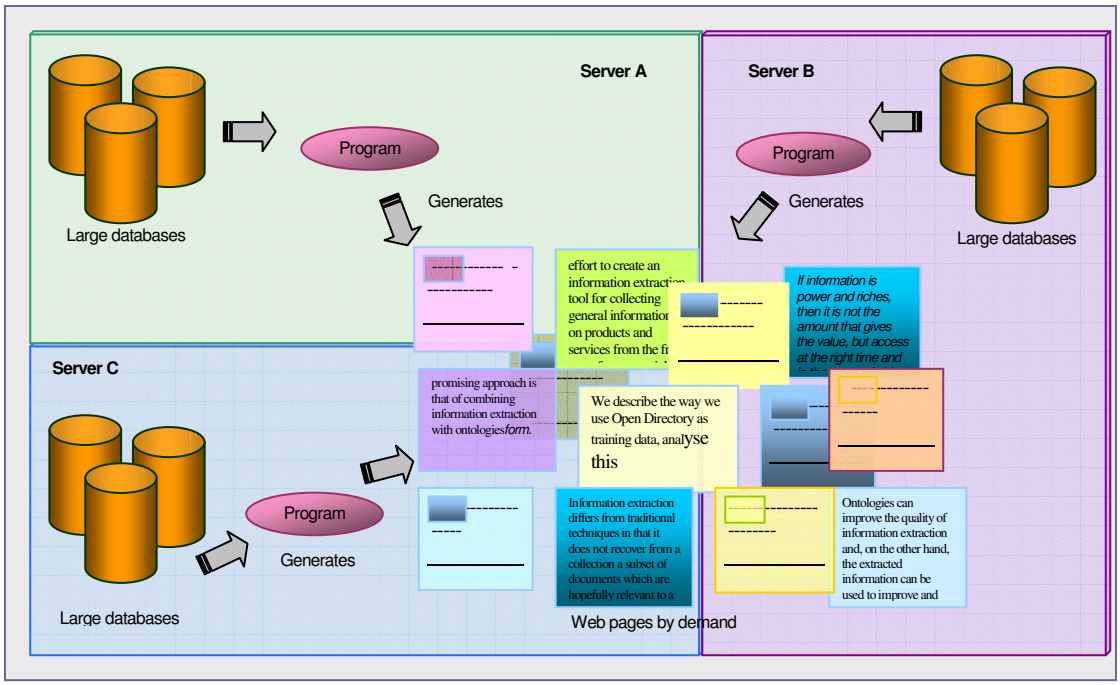


Figure 4 - Current Web Overview

Most of these on-line documents are only made for human consumption, being impossible for the machines to understand the meaning of these documents. Also the human searching is often a hard task and has several limitations, as it is explained below.

7.3 Acquiring information in the current Web: information retrieval

Information retrieval refers to the act of recovering information from the vast amount of on-line documents; getting the desired documents and presenting them to the user.

This is the classic way to obtain information from the WWW.

It does not extract any information from a document; it just picks up some documents among all the available documents in the Web. The user will get a document or set of documents he/she will have to analyze if he/she wants to find the desired information

The non-structured languages of the current Web make difficult for humans, and more for the machines to locate and acquire the desired information. The current methods to retrieve information are *browsing* and *keyword searching*; next picture shows a schema of this information acquiring.

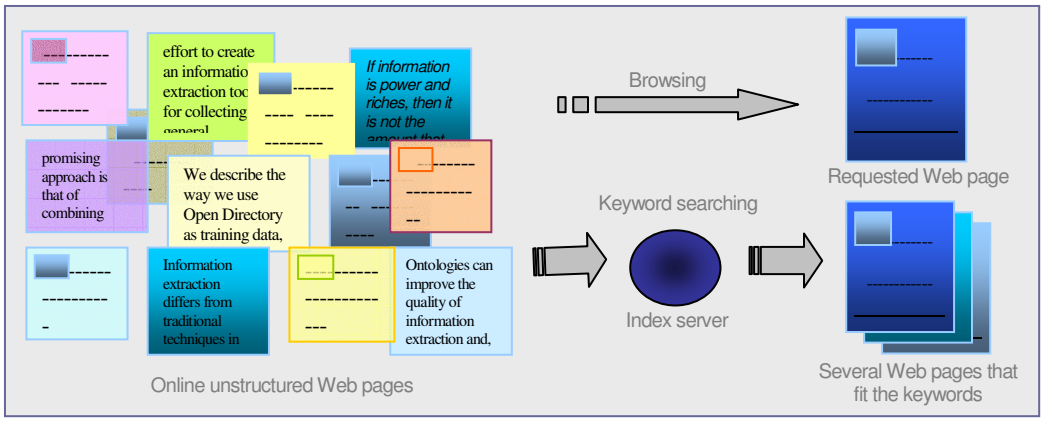


Figure 5 - Current Web Information Retrieval

Both methods have several limitations:

7.3.1 Browsing

Browsing the Web refers to" *the act of retrieving a web page by means of its URI* [see Glossary] *and displaying it in the local browser to see its context"*.

Anybody familiar with the WWW knows the inconveniences of looking for information by means of browsing:

- It is very time consuming
- It is also very easy to get lost and disoriented following all the links; suffering what it is called the "lost-in-hyperspace" syndrome.

7.3.2 Keyword searching

Keyword searching is an easier way to retrieve information.

It refers to the act of looking for information using some words to guide the searching. These words the user wants to look for are entered in an index server which will perform the searching in the Web. The index servers search the WWW following the links and trying to match the input words with what it is written in the web pages.

Keyword searching is more useful than just browsing when looking for information (the user does not need to know the exact URI of the desired web page) but it still has several disadvantages:

- The user must be aware of the several index servers available, and choose the one that fits his/hers necessities.
- The keywords entered are the ones the user considers more relevant in the context he/she wants to look for, which is very subjective.
- These words have to exactly match the words in the web pages.

- Keyword searching normally returns vast amounts of useless data the user has to filter by hand.

"Although search engines index much of the Web's content, they have little ability to select the pages that a user really wants or needs" [Berners-Lee: http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214349,00.html]

7.3.2.1 Example of information retrieval by keyword searching

Let's see a practical example of keyword searching and the subsequent browsing, within the recipe's context:

Imagine for example that someone is looking for a beef recipe that does not take so long because he/she does not have much time to cook today, so he/she enters these words in an index server (Google in this case): *recipe beef cooking-time 1 hour*

The test has been made and 13,700 references have been obtained. This is useless, as it will take the user more time to read and sort the recipes than the hour he/she wants to spend in the kitchen.

He/she can try to redefine the searching to be more accurated: *recipe beef cooking-time less than 1 hour.* This new search "only" returns 4,930 results.

If the user has experience using the index server, the search can be improved with a better use ot the quotes, for example: *recipe beef cooking-time "less than 1 hour"* and then get a more reasonable result of 25 pages. Although the searching has been improved considerably, the user has to still browse all the recipes to decide which one fits his/hers necessities. With this kind of information retrieval, it is not assured that all the pages are recipes' pages. Morevoer, although they belong to this subject, some undesired web pages can be found, for example it was found one with the text: "***not*** *less than 1 hour***"** which is not at all what the user is looking for.

8 What is the Semantic Web?

"The Semantic Web is an idea of World Wide Web inventor Tim Berners-Lee that the Web as a whole can be made more intelligent and perhaps even intuitive about how to serve a user's needs. He foresees a number of ways in which developers can use self-descriptions and other techniques so that context-understanding programs can selectively find what users want." [http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214349,00.html]

8.1 Brief History of the Semantic Web

Because of the incredible growth of the WWW, and the difficulties to cope with these available information (as explained in the previous chapter); the father of the web, Tim Berners-Lee, is now trying to bring it out to a new stage. He has developed a new concept of Web where people and machines could work together and collaborate to share all kind of information. This is called the Semantic Web.

The aim of this new phase is to make the machines capable to understand the semantics of the web. To be able to "read" the web as a human does. For this purpose, many different approaches have been formulated by a lot of researchers. Most of these methods are detailed all trough this project.

8.2 New ways of acquiring information within the Semantic Web: information extraction

Instead of returning the whole Web document, like the information retrieval does; a new way of getting information from the web is needed. This is called the *information extraction*. It consists on extracting pre-specified information out of the document, and structures it in some way so humans and also machines can understand it and treat it. It gets facts out of the web, instead of documents.

Information extraction is much more difficult than information retrieval, but also much more beneficial; the main reason is that the data extracted is structured data, so machines can "understand" it and work with it.

The reason of doing that is because a lot of information is already online in the web, but posted in so many different ways. There is no way to access the information in the servers to make the desired queries, this is only possible trough the already-generated web pages, and as long as they are normally unstructured web pages, only humans can read this pages. So is time to reverse this process. Instead of querying the databases lets query the web.

This will also allow taking data from different heterogeneous sources and merging all the vast information that is published on the Web, giving tailored information to the user.
This way it will be possible to get all the information that is sparsed across the web and reunify it. This allows combining different sources maybe written by so different people, for so different purposes, in so different stiles and with totally different layouts.

But it is a hard task to automate this process, because the machines do not "understand" the meaning of the plain data.

8.2.1 Information Extraction within the New Semantic Languages

Once a web page is written in a semantic language, extracting information is a very easy task. The semantic-oriented languages are just designed to support semantic queries. The user only has to use an appropriate query language to retrieve the desired information.

8.2.2 Information Extraction within the Current Languages

A lot of information is already available on the Web. We can not expect that the entire Web will be rewritten in a structured way. This maybe is never going to happen, as the Web is not a controlled organization were some rules can be applied. Contrary it is a very decentralized and unconstrained place where everybody can post anything they want (with the only constraints of the law rules of a determined country)

As explained before this big amount of unstructured on-line information requires new methods to gather all the spread documents and present sensible information to the user. There is a need to make better use of the current available information. The aim of this project focuses on this task: find a way to extract information from the current web, although it is not structured properly. There is a need to find some methods to "simulate" the semantic web on the current web.

8.2.2.1 The difficulty of information extraction

The information extraction consists of a system that goes over a text with respect to a predefined context, looking for the desired information that fits the context specifications. Afterwards this meaningful information can be structured in some way.

Information extraction is a more powerful way to query the Web, but it presents some difficulties. It does not look for words that syntactically match the words the user wants to look for. Instead it searches the Web looking for facts, for entities and their relationships, in short, for their semantic.

The problem the information extraction systems have to face up refers to the intrinsic complexity of the natural language; there are a lot of ways to express the same fact. Below is an example of these many different ways to express the same idea in the natural language, referred to the recipes context.

- *"You need five tomatoes of fifty grams each to make the tomato soup"*
- *"Five tomatoes of fifty grams are needed to prepare the tomato soup"*
- *"This tomato dish is prepared with five tomatoes which should weight fifty grams each one to get a perfect and tasty result"*
- *"Ingredients for the tomato soup: 5 small tomatoes of 50 grams"*
- *"Take the 250 grams of tomatoes (5 approximately) and…"*
- *"With a quarter of kilo of tomatoes, which corresponds to five small ones, you can prepare a delicious tomato soup"*
- *And so forth…*

The way of achieving the information extraction is making some intelligent programs that could "read" the web pages and redefine them in a structured way, understandable for a machine.
A brief schema of this process is shown in the next picture:

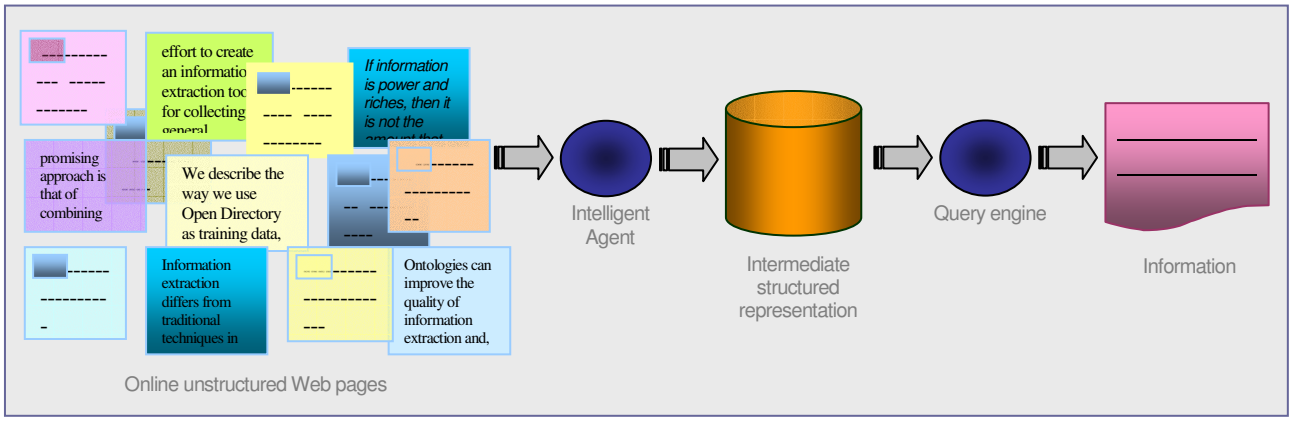


Figure 6 - Semantic Web Information Extraction

*"One of the biggest problems we nowadays face in the information society is information overload. The **Semantic Web** aims to overcome this problem by adding meaning to the Web, which can be exploited by **software agents** to whom people can delegate tasks"* (Esperonto Project IST-2001-34373) [http://www.esperonto.net/semanticportal/jsp/frames.jsp]

8.2.2.2 What is an intelligent agent?

The notion of an agent belongs to the AI field. Agents have application in many AI areas, like process control, electronic commerce, information management, etc. This last application is the one that concerns to this project.

Agents and intelligent agents are not the same, to show the different, both definitions are given:
"Agents are simply computer systems that are capable of autonomous action in some environment in order to meet their design objectives" [1]

"An intelligent agent is … one that is capable of flexible autonomous action in order to meet its design objectives" [1]

Where flexible refers to: respond differently depending on their environment, taking initiatives to achieve their goals and interacting with other agents or humans.

There are several ways to provide knowledge to this agent. Most of them are deeply described next in section

With information extraction the data and its relationships are extracted and structured so the user can make advanced queries and obtained the desired information.

8.3 Semantic Web Languages Overview

So many different languages oriented to create the Semantic Web have appeared within the last years. All these languages are structured languages that can carry on meaning besides giving structure to the text.
They have different characteristics among them. Some are newer than others, and so the newest ones use to make progress from the previous ones, evolving and improving their characteristics.

Different levels of semantic are reached: some languages provide meaning to the texts; others go further and can make assertions and infer knowledge, etc.

- Darpa Agent Markup Language **(DAML+OIL)**. It is an extension of XML and RDF. It can conclude statements by itself.
- Web Ontology Language **(OWL**): The new Semantic Web Standard. It has just became a W3C Recommendation the 10 Feb 2004
- Resource Description Framework (**RDF**): Became a W3C recommendation in 1999. It is a general framework to describe the contents of an internet resource. It is based in Metadata (data about data, definition or description of data).
- eXtensible Markup Language **(XML)**: It is a flexible text language, derived from **SGML.** It can define both the format and the data, and exchange it all over the World Wide.
- Standard Generalized Markup Language **(SGML):** It is a system for organizing and tagging elements of a document. SGML was developed and standardized by the International Organization for Standards (ISO) in 1986
  [http://www.webopedia.com/TERM/S/SGML.html]

In further chapters all these features will be explained in detail and a comparative of all the semantic languages is presented.

8.4 World Wide Web Consortium (W3C)

There is a consortium that actively helps to the achieving of the Semantic Web, and can be considered as one of its main supporters.

"The World Wide Web Consortium (W3C) develops interoperable technologies to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understand" [Definition found at the official page of the consortium: http://www.w3.org/]

The director of the consortium is non other than the "father of the web", Tim Berners-Lee.

He invented the Word Wide Web in 1989, creating the first WWW client and WWW server; he has also defined the URLs [see Glossary], HTTP [see Glossary], and HTML [see Glossary].

The W3C group develops some standards (like recommendations) concerning to the WWW (e.g.: Web definition languages: HTML, semantic web languages: OWL, RDF, XML, etc)

The W3C's goals can be summarize in three ways:

- Provide *universal access to the Web*, making accessible for everybody
- Develop *the Semantic Web*. Make a software environment that allows the users to better use the resources available on the Web.
- Develop a *web of Trust*: Consider the legal, commercial, and social issues caused by the WWW technology.

This project has the ambitious aim to collaborate to the second goal, trying to improve the current Web, raising it to the second Web generation: The Semantic Web.

8.5 How is the Future of the Web?

Step by step the current Web will hopefully turn into the new Semantic Web. But this is not something that is going to happen suddenly.

A study about the future of the web [http://www.aktors.org/technologies/gate/] reports that:

"for at least the next decade more than 95% of human-to-computer information input will involve textual language […] by 2012 taxonomic and hierarchical knowledge mapping and indexing will be prevalent in almost all information-rich applications […] The web revolution has based on human language materials; making the shift to the next generation (knowledge-based web) human language will remain key" [2]

Most of the experts agree on that this is a slowly change. The users and developers of the Web will not change their minds to the Semantic Web unless they have enough motivations and/or facilities.

The main challenge is to provide new tools (servers, editors, browsers) to construct and browse the new semantic Web pages in an easy way; so developers do not have to spend much time and effort creating Web pages with semantic contents; and users do not even notice that they are looking for semantically related information. If they have to spend much time and effort this change will never happen.

Until the Web is beginning to grow semantically, there is a need to simulate the Semantic Web on the current Web using different language-based technologies, which are deeply analyzed in next chapters

VI Problem Analysis

This chapter presents an overview of the specific topic that has been chosen to develop this project.

9 Subject Analysis

The topic chosen to accomplish this project is the online cooking recipes. Several topics were discussed at the beginning, and after a detailed study this was the chosen one.
Other topics considered were: a travel planner, a TV-planner, and the world heritage.
They were discarded for many reasons (like their easiness, narrow relevant information or the lack of personal motivation for these topics)

9.1 The Recipes Overview

There are a countless number of recipes all over the Web. This is a very common topic many people are interested in. This is why it is so spread out and why so many different web pages have been found about this topic.
Some examples of different web pages from different consulted web sites are described in the [Appendix-1] along with an explanation about the different parts and recognizable elements of a recipe.

As the current web agglomerates documents posted by many different people, without any restriction in the way of describe de contents, some discrepancies were found among the studied documents, being a challenge for the IE to cope with this data sparseness.
Some of these differences are related below.

9.2 Complexity of the current recipes: Non-normalized features

After studying a big amount of online recipes I found out the lack of standards in this topic. Some of the differences founded among several recipes are explained in detail (they can be also observed in Appendix-1]

- The nutritional value of a recipe refers to different concepts depending on the consulted web page. (e.g.: some recipes state this value per 100 grams, others per each fellow dinner, other per serving, etc.)
- The measure unity of the nutritional facts (cholesterol, fats or carbohydrates, etc) varies from a recipe to another one. (It is normally expressed in grams, but it can be also stated in kilograms, ounces, etc…) The IE process has to be able to recognize and relate all these different data types.
- Neither the energy value can be assumed to be in a certain unity, it can appear in different units (e.g.: calories, kcalories, kilojoules, etc)

- The same problem appears in the price of the recipe. As the web agglomerates documents posted by all kind of people from all over the world, the price may be expressed in many different currencies (euros, crowns, dollars, etc.)
- The time units do not either follows a standard (Some recipes state it in hours, others in minutes, others in hours and minutes…etc.)
- The way of expressing time also varies from one to another recipe (e.g.:1 hour and 30 minutes, 1h and 30 min, 1:30 h, 90 min, one hour and thirty minutes, ninety minutes, etc.)
- The temperature unit is neither standard (can be expressed in degrees centigrade as well as in degrees Celsius.)
- At last, the numerical values (like the quantity of an ingredient, number of fellow diners, etc) are not express either in a normalized way. (Some recipes express these quantities with numbers: 1, 2, 5; and others with letters: one, two, five ... The fractions are also expressed in many different ways: ½, half, 0.5, etc.)

Some way of converting this data to a certain standard is needed to be able to operate and make comparisons with these data.

Another big challenge is the non-standard way of defining the ingredients. There are no standards or common criteria to express the ingredients of a recipe, several ways were found among all the recipes consulted. Next subchapter will go more deeply into this problem as it is very important to classify correctly the ingredients of the recipes,

9.2.1.1 No standardized way of referring to an ingredient

As there are no standards about describing an ingredient, several ways are used. Some recipes refer to the kind of ingredient, others to its origin, others to its parts, etc…

Kind of ingredient vs. its parts

It is very common to find in a recipe description, the whole animal as an ingredient (e.g.: "250 gr. of **chicken**"), sometimes this information is improved with the part of the animal should be used (e.g.: "8 chicken **wings**"). But many others only describe the part of the animal without referring to any animal in particular, for example: "200 gr. of **liver**". In this kind of description the decision about which kind of animal should be used is leaved to the cook.
All this different ways are (unfortunately for the IE task) very common to express ingredients in the recipes, and they are combined within different recipes.

Kind of ingredient vs. its origin or other characteristics

Another example of the lack of standards is explained below. It does not concern to the parts of the ingredient but to the *type, origin* or *characteristics* of ingredient.

This is for example the problem that faces the cheese classification (among others):
There are a big amount of recipes that explain the ingredients like this:

(Referring to the kind of ingredient) "100 gr. of **cheese**", others present the next ones (the sub-classification of the ingredient) "100 gr. of **mozzarella**", "100 gr. of **parmesan**", "100 gr. **ricotta**", and others have both (the ingredient and the kind of ingredient): "100 gr. of *ricotta cheese*". It is also normal to find the following cheese classifications based on its kind, without specifying a concrete one: "200 gr. of **firm** cheese", "250 of **semi-firm** cheese" etc. Also sometimes classifications like this are found: "150 gr. of **French** cheese" etc.

Another problem is faced about the origin or other characteristics of the ingredient. For example in the *wines* description some recipes describe it just like "**wine**", others refer to its *color* "**red** wine", "**white** wine", "**rosé**", others refer to the *origin* of the wine "**Rioja**" "**Ribera del Duero**", "**Bordeaux**", others to their *age* "**vintage** wine" " **new** wine" "**reserve**" etc.

The normalized way of expressing these ingredients would be: "250 grams of soft Italian cheese named mozzarella", "a red reserve wine from the region of Bordeaux…", where the entities *cheese* and *wine* are detailed with other attributes referring to its origin, kind, or other characteristics. The IE task would be very easy, it would recognize the main entity (ingredient) and then some additional information can be added about the other characteristics.

The problem is that the majority of the ingredient descriptions do not have explicitly written the kind of ingredient they are referring to (wine, cheese, chicken, etc). This main word is left out because the user is supposed to know what these features refer to. For example that "Rioja" refers to a wine and "Mozzarella" refers to a cheese. The aim is to make the intelligent agent to know this as well, but so much information has to be carefully detailed in order to provide this knowledge.

These lacks of standards or official sites have caused the greatest problems during the development of this project. But this was also the most interesting challenge I had to face, and it reflects the real state of the current web: no standards, no consensus, no rules … just a free space where anyone can post its ideas, this is the ideal of the World Wide Web

9.3 Desired improvements

What this project pretends to finish off is this lack of standards in the recipes field by automatically understanding the different ways of expressing a recipe, extracting its relevant information and structuring in such a way that a machine can easily understand its content.

10 Information Extraction (IE) Analysis

This chapter will analyze the different ways to perform Information Extraction within the current unstructured Web.

10.1 Information extraction approaches comparison

Bellow there are deeply described several current information extraction approaches.

They have been all compared, highlighting their weaknesses and strengthens and explaining which kind of texts each one is focused on.

All of them have been considered to fulfill this project information extraction task. I will show the one I have focused my Master Thesis explaining all the reasons that made me make this choice.

10.1.1 Annotations

Although this approach does not really retrieve information from the unstructured current webs, it can be said as a part of the incoming semantic web, because it improves the meaning of the current web pages. So it is fair to take it into account and explain it here.

10.1.1.1 What is an Annotation?

Annotations are commentaries, notes, texts or append files made on an existing web file. These annotations are external documents that improve the current source without changing the web code.

10.1.1.2 How does it work?

Everybody can leave annotations on a web page (if it allows it). The user needs an annotation client installed in his computer so he can introduce an annotation in the web page. Immediately afterwards this annotation is stored in an annotation server, so all the users that visit the page can see it.

10.1.1.3 Pros and cons

A summary of the advantages and disadvantages of using annotations to improve the meaning of the current web are shown in the next table:

Advantages	Disadvantages
The original web page is the same; it does not change at all, since the annotations are attached to the web documents in an external way without modifying its code. They are stored as independent documents in another server (the annotation server) They do not interfere or change the original	It is still difficult to annotate pages, and not everybody knows about it.
	User needs to be aware of what annotations are and install an annotation client in his computer

web page and the efficiency and speed of the downloading rate of the page is not damaged.	
There is a W3C open annotation called Annotea.	It is time consuming and does not assure that it provides meaning to the web page, the annotations can just be some plain text that users post to give suggestions or extend the web contents but without providing any semantics to the page.
	They are sometimes also difficult to entrust, due to anybody can post an annotation.

10.1.1.4 Required document's features

Any kind of document can be annotated as long as it is related to an annotation server. More information about the W3C annotation project, can be found in the [Appendix-2]

10.1.2 Natural-language processing (NLP)

10.1.2.1 What is the NLP?

The approach of Natural Language Programming tries to identify information within natural-language written documents.

10.1.2.2 How does it work?

It makes use of some techniques like: filtering, parsing, lexical and semantic tagging, part-of-speech tagging [see Glossary], relationships among phrases and sentences, grammatical rules, etc.

Human natural language, its rules and characteristics are the backbone of the NLP approach. This approach tries to extract knowledge by deeply studying the texts characteristics.

This is an old approach used in the AI field long time ago. It now aims to teach the computers to understand human language like a human does. This way humans and computers could completely interact. Some researches done in this field try to carry on conversations between humans and machines make the machines able to answer questions, give advices, and a big list of etc.

10.1.2.3 Pros and cons

Advantages	Disadvantages
They are highly effective in plain free text	Non effective with non complete language structures
	Difficult to apply, unnecessary or ineffective in web pages, because of the extra linguistic structures (HTML tags, documents formatting, etc)
	Laborious to develop
	It is content search. Ignores the information the web structure providess.

10.1.2.4 Required document's features

It is necessary to have the data written in natural language and it performs much better if the sentences are complete and follow the grammatical rules.

10.1.3 Ontologies

10.1.3.1 What is an Ontology?

"An Ontology is a formal specification of a shared conceptualization" [[Studer, R.; Benjamins, V.R.; Fensel, D. Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering]

10.1.3.2 How does it work?

The Ontologies are conceptual models that describe the data of interest and control the information-extraction process. They do not rely on the underlying page structure; otherwise they rely on recognizable constants that describe the document's content, so they are fixed to a certain field of knowledge.

This conceptual model instance describes the lexical appearance, the keywords and the relationships of the data of the domain of interest. The ontology will provide the schema to extract and structure the data. It will guide the information extraction from the texts and its subsequent structuring.

10.1.3.3 Pros and cons

Advantages	Disadvantages
The ontology is made manually, but only once for each domain, (it covers all the web pages for that domain)	An ontology is only useful for the domain it was constructed for. If the domain changes then the ontology has to be redefined. This has the additional work to have to make a different ontology for each topic
It is insensitive to changes in web-page format	The pages need to have some particular characteristics to apply this approach.
This approach does not rely on the order or data	Another inconvenience is the language it is focus on. Ontology is a conceptual model for a certain domain in a certain language.
	Also a great knowledge of this domain is required by the ontology developer, who has to perfectly know the entities of this subject and the relations between them

This approach presents some inconveniences, but on the other hand several advantages are reached with this approach. It is very precise (very good rates of performance can be obtained when a good implementation of the ontology is made).
As long as it relies on the data, if the data appearance or its order changes (and web pages usually change very often) the same application can still extract information without doing a single change.

The only dependent module is the ontology model, so if it is necessary to reconstruct the knowledge-extraction system to another subject or to another language, it is only necessary to change the ontology that describes the domain, the rest of the application will remain the same.

10.1.3.4 Required document's features

The Ontology conceptual modeling can be easily applied to unstructured documents with the following characteristics:

Required document's features	
Data-rich	A document is rich in recognizable constants if it has several identifiable constants like dates, names, account numbers, ID numbers, part numbers, times, currency values, etc...
Multiple-record	A texts contains multiple records of information for the ontology if it contains a sequence of pieces of information about the main entity in the ontology
Narrow in ontological breadth	A texts is narrow in ontological breadth if it is possible to describe the application domain with a relatively small ontology

This is very powerful approach, but it is not feasible to use it with all the Web pages posted on the web (if a good performance is desired). However, many of them accomplish these characteristics, so if the domain web pages fit these characteristics, the Ontology approach is as a very good candidate to extract their information.

10.1.4 Web Query languages

This is not a method to extract information from unstructured documents, but from structured documents written in a suitable semantic language. Although that, it is described here because of the importance for this project: Once the information is extracted from unstructured web pages, it can be transformed into a structured web language and then make queries in a very easy way.

10.1.4.1 What is a query language?

The web query languages address the web as a big database where a declarative language can be used to query it. Several query languages for semi-structured web languages have been developed:

10.1.4.2 Pros and cons

Advantages	Disadvantages
Very effective in the query task	They can only be applied to structured or semi-structured webs.

10.1.4.3 Required document's features

The document has to be structured in some way the query language knows, so it can perform the extraction of the information.

10.1.5 Wrappers

Using wrappers to extract information from the Web was one of the most (or maybe the most) used way so far. The wrapper approach parses the unstructured data and maps it into a structured one, relying on the web page structure (HTML mark-up tags for instance) and patterns.

10.1.5.1 What is a wrapper?

This approach builds a wrapper around the Web page and then uses traditional queries to extract the desired information. The wrappers use the underlying structure of the page to format the information contained on it.

10.1.5.2 How does it work?

There are several main tasks while developing a wrapper,

I. Structure the source

The first step aims to *identify the sections and subsections of the page*. This is made by identifying the tokens of interest, such as keywords or maybe complete sentences that indicate the heading of a section dividing the source into sections.
For example the sections of a recipe are the ingredient part and the way of doing part.
This work is done relying on the HTML tags and the text appearance (like bold font, upper case, lower case, letter size, inclusion of special characters, etc)
The most common approach to develop this task is making use of a lexical analyzer, that parses the text looking for certain words that fit its regular expressions identifying them as the page headings.

The next step is *finding out the nesting hierarchy of the Web page*. For example in the recipes context, the nesting structure of the ingredient part is that it is composed by several ingredient descriptions, each one having a quantity, a measurement unit and an ingredient name. The nesting hierarchy within the sections and subsections can be identified by the use of other heuristics. Most of the wrapper developers make use of these algorithms:

Font-size: It has been proved that in some Web pages (not all) font-size is normally decreasing as we go deeper into the nesting structure. Headings use to have bigger font size than their sub-headings.

Indentation space: The indentation space that normally means that one section is nested into another one.

This structuring task states which the interesting tokens and the nesting structure of the Web page is.

II. Build a parser for the source pages

The next function is to generate a parser for the selected source pages. This parser can be automatically made to analyze the incoming pages according to the lexical (tokens of interest) and syntactical (grammar of the nesting structure) results obtained in the previous section.
A parser can extract the desired sections from any source, as long as it follows the source structure determined in the previous step. For any other sources it is useless.

10.1.5.3 Pros and cons

Advantages	Disadvantages
It is domain insensitive. When changing the domain the wrapper remains the same	It is sensitive to changes in web-page format. If the lay-out changes the wrapper is useless and has to be changed.
Valid for all kind of data characteristics	It can easily fail to identify tokens or highlight tokens incorrectly, and it can also fail to guess the document nesting
The Web sources can be queried in a database-like manner, being this way very familiar to many developers.	It is very time-consuming to make a wrapper and generate wrappers by hand is impractical and almost impossible.
Several web pages can be integrated with this approach, building a wrapper around them all.	All this pages have to be similar in layout to be integrated by the same mediator.
Effective when it is applied to highly structured HTML pages	It is only valid for semi-structured texts, not effective when applied to unstructured (plain) texts because of the data sparseness
	Structure based. Ignores the context meaning.

10.1.5.4 Required document's features

As it can be guessed by the wrapper approach, the documents have to follow some strict structure.

They need to be written in some markup language (HTML in my case of study) as long as they rely on the markup tags to guess the structure of the page; they are not meant to be used over plain texts, which make the task more difficult.

The pages also need to be well-structured, with sections and subsections well defined and following a strict agreement of how to represent the different parts of the texts, so they can be easily recognized by their characteristics.

11 Most suitable IE approach for the project subject

11.1 Analyze the recipe's text characteristics

A wrapper or a NLP based approach can be chosen to implement this project, but taken a look to the online recipes' documents fulfill, the following characteristics were found out:

Data-rich

Studying a great amount of recipes I have found out that all of them have several recognizable instances. They all have some fixed sections: the *ingredient description part*, and the *way of doing part*. All the ingredient descriptions are compounded by the *name of the ingredients*, the *quantity* of each ingredient, and *measure* unit. The way of doing contains normally the *cooking time*, the *cooking method*, etc. Some of them also have additional information like the *season* of the ingredients, the *kilocalories* of the dish, and further entities. So many recognizable data is found in the recipes context.

Multiple-record

All the recipes I have found so far have multiple ingredient description.

It is normally found one ingredient description per each line of writing, but this is only as irrelevant information for an Ontology (the contents is what guides the information extraction, not the layout). This information would be useful for the wrapper approach instead.

Narrow in ontological breadth

The recipes domain can be modeled with a relatively small Ontology. All depends on the level of detail wanted in the ingredients classification, but the general recipes model is easy to handle.

11.2 Selected IE approach: Ontology-based

After a deeply study of all the available methods to query the current web, The Ontology-based approach was chosen.

The reasons to follow this conceptual modeling extraction are basically the documents features. So as long as the recipes' structure perfectly fits with the Ontology-based approach, this has been the one chosen, due to it can be applied to all kind of web pages (both to high structured, as well as to more free texts)
The Ontology approach is not so tedious like the NLP one, and is more web-oriented than this one. While the NLP is more oriented to plain texts, Ontologies are to web texts.

Wrappers have been also considered but they were discarded because they are only focused on the data structure, not the data meaning. The data layout of different recipes has been studied, finding that not all follow the same patter. Some are designed with some indentation, others with tables, and others with blank spaces…etc. So no fixed patter can be applied to follow the wrapper approach.

Although this project focuses on HTML pages, because these are the most common pages posted in the net nowadays, this approach can be directly applied to any kind of unstructured texts posted in the net, as well as plain text without any format at all, as long as the data is written in text, not in graphs, pictures, animations, or any other multimedia way.

12 Ontology Building Approach

12.1.1 Ontology Definition

The Ontology definition has always carried a lot of controversy. It has been defined in very different fields, each of them focusing in the characteristics they want the Ontology for. Some of these definitions are exposed below:

The most traditional description of the term Ontology can be found in any dictionary: *"the science or studying of being"* as described in the Oxford English Dictionary. This agrees with the **etymology** of the word Ontology. It comes from Greek and means the *science of beings, or* the *general doctrine of being. Onto* means *existence, being.*

Other fields also give their particular vision of what an Ontology is:

The Ontology concept belongs to **metaphysics**; it is actually the main part of it.

In the **philosophica**l environment is referred as *"the branch of metaphysics that deals with the nature of being"* or *"the study of the kinds of things that exist". The philosopher Aristotle attempt to classify the things of the world*

In the **logic** circles an Ontology is know as *"the set of entities presupposed by a theory"*

In terms of the **Artificial Intelligence** the Ontology is defined as *"the specification of a conceptualization"*. This means, to define terms and the relationships between these terms, in some formal way.

This is the most useful one, as the IA is the field of this project. So from now and on the Ontology will be referred to *as "a set of knowledge terms, including the vocabulary, the semantic interconnections and some simple rules of inference and logic for some particular topic"* [2]
Another definition, refers to the AI systems need to reuse and share knowledge. For this purpose is necessary to define the common vocabulary in which this knowledge is represented. For this purpose: *"A specification of a representational vocabulary for a shared domain of discourse -- definitions of classes, relations, functions, and other objects -- is called an ontology"* [Gruber, T. (1993). A translation Approach to Portable Ontology Specifications. Knowledge Acquisition]

12.1.2 Ontology Purpose on the Semantic Web

Ontology technologies appeared in the 1990s. Their main purpose (on the IA field) is to enable *knowledge sharing* and *reuse,* providing meaning to the Web.

Some structured web-programs appeared also at that time to support the development of the Ontologies. (XML, OWL, DAML, etc)

An Ontology specifies a conceptualization, it represents an abstract and simplified view (vocabulary, relationships and logical rules) of the piece of reality it wants to represent.

Committing to an Ontology, Web agents will know which field and vocabulary they are referring to. It facilitates the knowledge sharing in a certain field, laying the foundations of the language in this context, so it allows several agents to interoperate among them in a certain field.

When two remote applications or agents dialog between them there has to be an unambiguous frame and a common language to talk about. This can be achieved sharing references to the Ontologies currently available on the net. The Ontologies are a consensus about a common domain of discourse. The Ontologies lead the conversation between web agents and they give a possible interpretation of the data that is posted in the web, but never constraining what can be published. This understanding is essential to accomplish automatic tasks on the Web, like transactions, e-commerce tasks, B2B, B2C, etc.

However, the submission to an Ontology does not guarantee the complete interoperation among agents. Some agents can have the capacity to assert some answers to determined queries while others can assert other kinds of knowledge. What the Ontology does guarantee is the coherence and consistency of the knowledge sharing among different agents, not its completeness. [35]

Some of the Ontology utilities for the semantic Web are:

- Web Querying: How to query the web in an efficient way to easily find the documents with the desired characteristics.
- Web sources integration. Find out similarities between different web pages about the same subject and integrate them all increasing their knowledge.
- Restructure current sites. Present different views about the same thing.

There are two possible approaches of how to implement these functionalities; both approaches will be discussed in detail in chapter 23.

12.1.3 Level of detail of the knowledge-model

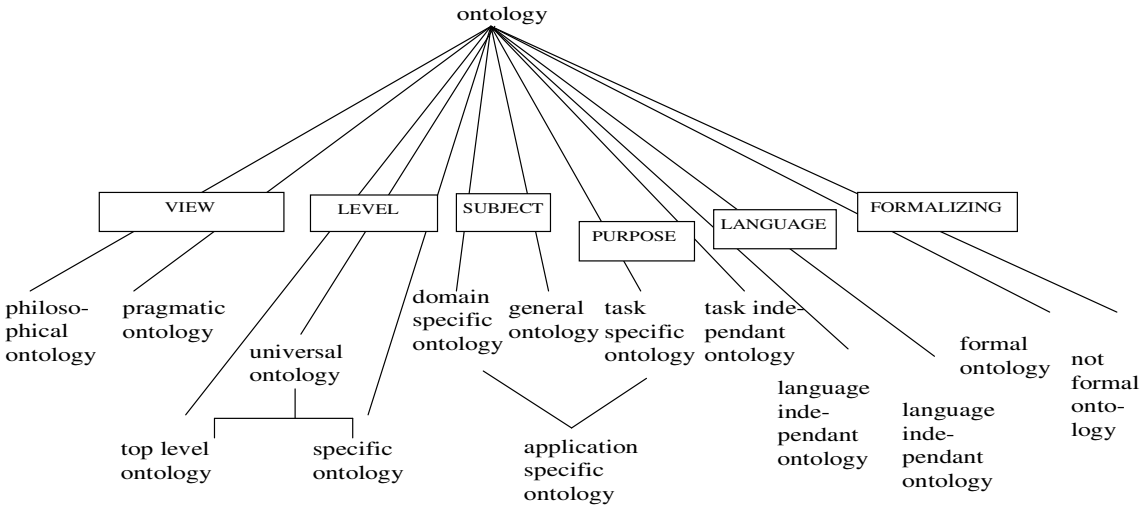
Several attempts to create an Ontology can be made in different degree. Depending on the level of detail we can refer to different concepts:

- The simplest one is a simple group of lexicons and vocabularies
- More complete is grouping together the words that have a similar meaning; creating thesauri [*see definition in the glossary]*.
- We can go further and create a taxonomy *[see definition in the glossary]*, this is a system where the things are hierarchically organized and named in groups with similar characteristics and which can be given different properties.

- Finally, a complete Ontology can be defined when the concepts are related to other concepts. The most advanced stage of an Ontology is when it is capable to define new knowledge.

12.1.4 Different kinds of Ontologies

This project aim is to create a complete Ontology, defining all the relationships among the concepts.
Several kinds of Ontologies can be defined basing on different features. Next picture shows the different kind of Ontologies existent based on different criteria: like the point of view, level, subject, language, etc. [Approaches to ontology design, by Jørgen Fischer]



Bodil Nistrup Madsen, based on a.o.:

Guarino, Nicola (1998). Formal Ontology and Information Systems,. In: *Formal Ontology in Information Systems, Proceedings of the First International Conference (FOIS'98)*, June 6-8, Trento, Italy, 3-15. Ed. Nicola Guarino. Amsterdam: IOS Press.

Figure 7 - Different Kinds of Ontologies

I will explain only the level-based classification; there are several kinds of Ontologies depending on their level.

- **Upper level or Universal Ontologies:** Describe the concepts and relationships of any information of any domain in natural language. Provide a unified upper-level vocabulary that allows different system to communicate between them.
- **Top Level Ontologies:** State fundamental categories and their connections. It eases and guides metadata representation and organization.
- **Specific Ontologies**: Ontologies specialized in a given domain
    - *Regional Ontologies*: Describe a more concrete domain level. Describes specific fields like medicine, culinary, business, etc... Normally comprises diverse local Ontologies.
    - *Local Ontologies*: Even more specific than regional Ontologies. The recipes Ontology can be classified as a local Ontology that makes part of the *culinary* regional Ontology.

The upper level Ontologies is a relatively new approach. It is very interesting and very ambitious as well. It pretends to create some Ontology that can be used for any context defining standards for the Semantic Web. [5]

The ontology defined in this project has the following characteristics: view: pragmatic, level: specific, subject: specific, purpose: task specific, (so it is an application specific ontology), language dependent (only for English language) and formal (follows a methodology)

12.1.5 Ontology's instantiation

Every kind of Ontology has two main parts:

- *Terminological component*: This is the Ontology schema part. This is similar to the database schema. Defines the terms and their structure in the ontology (their relations)
- *Assertion component*: This is the instance data. This is the population of the ontology with individual instances. This part can be taken apart from the ontology and kept in a Knowledge Base. (See chapter 25.6)

12.2 Ontology development (theory): Parts of an Ontology

Afterwards is depth related the parts that compound an ontology.

- Object-Relationship model instance
  - Object sets.
  - Relationship sets.
  - Participation constraints (Designate the minimum and maximum number of times an object in the set participates in the relationship)
  - Generalization/Specialization. (Inheritance)
- Data-frames
  - Constant patterns
  - Context keyword
  - Lexicon patterns

The model instance can be defined with any design language like ER diagrams or Object Oriented languages (UML, for example)
There are two kinds of objects in an ontology domain: **lexical** and **non-lexical objects**. They have some differences in their data-frames.

Only the lexical objects describe a constants patterns and lexicon patterns for its member objects (a set of possible strings)
Both lexical and non lexical objects the data frame contains context keywords; which indicate the presence of an object in the object set.

A data frame is composed of some constants, keywords and lexicons. Below each one is explained:

- Constant patterns: Are some regular expressions (RE) that define the element. For example: the pattern [00-24]:[00-59] is the RE of a time element (referred to a 24 hours way of expressing); the next RE: [1-12]:[00-59][am|AM|pm|PM] is another constant pattern for the same element. The next one is the RE that can model an ingredient constant pattern: [a-z A-Z]* (any letter in uppercase or lower case)

- Lexicons of constants: It is a common set of constant examples that may match a certain object. This is the dictionary of an element. A lexicon for the ingredients can be a list of names from "apple" to "yolk". A lexicon is useful to recognize constants that are difficult to describe with simple patterns, like names for example, that do not follow a certain pattern.
- Context keywords: words that indicate the presence of an object in an object set. Quantity and measure are context keywords for an ingredient. Hour or minutes are context keywords for a time.

There are two kind of set of objects: the bounded sets (times, degrees), which are easily discovered by regular expressions with a high level of accuracy; and the unbounded sets (names of ingredients, cooking utensils) which are not easily expressed with RE. This is when lexicons are defined to check against the objects.

The problem of the lexicons is that they do not sometimes provide a good accuracy; because some overlapping between domains can occur (they only check for the elements, do not care about the context they are defined in). It is when context information should be used to reduce these misclassifications.

Large training sets and tuning can help to solve this problem and give more accuracy to the Information Extraction process.

12.3 Ontology challenges

The construction of the semantic web is a big challenge nowadays. Its semantic meaning is really close to Ontologies, but the use of Ontologies still present several problems. All the ontology developers agree on that the design of an Ontology is not an easy task. [2]

12.3.1 Wide range of non-compatible Ontology tools

The apparently simply task of choosing the right Ontology supporting tool takes a big effort. There are many different Ontology tools for many different purposes: Ontology edition, Ontology merging, translation into Ontology languages, annotating web pages with Ontological information, etc. The majority of these tools have been created isolated and independently so they can hardly interoperate among them. An Ontology developer has to carefully survey all the available tools in order to select the one tat best fits his/hers project requirements. There is still a lot of work to do in this field.

12.3.2 Spread knowledge

If users and developers want to share the knowledge of the Ontologies on the Semantic Web, they need to reach a consensus on their terminology and taxonomy.

For example, in the cooking context, many different Ontologies can be designed. Some cooking Ontologies would take care about the ingredients and its origin while others would focus on their nutritional value, taste, easiness of the recipes, if the final product will be eaten or drunk, way of doing (distinguishing for example between chop and crush, simmer and deep-frying to present a detailed way of doing to the user) etc. Any human or machine that work with these Ontologies should be capable to relate them and then acquire more knowledge of the same domain.

Many experts say is not possible to reach a consensus as it is to put constraints in the new generation process. Moreover some experts on this field predict that the next web generation will be formed by a big number of heterogeneous Ontologies made by different people for many different purposes, rather than being formed by a little amount of well-formed Ontologies made by experts. (This believing bases on the ideal of decentralization and freedom that has always lead the web)

This is a nice ideal of freedom on the Web, but some mechanisms have to be provided in order to relate all these Ontologies. The agents in charge of the information extraction should be able to understand all those different semantics given by different developers and relate them all as a human would do.

Now that the Ontology approach is becoming important within the Semantic Web project, is when some of these difficulties are beginning to become crucial. There is a need to provide consistency to the Ontology approach in some way.

There are some initiatives to sort out all these emerging Ontologies, without putting constraints to the Ontologies made by particular developers.

12.4 Solutions to Reach an Ontology Consensus.

12.4.1 Reusing Ontologies: Merging

Because of the Ontology problems exposed above, the future challenge to the Semantic Web concerning Ontologies is to make easier to create and reuse Ontologies. Currently some of the existing ontology servers have an Ontology library, where the Ontologies link to other Ontologies so they can reuse, change and/or merge terms
These libraries contain different kind of Ontologies (generic reusable Ontologies, domain dependent Ontologies, upper-level Ontologies …etc) but they are still under construction.

In USA there are two current initiatives to develop a Standard Upper Ontology (SUO) and an Ontology Library System, carried out by the IEEE Standard Upper Ontology Working Group

and the DARPA Agent Markup Language (DAML) Program. This library can be found on http://www.daml.org/ontologies/ . It is a free and public library system, which comprises 282 Ontologies up to now. Mainly all the Ontologies are written in DAML. This effort pretends to help to the Semantic Web achievement by the sharing of information from all the developers (everybody can submit and download an Ontology)

Unfortunately we lack of one similar initiative in Europe. Several Ontology researches claim the need of European reference ontology, which should be multilingual and multi-domain. This would be of a great value and enormous help when developing applications based on Ontologies. For example: for the Semantic Web, Natural language applications, Artificial Intelligence applications, Information extraction tools, Natural language processing programs, Knowledge management, e-commerce...etc.

The picture below shows how the euro-reference ontology project, dreams about the future of the Semantic Web; as some generic Ontologies that inherit from generic upper level Ontologies and also gather together and homogenize the individual Ontologies developed by anyone, both experts and non-experts [5].

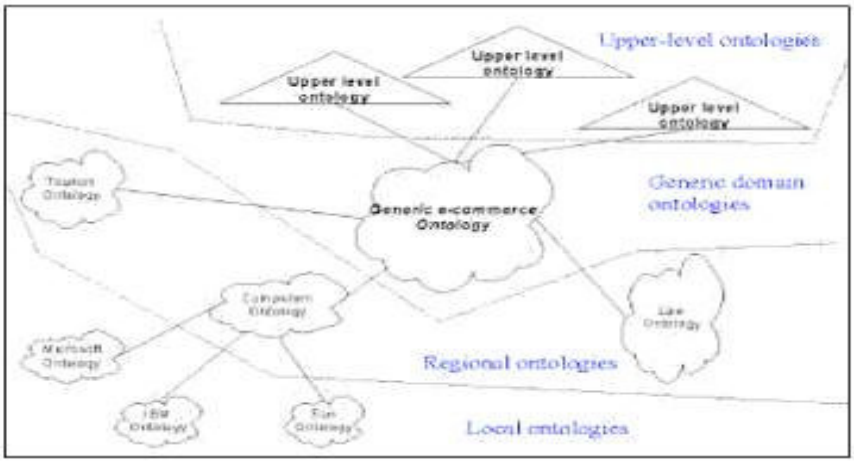


Figure 8 - Ontologies Unification

This project has been extended and improved with the merging of a context-related ontology, which provides extra knowledge to the developed Ontology. More practical information about ontology merging is given below in chapter 31.1.4

12.5 Ontology conclusion

The Ontologies have been used to guide the knowledge extraction task in some projects, and their usefulness is already proved, nevertheless they have much more potential that is still under development, and many problems like the non-standardized methodologies to develop Ontologies, or the non-standardized languages to exchange Ontologies trough different

Ontology tools were found. This is owing to the immaturity of this field. These problems and lacks will be explained in detail in the implementation part, as they appear

VII Requirements Specification

In the real life, the requirements specification document reflects the client requests. This document states the agreement between the client and the developer about what functionalities the system is going to perform. This agreement serves as a contract, were the developer commits himself to provide a product that fulfills all the requirements, and the client also commits himself to accept (and pay for) the product if it fulfills all these characteristics.

This project is not made to fulfill any client desires, but to improve an analyzed problem; so I have stated the requirements specifications myself according to the problem analysis.

13 What functionalities the system should perform

This project tries to fulfill some of the lacks in the context of the online recipes. To reach the objectives stated in chapter 2 the system will perform the following functionalities:

- Automatically find relevant information within the input documents
- Automatically extract the relevant information
- Automatically relate and structure the extracted information
- Automatically storage the information in a structured way

As previous steps to reach these objectives, this project will analyze the problem context (online recipes) defining which kind of documents it can be applied to: define the input data This project will also analyze the input documents, to state what can be considered relevant information.

As an additional feature the system will also perform some queries to retrieve the extracted data. This is an example of how the user can make advanced queries to semantically look for concepts. This is an example of IE from an already structured document.

14 Example of the allowed queries the system should resolve

Afterwards an overview of the functionality of the system from the user point of view is described. Some of the queries the system will perform are the following ones:

Concerning the recipe:

1. Show all the recipes that contain a **certain amount of** one **ingredient** (more less or equal than certain quantity introduced by the user)

2. Show all the recipes with a certain **cooking time** (or less that a certain cooking time)
3. Show all the recipes with a certain nutritional values, for example:
    a. All the recipes with (or less than) certain amount of fats.
    b. All the recipes with only unsaturated and not saturated fats.
    c. All the recipes with certain amount of fiber
    d. Etc

Concerning the ingredients:

1. Show all the recipes that **contain one** (or more) **ingredient/**s selected by the user (e.g.: all the recipes with chocolate and cream)
2. Select all the recipes without an specific ingredient (e.g.: all the recipes without butter)
3. Show all the recipes that contain some selected ingredients but not others (e.g.: all the recipes with lamb and beef but not pork)
4. Select all the recipes without a group of ingredients. This option is useful for people that have some ingredient-groups limitations (like Moslem people or children that can not (or should not) drink alcohol at all), also for vegetarian people that do not want to eat animal-origin food, also for people who dislike some groups of food (fish or fruits for example), people allergic to some compound (like the lactose present in all the dairy-products for example) etc.
5. Show all the recipes that contain ingredients from a selected group of ingredients (e.g.: all the recipes that contain some kind of vegetables)

15 Domain limits

The whole process of IE extraction from unstructured web pages, its conversion to a proper structure that a machine can understand, and the subsequent user queries is shown in the next picture.

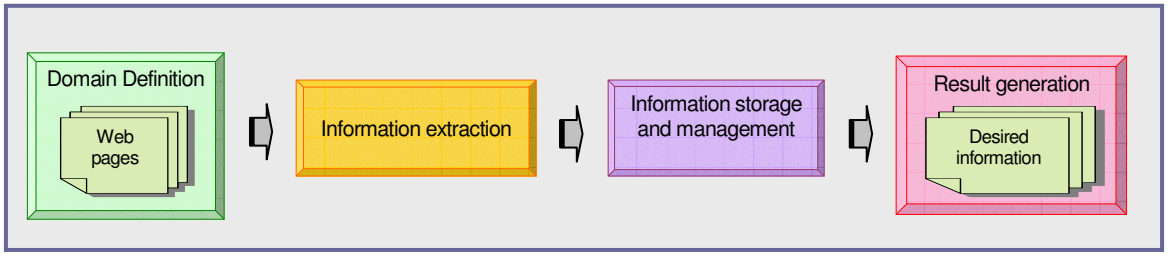


Figure 9 - Information Extraction with Additional Features

These are all the stages to develop a complete system that could interact in the real Web. All this steps are guided by the domain Ontology.

This project focuses on the backbone of this process; this is the Information Extraction and Information management. This corresponds to the second and third stage of the process represented above in Figure 9. And it is detailed below in Figure 10.

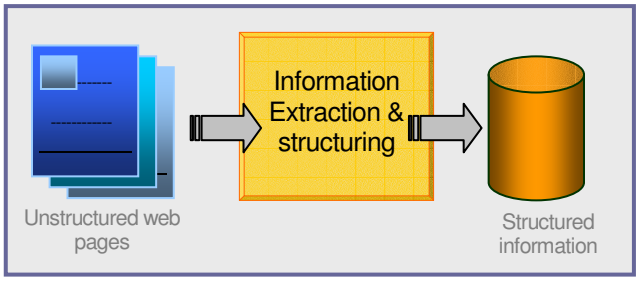


Figure 10 - Information Extraction System

15.1 Input Corpus

This project is focused on the extraction of information from web-pages, concerning to the recipes context and written in HTML. The input pages also have to fulfill the prerequisites stated in chapter 10.1.5.4 about their structure. If they do not roughly follow this structure the success of the Information extraction can not be guaranteed.

15.2 Output

The output of this project is the information extracted information, structured and stored in an online server.

15.3 User Interface

As this is an investigation project, no friendly graphic user interface has been made. The main process that controls the information extraction is run with an MS-DOS command

16 Additional features

A prototype of the final Web where the user can make queries to the system is shown in [Appendix 14].
It is also provided a demonstration of how to make advanced semantic-guided queries and how to present the extracted information to the user in a friendly way. The provided documents are the semantic queries and the final web pages the user will see in its browser each time he/she makes query to the system. These are shown in [Appendix 15]

17 Capacity

The system will be able to cope with a reasonable amount of input pages in a reasonable time. As it is only a prototype to show how the theoretical approaches of the Information Extraction on semi-structured documents can be applied to the practical use, some performance parameters like the time response or capacity of the system have not been taken into account.

VIII Domain Modelling

After identifying the relevant knowledge in the recipes specific context, this big amount of unstructured knowledge needs to be organized and modeled into the Ontology.

This knowledge has to be conceptualized: *"organize and structure knowledge using external representations that are independent of the implementation languages and environments"* [6]

18 Entity relationship vs. Object Oriented

Then first step when modelling the project domain is to choose the kind of representation is going to be used.
We can choose between an entity-relationship (ER) approach and an object-oriented (OO) one.

The object oriented approach was considered but it was rejected for several reasons:

- The ER approach is mainly focused to guide the database design; which is the task it is needed to in this project. It is wanted to modelling the Ontology, and to convert it afterwards in a database to store all the extracted knowledge.
- The OO approach is suitable to design all kinds of systems. It is also capable to model database, but its purpose is more general and complex than the ER one.

This is why the ER approach has been chosen. The conversion between these two approaches can be done relatively easily if needed. The *associations* in OO and the ER *relationships* can be considered equally. *Generalization* in OO is equal to *IS-A* relationship in ER. Also the *aggregation* in OO can be transformed into *contains* relationship.

Moreover the Ontology designing tool has its own methodology and designing language, with its own concepts and relationships. As explained later on in chapter 27.1.2

So, a first schema of the Ontology domain is made in an ER diagram; this is made for a clarifying purpose, to express the structure with a standard notation understandable by any software-engineering expert, but the Ontology was finally developed with the editor's modelling language.

19 ER models of the recipes context

19.1 Introduction

The entity-relationship model is a very common way to express knowledge by means of conceptual modelling.

The ER models can be used to model Ontologies, both general purpose top Ontologies and application-specific Ontologies like the one described in this project.

19.1.1 Kinds of relationships

The application domain can be divided following several criteria. A deep study of the most important kinds of relationships has been made to get an idea of how to model the domain in a correct way. These relationships are detailed in [Appendix-6]

Each of these classifications will lead to different partitions and classification of the domain entities. Beyond this, also the entities of the domain can change depending on the criteria followed. That is why it is vital to choose the right kind of relationships

19.1.2 Kind of Notation: Chen

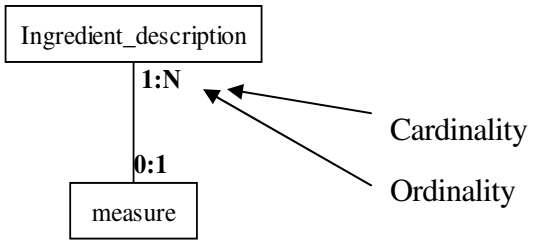
The next Entity Relationship diagrams are modeled following the ***Chen*** notation.
This notation is one of the official ER notations. This is for me the most intuitive one, so that is why I have chosen it.

I will explain the notation with some examples:

- **Entity sets** (also called entities): An entity set is a collection of items with the similar properties. And individual item of the entity set is called an instance of that set. The entities are modeled as squared boxes in the ER model.

- The entities are related to others trough the **relationships**. They are modeled as lines connecting entities.

- The numbers besides the entities on the relationship indicate the minimum and maximum of times the entities appear in the relationship. (**ordinality: cardinality**)

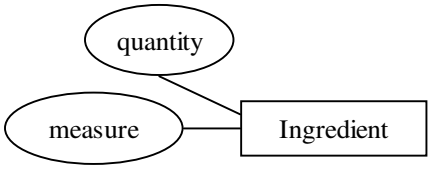


  - The fist number is the ordinality: describes the minimum number of entities that should appear in the relationship. If 0 it is optional, if 1 or more it is mandatory.
  - The second number is the cardinality: describes the maximum number of entities that can appear in the relationship. Can take any value between 0 and N (meaning many)

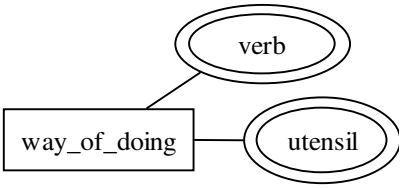
The cardinalities should be read the following direction (a relationship can be read in both directions, as long as it always relates one entity to another one and also the inverse way)

"An ingredient description entity relates at least 0 and at most 1 measure"
"A measure entity is related to 1 or more ingredient_description"

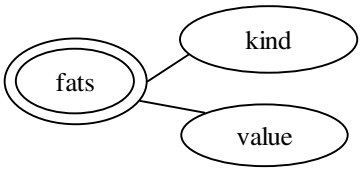
- The **attributes** are the values that describe properties of an entity. They are modeled like ovals attached to the entity. The next drawing shows an entity *ingredient* with two attributes *quantity* and *measure*.
  - Normal attributes (simple-valued)



A simple-valued attribute can only have one value
  - Multiple-valued attributes



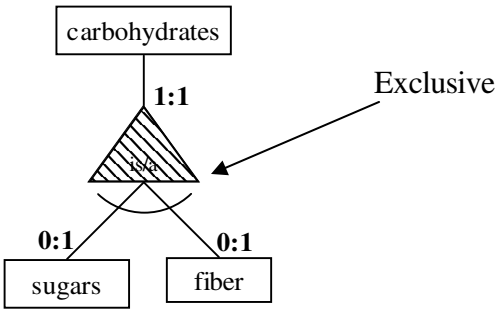
A multivalued attribute can have more than one value.
  - Composite attributes



This is an attributed attribute. Some attributes can be assigned to one attribute, as well as to an entity, to state its characteristics.

- Some of the relationships (the most important ones) show their name in the diagram.

- The **ISA** relationship (meaning inheritance) is modeled by a triangle.



The upper class is the parent class; the lower classes are the ones that inherit from the top one. There are two ways of inheritance: exclusive and non-exclusive. The exclusive one (the upper class can only be of one kind of the lower classes at the same time) is modeled with an arch like in the example. The non-exclusive one (the upper class can be both of the lower classes) does not have the arch.

19.2 ER Attributed Model

The picture below shows the first attempt to create an ER model that reflects the reality of the domain.

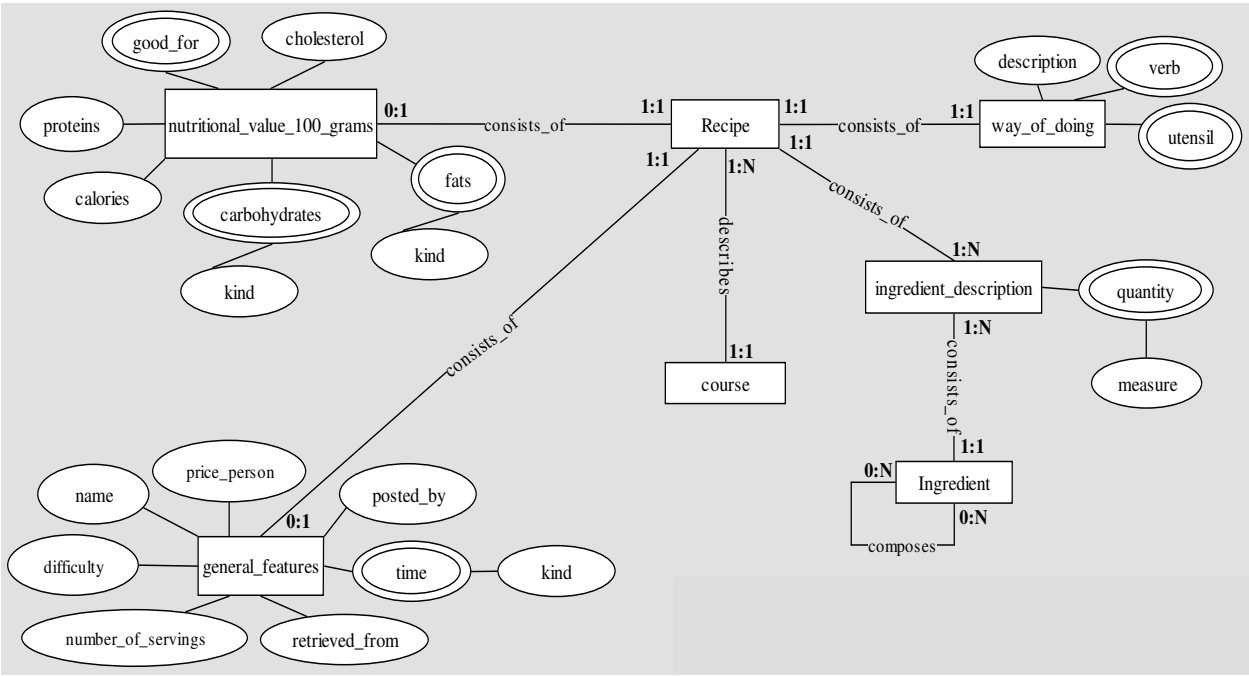


Figure 11 - ER Initial Diagram

19.2.1 Description of the elements

This chapter will describe in detail the diagram presented above. All the entities and their attributes are explained in detail

Recipe entity description	
Attributes	It does not have any attributes, because they have been spited up in its related entities.
Relationships	It holds several relationships with the other entities: ▪ A recipe *consists_of* 0 or 1 *nutrional_value* entities. This means that the nutritional value part is optional in a recipe (some have it and some does not), and can appeared at most once in each recipe. ▪ A recipe *consists_of* 0 or 1 *general_features* entity. (the same explanation as the *nutritional_value*) ▪ A recipe *consists_of* 1 or many ingredients. At least one ingredient has to be present in a recipe. (If no ingredients are used it can be considered a recipe). There are no constraints for the maximum number of ingredients. ▪ A recipe *consists_of* one and only one *way_of_doing*. This part is mandatory and only can appear once in each recipe. ▪ A recipe *describes* one and only one course. Each recipe generates one course. The 1 to 1 relationships can be eliminated from the model if they are considered redundant. This can be done merging the entity recipe with its 1 to 1 related entities: *way_of_doing* and *course*. They in fact do not provide additional meaning, but they are modeled as separated entities to give more clarity to the model, and make it more understandable.

nutritional_value_100_grams entity description	
Attributes	This entity models the concept of the nutritional value a recipe provides per 100 grams of food. It has the following characteristics (modeled as attributes) ▪ *Cholesterol*: Numerical attribute that models the total amount of cholesterol of the recipe ▪ *Carbohydrates*: This is a numerical multi-valued composite attribute. It can have more than one value in a recipe. There are two types of carbohydrates: sugars and fibers. That is why it has been modeled as a composite attribute. The attribute kind states the kind of carbohydrate. The carbohydrate attribute itself states the amount of carbohydrates present in that recipe. ▪ *Fats*: They are modeled equal than the carbohydrates. With a composite multi-valued attribute. Fats can be of several kinds: saturated, unsaturated, monounsaturated a polyunsaturated. One or more kind of fats can be present in a recipe description. ▪ *Good_for:* this is a text-valued attribute. Some recipes state what the recipe is good for (control blood pressure, reduce weight, reduce

	cholesterol...etc) It is a multi-valued attribute because it can have more than one value within a recipe. ▪ *Proteins*: Numerical attribute. States de number of proteins a recipe contains. It is a simple attribute because only one protein-value is stated in each recipe. ▪ *Calories*: Numerical attribute stating the number of calories per 100 grams.
Relationships	The nutritional_value entity only refers to the recipe entity between the relationships consists_of. A recipe consists_of cero or one nutritional_value entities (it is an optional part within the recipe, some can have it and others not, but at it appears at the most once). One nutritional_value entity is referred to one and only one recipe.

This is the nutritional value how the entire recipe. It has to be in mind that the recipe is compounded by ingredients, and those have their own kcalories, vitamins, proteins, fats, carbohydrates, fiber, cholesterol, etc. The nutritional values are not modeled in each ingredient, because the recipes do not describe them, but if we had some additional knowledge about the nutritional values of the ingredients, they could be modeled as attributes as well.

Then it should be taken into account that the following restriction: amount of nutritional values in the recipe is less (some of them like the vitamins can be lost during the cooking) or equal to the total amount of them in all the ingredients

For example: $Recipe_{proteins} = \Sigma ingredient\ (i)_{protein}$

way_of_doing entity description	
Attributes	The way_of_doing entity is composed of three attributes: ▪ *Description*: this attribute is of string type, and contains the plain text that describes all the process to prepare a recipe. ▪ *Verb:* States the verbs used to describe the way of doing a recipe; this is a multi-valued attribute because more than one verb is normally used to cook a recipe. ▪ *Utensil:* States the utensils used to describe how to prepare a recipe, as they can be many, this is a multi-valued attribute.
Relationships	The way_of_doing is related with one and only one recipe, and one recipe is related to one and only one way of doing.

general_features entity description	
Attributes	The general_features entity is composed by several attributes: ▪ *Name*: String attribute that stores the name of the recipe. ▪ *Difficulty*: String attribute that states the difficulty of preparing food following the recipe it refers to. There is not a standard scale to state the difficulty in the recipes world, so this attribute can have different values (easy, medium, complex or 1, 2, 3 or beginners, experienced, experts, etc.). These values should be normalized see chapter 19.3.2

	to be able to operate with it. ▪ *time:* Several kinds of time can be present in a recipe. Some of the recipes refer to a global notion of time, while others specify the cooking time and the preparation time. This is modeled with the associated attribute: *kind*. The attribute time is also multi-value because several times can be stated in a recipe (one cooking time, one preparation time and one global time, or several partial times of different tasks) ▪ *Price_person:* Refers to the estimated cost this recipe has per person. ▪ *Posted_by:* The name or email of the person that has posted this recipe on the net. (this is a very common feature in most of the recipes surveyed) ▪ *Number_of_servings:* Refers to the number of people that can eat with the quantities stated in the recipe. ▪ *Retrieved_from:* The address where the recipe was retrieved from. (http://.....)
Relationships	The general_features entity is related with one and only one recipe, and one recipe is related to cero or one general features entity. This means that this part of the recipe is optional

Ingredient_description entity description	
Attributes	It has only one attribute called *quantity*: this attribute states the how much of each ingredient the cook has to add to make the recipe. This attribute is a composite attribute because it has itself another attribute named *measure*. This attribute states the measurement unit in which the quantity is referenced. This can be of different types and also null.
Relationships	The ingredient description entity is related to the recipe. One recipe consists_of one or more ingredient description (it is mandatory at least one to be considered a recipe), and one ingredient_description belongs to one and only one recipe. It is also related to the ingredient entity. Each ingredient description consists_of 1 and only one ingredient, and one single ingredient can be present in more than one ingredient_description.
Example	The aim of this entity is to model this objects of the real world: "250 grams of cheese": (ingredient: cheese (quantity: 250(measure: grams))) "½ kg of tomatoes":(ingredient: tomatoes (quantity: ½ (measure: kg)))

Ingredient entity description	
Attributes	This entity comprises a very complex classification of other entities. It is going to be carefully studied in chapter 21 It has been separated from the ER model, to make it clear and not to confuse the reader.
Relationships	The ingredient is related to the ingredient description as explained before. An ingredient can also be related to another ingredient. It has a recursive relationship called *composes.* There is a big amount of ingredients that are

	composed by other ingredients. For example: "bread is composed by wheat" "wheat composes bread" "wheat composes pasta" etc… This is an optional relationship in both ways: an ingredient can be composed by many ingredients (composed ingredient) or by none ingredient (simple ingredient). On the other way an ingredient can be part of several or none ingredients.
Example	This entity pretends to model every single ingredient of the recipe: "Cheese", "tomato", "egg", etc…

course entity description	
Attributes	Has no important attributes to remark in the model
Relationships	One recipe describes one course, but one course can comprise various similar recipes.
Example	Some instances of this entity are: "fish-based course" "meat-based course" "soup" "pasta course" "vegetarian course" etc.

19.3 Problems faced when modeling the recipes' context

With a common project of filling a database with some information, a schema like the one stated above would be detailed enough to model the database. In this kind of projects the data is given in a suitable manner so the developer does not have to care about these details. With a suitable manner I mean the following:

- The kind of the input data is the one it is supposed to be. For example: the quantities are some kind of numerical value, the times have their suitable kind, the data values are strings, the logic values are Boolean kind…etc.

- The input data normally has a fixed structure, so the developer can design a database making some assumptions about the input data. For example: The energy value is always stated in calories, the time in minutes and the measures in grams. So these assumptions do not have to me modeled in the database.

But this project is not about populating a database with some predefined information; instead it is about extracting information from heterogeneous unstructured sources. So this is something to bear in mind when modelling the ER diagrams (due to this is going to be the database structure)

There are several values that are not normalized and can be found written in many different ways in the online recipes.

19.3.1 Type of data

This is another problem about extracting information from free-texts. The kind of data someone would expect is not the same than the one is received from the information extraction process.

For example: Anyone would store the time as a numerical value in the database, with a determined format. But when extracting times from online webs many different formats can be found, for example: 1 hour and 30 minutes, 1h and 30min, 1:30 h, 90 min, one hour and thirty minutes, one hour and a half, ninety minutes. All these values will be returned as a string value, as they are text in the Webs.
Some way of converting this data to the correct kind of data is needed, and then some comparisons can be made to know whether these values are equal, bigger or smaller than others (allow comparisons among recipes)

This problem can be sorted in different ways. The different solutions will be related in the implementation part, chapter 33.

19.3.2 Not normalized data

As explained in chapter 9.2 the online-recipes lack of standards in the way of expressing the data, so some assumptions of the previous model can not be made, for example:

- The nutritional value of a recipe can not be assumed to be per 100 grams of food. So some changes in the diagram are needed to distinguish among these different values. An additional attribute *for_each* has been added to the entity (that now it is named as nutritional_value instead of nutritional_value_100_grams) to distinguish what it refers to.
- The measure unity of the cholesterol, fats or carbohydrates can not be assumed, as it varies from a recipe to another one. An attribute *measure* related to these attributes is added. Some explanation of the measurement problematic and comparison tables between the different measurement units is shown in Appendix-20.
- The previous model had an attribute named *calories,* but this assumes that the energy value is given in calories. Because of the lack of standards in the recipes context, this assumption can not be made. The correct model should have an attribute named *energy* with a related attributed that states the *kind* of measure it is weight in. (calories, kcalories, kilojoules, etc)
- The same problem appears in the *price* of the recipe. A related attribute about the *currency* is needed in the model.
- At last, the cooking-time *measure* has to be explicitly declared as well (hours, minutes, seconds …)

The next ER model shows how the previous one can be extended to deal with these restrictions. The new attributes inserted are colored in order to highlight them:

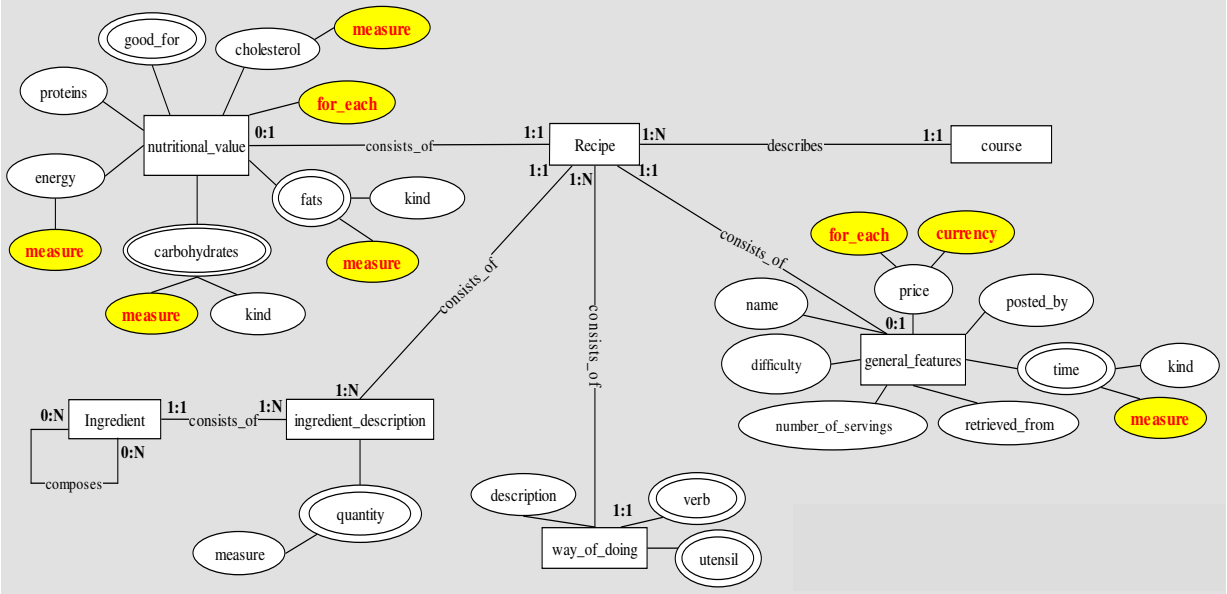


Figure 12 - ER Diagram with additional attributes

20 Dishes Taxonomy

The dishes taxonomy is focused on: the *kind* of dish and the c*ourse classification*.

There are two important classifications of the recipes.

- Basing on the kind of dish they are: soups, pasta and pizza, bread, dessert, meat dish, vegetarian dish, fish and seafood dish, etc
- Basing on when they are normally eaten: appetizer, starter, main course, dessert, etc.

If the application should classify the recipes following those classifications then it would help if the ingredients classification correlates them in some way. For example: If the system needs to give menu suggestions to the user, then it has to classify the recipes following the second classification (that can be based in the first one). As the only way it has to classify a recipe is because the ingredients it contains, those classification should be able to give some clues about this task.

The unique way of carry out this task is making some rules that classify a recipe as a certain course depending on a group of ingredients that are normally common on that course. For example, it can be stated that a recipe is for a starter if it has mostly all the ingredients from the group of (cereals, grains, potatoes…) a main course if it has eggs, meat, fish, poultry, potatoes… a dessert if it contains sweet ingredients, etc.

But this is a very complex feature because it is very subjective and these classifications vary a lot depending on the person tastes, the country, the culture, etc.

Also some dishes can be served as more than one course (the cheese in Spain can be a starter as well as a dessert, for example)

So a simple dish classification has been made, depending on the ingredients it is compounded by (e.g.: fish-course, dark-meat-course, pasta-course, etc). The other inferences about the menus can be done afterwards; there is no need to model them in the diagram. The selected course classification is shown in [Appendix-7]

21 Ingredients Taxonomy

The ingredient taxonomy has been designed very carefully and with a lot of detail, because it is one of the main parts of the recipes classification. This chapter pretends to show all the investigations and models made in within the ingredients context.

21.1 Different classification criteria

21.1.1 Introduction

It is not an easy task to classify all the ingredients in the world. A lot of research has been made in this field, and a big amount of classifications have been done and then discarded for some reasons detailed below.

The aim is to create a good taxonomy of all the existing ingredients within the recipes context.

Taxonomy definition: *"Taxonomy (from Greek* taxis *meaning arrangement or division and* nomos *meaning law) is the science of classification according to a pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval"* [posted by http://whatis.techtarget.com/definition/0,,sid9_gci331416,00.html]

This chapter will be focused to the ingredient taxonomy, and the course taxonomy, which are much related to each other.

There are several different possible taxonomies, each one focuses on different features.

The ingredient taxonomy can be based on different features: Ingredient *flavor*, Ingredient *origin*, Ingredient *parts*, *Pyramidal nutrition*, *culinary approach*, Ingredient *state*, *Simple or compound* ingredients among others. A detailed study of all these approaches is shown below.

21.1.2 Ingredient flavor

One possible classification is based on the flavor of the ingredients.

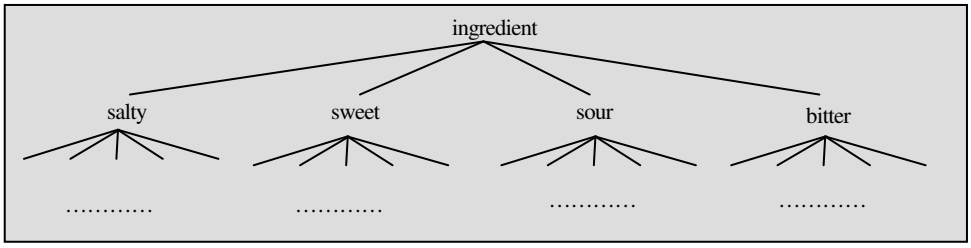


Figure 13 – Ingredient classification by flavor

This ingredient classification is basically orientated to the menu features: the idea is to know the flavor of the ingredients so the system can check if a dish fits with other, it can infers which kind of course a certain dish is (a starter is normally salty, the main dish is also salty, a dessert is normally sweet, etc..) or to infer the origin country (Chinese food is usually sour for example)

But besides that this are not basic features of the system and that it is a very subjective classification (on person maybe likes to eat sweet dishes as the main course) it has another disadvantage: when several ingredients are combined to prepare a dish, the food that results does not usually preserve their flavor and so it is difficult to determine the resulting favor.

Moreover, none of the recipes consulted says anything about the flavor of the ingredients, so as there is nothing to retrieve there is no reason to model it as an entity (could be a complementary characteristic that can be inferred from another source).

According to the kind of relationships theory [Appendix-6] this is an *attribution relationship*. The flavor is a property ascribed to an ingredient as its attribute not an element itself, so it can not be modeled as an entity and related to the ingredient. The classification based on the flavor of the ingredients was finally discarded and this feature will be added as an attribute *Flavor* in the *Ingredient* entity.

21.1.3 Ingredient state

This classification makes the main division of the ingredients dividing them into eatable-ingredients (food) and drinkable-ingredients (drinks). (It has to be taken into account that not all the liquid ingredients are for a drink purpose, e.g.: oil or vinegar)

The drinks are also considered in the taxonomy because some of them are used for cooking purposes, and also because some additional features want to be implemented in the system: e.g.: wine suggestions depending on the kind of food (e.g.: red-wine for meats, white-wine for fish, etc., although this is a very rough approximation, and some wines expert advices should be needed to make a good suggestion)

A first attempt to create this classification is showed in the picture below:

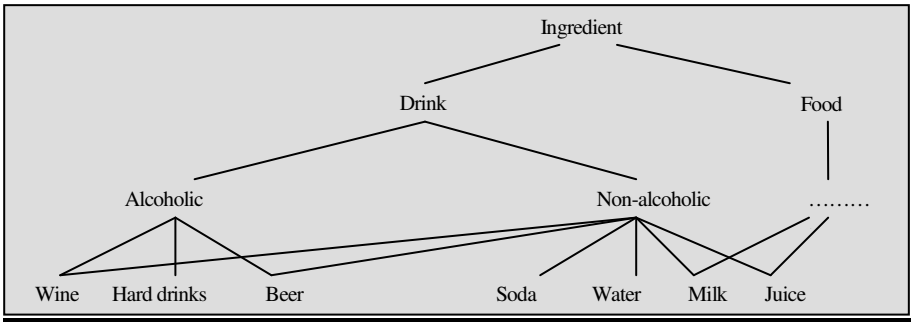


Figure 14 - Ingredient classification by state

This model presents a multi-inheritance structure. This kind of classification is deeply discussed in chapter 21.2.2 along with its advantages and disadvantages.

21.1.4 Ingredient origin

This classification is based on the origin of the ingredients: whether an ingredient is vegetal or animal. The animals are divided in fish, mammals, birds…then it would be very easy pick up some dishes or discard them to make menus to people with food-limitations (vegetarian people, Moslem people, allergic people, etc)

According to the kind of relationships theory [Appendix-6] this is an *IS-A relationship*. The sub-elements are not part of their parent elements (as in aggregation); instead they inherit from them.

A first attempt to create an ingredient taxonomy basing on the origin of the ingredients is shown in the next picture:

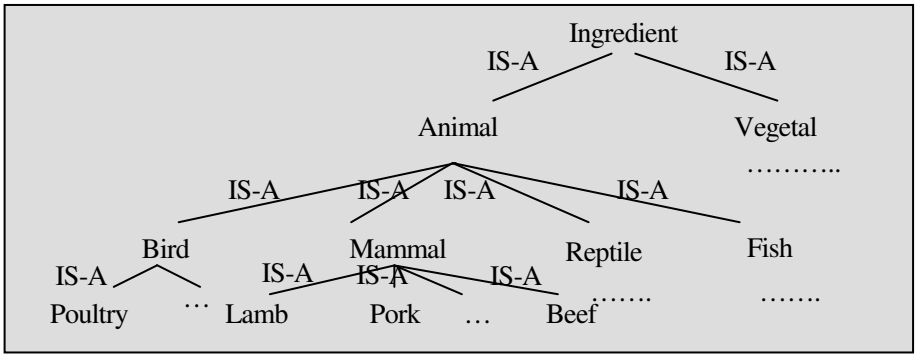


Figure 15 - Ingredient classification by origin

This is a consistent classification. Because of the backbone of the taxonomy is an IS-A relationships, most of the features can be inherited from the top of the classification, and some knowledge can be inferred. See [Appendix-6]

This classification can be combined with other criteria, to improve the knowledge of the taxonomy. The following classification can enhance this one.

21.1.5 Ingredient parts

Another possible way of classification that complements the previous one is to divide the ingredients into their parts.

This can not be made at the first levels of the hierarchy, because the parts in which an ingredient can be divided vary a lot depending on the kind of ingredient (e.g.: an animal can be divided in meat, entrails, bones, etc. The meat can be again sub-divided in leg, chop, loin, etc. (but this depends on the kind of animal as well). The fruits have another kind of parts (seeds, pulp) …)

So this is a good criterion to follow if a very detailed classification is needed, but it will be applied in the low levels of the classification.

This relationship is a kind of aggregation relationship called the *component-integral object composition*. The characteristics of this kind of classification are explained in [Appendix-6]

The schema of the resulting classification of combining the ingredient parts with the ingredient origin classifications is shown below:

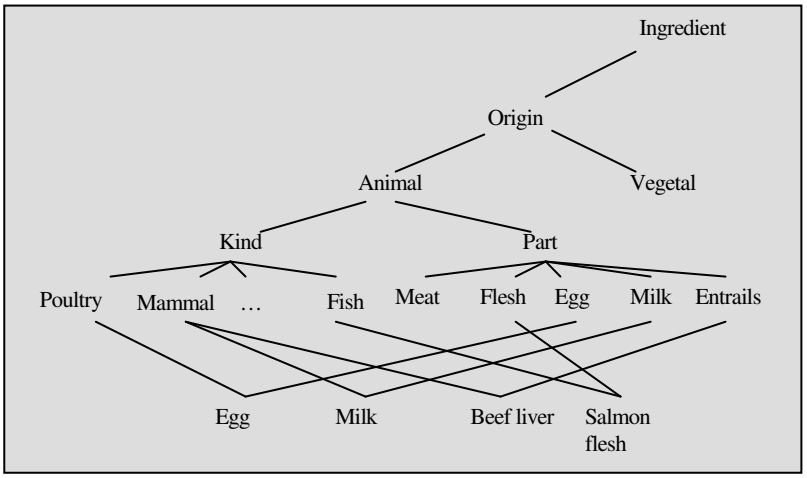


Figure 16 - Ingredient classification by parts

The leaf entities will have more information than in the previous classification.

This divisions have to be carefully made, because the parts in which an animal can be divided depend on the kind of animal, so the *partly* classification should be made in a lowest level than the *kind of animal* division.

This criterion can be followed to divide all the ingredient taxonomy, combining the kind of animal with the parts of the animal. The kind of animals can be more specified, as well as its parts. Next picture shows a little part of the final taxonomy divided by this criterion (and improved with the additional criterion fresh/precooked)

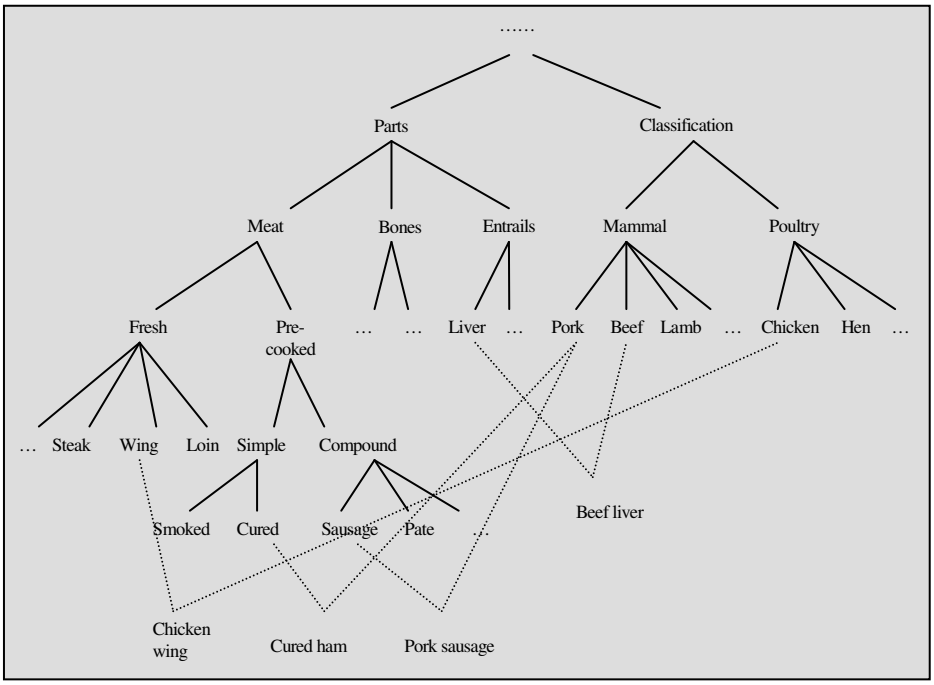


Figure 17 - Extended Ingredient classification by parts

Although it is a very detailed a complex taxonomy it has several disadvantages:

Firstly, this taxonomy is too big to cope with it during the Information Extraction process (as it will be explained in detail in the implementation chapter.

Secondly, the lack of homogeneity and the non-standardized way of describing the ingredients in the online recipes (as it was explained in 9.2), make this classification difficult to apply.

Finally, this classification may be suitable from a biologist or a naturalist point of view, but not from a cook point of view.

The interest of the taxonomy for this project is to classify the ingredients in a way that the system can easily work within the culinary context. That is why another ways of classifications need to be found.

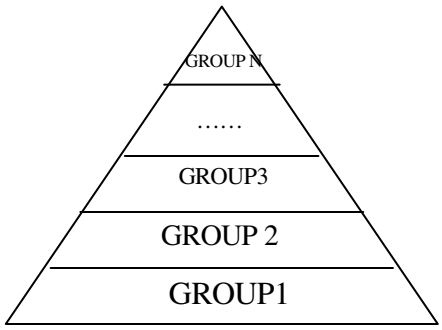
21.1.6 Pyramidal nutrition

To classify the ingredients depending on their nutritional values, I have studied the nutritional pyramids the food experts make. As there is no an official classification, several pyramids were found. Below some of them are presented briefly:

Nutritional value-based classifications	Pyramid-A	Pyramid-B	Pyramid-C	Pyramid-D
Group1	Vegetables, fruits	Bread, Cereal, Rice, Pasta	Cereals	Drinks
Group2	Cereals, grains, potatoes	Vegetables	Vegetables, fruits	Fruits, vegetables
Group3	Milk, eggs, meat, fish, poultry	Fruits	Legumes, nuts, potatoes	Bread, cereals, grains, potatoes
Group4	Fats, sugar, coffee	Milk, yogurt, cheese	Milk	Milk, dairy products
Group5		Meat, Poultry, Fish, dry beans, eggs, nuts	Meat, fish, eggs	Meat, poultry, fish, eggs, legumes
Group6		Fats, oils, sugars	Fats	Butter, oil
Group7				Sugar, coffee, alcohol

[http://www.hsph.harvard.edu/nutritionsource/pyramids.html]

This groups form the nutritional pyramid model, briefly explained below:



- In the bottom of the pyramid are the groups which should be eaten in more quantity every day. (vegetables, bread, rice, etc)
- The top of the pyramid groups the ingredients that should be eaten with very moderation (fats, sugars, coffee, etc)
- This is a very common way to group the similar kind of ingredients, basing on the nutritional values. These pyramids are made by nutritional experts and health institutions.

One advantage when following this taxonomy it that is very easy to state is a dish is balanced (it if has the right proportion of each group of ingredients) or not, which is really important when preparing healthy menus.

On the other hand, choosing this kind of classification has some disadvantages:

If following the pyramidal classification the difference between animal and vegetal ingredients is lost. *For example in the group 5 of the pyramid-B ("Meat, poultry, fish, dry beans, eggs, nuts" there are ingredients from both origins), also the fats for example, can be from animals*

(butter) as well as from vegetables (some oils, margarine). This feature should be modeled as an attribute or make another sub-classification to combine with this one.

Also the big amount of heterogeneous pyramids is an inconvenience in this kind of classification.

But the greatest objection is that there are not heterogeneous groups from a culinary point of view (how a chef would think about the way to prepare these foods). *For example, in the pyramid-A coffee, fats and sugar are gathered together in the same group, although they have nothing to do with respect the cooking way and nor the kind of dishes they are found in. Also there is no a homogeneous way of cooking the group of Meat, fish, eggs.*

All the classifications made have some advantages and are suitable classifications for some other kind of systems. But they were discarded in search for a better solution from the culinary point of view. The nutritional characteristics are of course important and will preserved through several attributes in the elements about nutritional facts (fats, fiber, cholesterol, vitamins, proteins, etc)

21.1.7 Culinary approach

This is another kind of classification, maybe a mixing of the origin and the nutritional classification. Their main classification criterion is the way a chef would think about the ingredients while preparing a dish. The cooking properties of the ingredients (the way of preparing them) can be very different (for example, is not the same to prepare a fish or to prepare a read meat, they have different cooking methods and cooking time) For example, the way a chef prepares a whale dish would be more similar to the way he prepares read-meat than to the fish, although it is a sea-animal.
This classification is very difficult to implement because no expert-knowledge could be consulted while developing the project.

21.1.8 Simple or compound ingredients

Another point of view when classifying the ingredients is to divide them into simple ingredients (e.g.: lemon, orange, lettuce, egg) or as compound ingredients (e.g.: bread, pasta, tomato sauce) this is the biggest challenge when classifying ingredients.

Most of the compounded ingredients can be made mixing other simple (or compound as well) ingredients following the instructions given in a recipe (e.g.: bread or tomato sauce) but some others, although it is possible to make them, they are normally bought already made (e.g.: pasta).

There are also some other ingredients that are not compounded but obtained form processing another ingredient by different ways (like cheese, cream, yogurt, butter, oil), these kinds of ingredients (although possible to elaborate) are almost always bought in a supermarket.

So the problem faced here can be solved in three different ways:

Solution1: Classify the ingredients in two big groups of "simple" and "compounded" ingredients, and relate the compounded ingredients with the simple ingredients they are made of. This solution will lead to a schema like this:

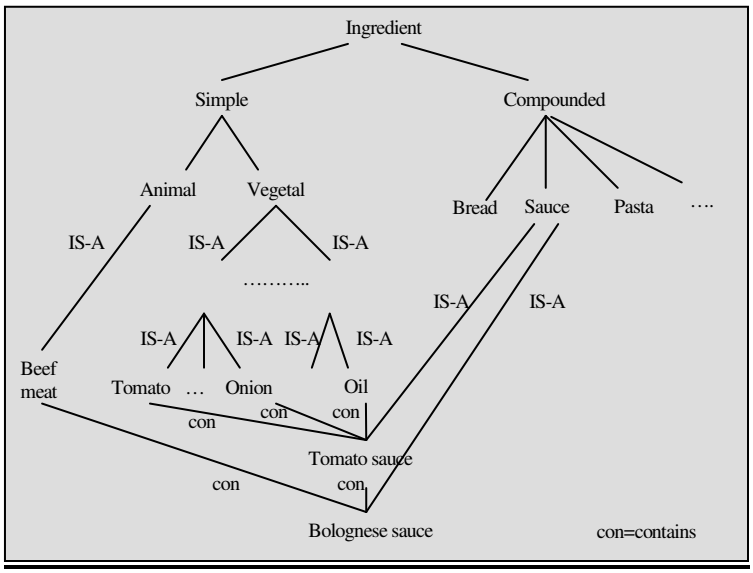


Figure 18 - Ingredient Classification by Simple or Compound

The compounded ingredients are related to the simple ingredients they are made of. This would be a very useful solution that will preserve the nature of the compound ingredients, but there is no universal way of defining a *tomato sauce* for example. Maybe the designer of the taxonomy creates a tomato sauce like the one in the picture that inherits from tomato, onion, and oil, while the user would rather buy it or make it without onion. The bread for example can be normally made with flour, but also rye bread exists, so if the IE tool finds an ingredient description like this: "5 slices of rye bread" and places the instance *bread* in the above classification it will make a mistake.

There are also some complex ingredients like *Soya sauce*, *sweet and sour sauce, condensed milk, tiger nut milk,* etc. or chemical ingredients that are very complex or impossible to make at home and are normally bought already made.

There is the additional problem of where to stop the ingredient division. What can be considered as simple or compound ingredient? Is bread simple or compounded of wheat? Is the wheat a simple ingredient or a compound because it is made from cereals? There is no clear limit of which can be considered the base-ingredients, and which ones the compounded ingredients.

There is a very risky decision to make, and also a huge diagram will result, although it would be the richest one in terms of ingredient-origin apart from the inconsistencies already explained.

The compounded ingredients follow a kind of composition relationship called *material-object composition* within the simple ingredients. In this kind of composition the parts of an object can not be removed (without them loosing their identity). It states what that ingredients are made of. This is more extensively explained in [Appendix-6]

Solution2: Another solution is to consider these compound ingredients, not as ingredients themselves whether as dishes prepared with a recipe. The ER diagram that models the recipes domain will now look like this:

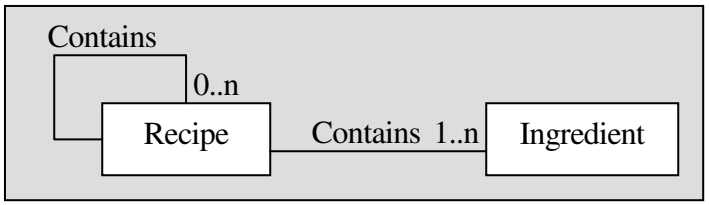


Figure 19 – Way of Represent Compound Ingredients

This diagram means that a single recipe can contain one or more ingredients (simple ingredients) and also can contain other recipes (the recipes that define the compounded ingredients).
With this schema it would be necessary to have all the "compounded" ingredients already stored, just in case other recipe uses them. It will not present the problem related above, because having several recipes of the same ingredient, the user could chose between them the one that fits his preferences better. A very nice feature of this solution is that the compounded ingredients can be presented to the user as a link to their respective recipes,

But this schema will also make the queries to the database much more difficult; each time a compounded ingredient is found in a recipe, it has to look trough other recipes to obtain the ingredients the ingredient is compounded of.

Solution3: The last solution is not to take care about if they an ingredient is simple or compounded, or just only tell the user that it is possible to buy that ingredient or make it following other recipes (can be design with an attribute compounded=Boolean in the ingredient entity).

With this approach all the ingredients will be treated in the same way, which is how it is in the recipes. As long as these ingredients can be bought already made, this is an approach that reflects the reality. This is the easiest way of classifying but a great implicit knowledge is lost; this is a decrease of the information about the final ingredients of the recipe.

21.2 Problems faced during the ingredients classification

As it has seen in the previous chapter, there are many possible ways to make the ingredients classification. As it has been explained, all of them have some advantages for some purposes and disadvantages for other purposes.

21.2.1 The non-standardized way of expressing the ingredients in the recipes

Besides of the big amount of ingredients in the world, the main challenges when making the ingredient taxonomy is the lack of standards and common criteria from the people who post recipes on the web, as it was detailed explained in chapter 9.2.
None of the classifications made can deal with the lacks of standards in the ingredient description. The only solution to this problem is to make the most possible detailed taxonomy, taken into account all the features of an ingredient: kind, parts, color, taste, season, origin, etc. Then each leaf instance would be a combination of all these features. Then it would be possible to parse the non-standardized recipes with this Ontology. But this is an almost impossible task, besides the IE process would not be able to cope with such huge Ontology. Another consensus has to be reached as explained in chapter 21.3

21.2.2 Multiple vs. simple inheritance

Many of the diagrams described above present a multi-inheritance structure: an element inherits from more than one element. A choice has to be made whether to model with simple or multiple-inheritance. The multiple-inheritance is a very powerful but also complex way of representation. Some of its characteristics are explained in next chapter.

21.2.2.1 Advantages of multiple inheritance

21.2.2.1.1 Compact knowledge

The objective of an Ontology is to create a compact and easily maintainable representation of knowledge. Multiple inheritance is a mechanism to make the representation more compact than it would be with single inheritance.

21.2.2.1.2 Allows multiple views of the same concept

One of the main features of the Ontologies is that is possible to merge them creating new more complete Ontologies. Two Ontologies have to be compatible (have similar characteristics) in order to merge them.

If they are modeled with simple inheritance they might have to give up some particular characteristics in order to get compatible with the others.
If multiple inheritance is allowed instead, the merging process is easier. The Ontologies can keep all their particular features as the result ontology will allow different consistent views of the partly Ontologies modeled with multiple-inheritance.

So, multiple-inheritance makes it easier to create upper level Ontologies [this concept was explained in chapter 12.1.4] and standards in the Ontology field.

21.2.2.2 Disadvantages of multiple inheritance

21.2.2.2.1 Compatibility with the implementation languages

One disadvantage of the multiple-inheritance is its application to practical projects.
Not all the development tools support it.

The final purpose of this project is to get all the information retrieved from the web pages and populate the knowledge base with it, having all the information available in a structured way. The kind of database chosen is a semi-structured database, written in XML language. As long as the current version of XML does not support multiple-inheritance [explained in chapter 24.2.3.3] the ontology has to be transformed into a tree structure with only simple-inheritance. This conversion is very easy to make, as explained below

21.2.2.2.2 The Nixon diamond problem: Theory

Another intrinsic problem of the multiple-inheritance is the so called "Nixon diamond problem". This diamond problem is normally explained stating the next problem:

Nixon was both a Quaker *Nixon was a republican* *Quakers are pacifists* *Republicans are non-pacifist*	→	*Is Nixon pacifist?*

So there is a problem getting conclusions about the pacifism of the connection element (Nixon), there is no way to conclude whether Nixon is pacifist or not.

The next picture models the problematic representation:

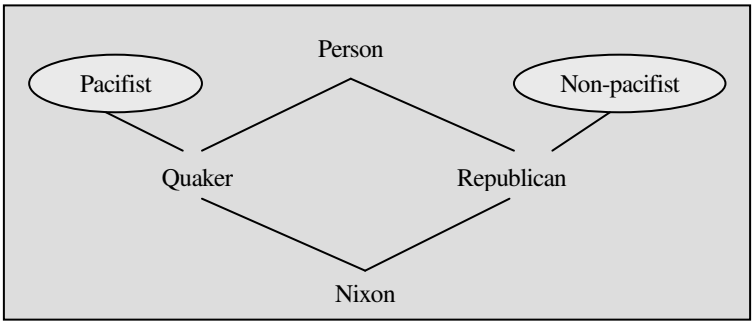


Figure 20 – Nixon Diamond Problem

There is an irresolvable conflict about the pacifism of Nixon; this inference pattern is known as the 'Nixon Diamond'.

There are many logician approaches that try to solve this problem. If the problem is tried to be solved with the *classical logic (skeptical unification)* no conclusion can be reached, as more

information about the subject is needed. If *default logic (Credulous unification)* is used for this purpose, different alternative solutions of the problem are found, but not any absolute solution.

These solutions to the information conflict caused in structures similar to the Nixon diamond problem differ in how much inconsistency they tolerate while searching for the conclusion.

The reasoning process of these methods is:

The *Credulous unification* (not deterministic): will try to resolve the conflict adding as much default information as it can be added without creating inconsistencies. This method will return more than one result. In this case it will conclude that both assertions: Nixon is a pacifist and Nixon is not a pacifist, are acceptable solutions for this conflict.

The *Skeptical unification* (deterministic): it will try to resolve the conflict with just the information that is given in the prerequisites. This kind of reasoning will conclude that is impossible to determine whether Nixon was pacifist or non-pacifist. [Skeptical and Credulous Default Unification with Applications to Templates and Inheritance]

With none of these methods it can be exactly conclude whether Nixon pacifist or not.

21.2.2.2.3 The Nixon diamond problem in the ingredients classification

This conflict appears as well in the ingredient classification; take a look for example to the attempt of modelling drinks (explained in picture 20 and shown in the next picture)

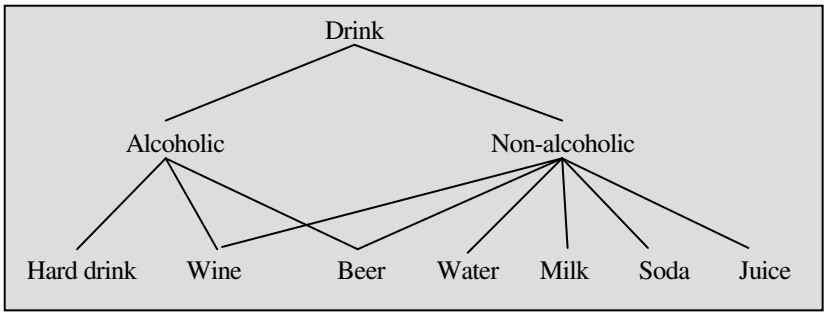


Figure 21 - Drinks Classification by State

This representation divides the drink entity into alcoholic and non-alcoholic drinks. This presents a poly-inheritance classification within some of the leaf entities: wine and beer, because there are alcoholic and non-alcoholic beers, as well as alcoholic and non-alcoholic wines (although this can be skipped by modeling non-alcoholic wine as apple juice).

This structure has a conflict across levels of the inheritance hierarchy as in the Nixon Diamon example. Focusing in the beer example, it has a structure like this:

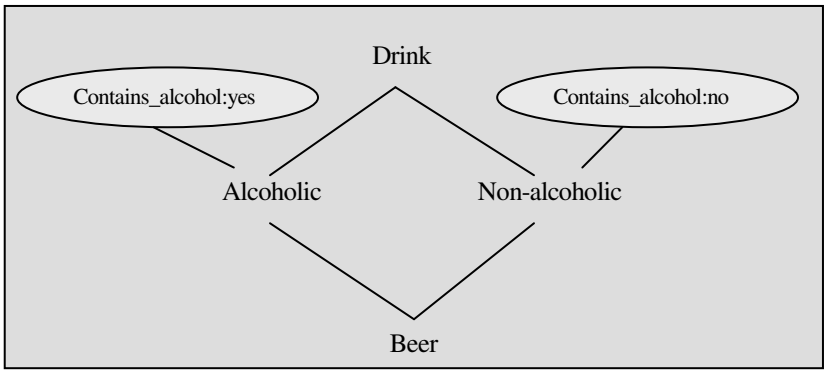


Figure 22 – "Beers Diamond Problem"

Beer inherits both from alcoholic and non-alcoholic drinks, so it also comes across a paradox like in the Nixon diamond. If we ask "Is the beer alcoholic?" with the predicates: "Beer is an alcoholic drink", "Beer is a non-alcoholic drink", and the default values: "Alcoholic drinks have alcohol" and "Non-Alcoholic drinks do not have alcohol" we can not conclude whether the assertion is true or false.

More knowledge is needed in this model to give it complete sense. It can be dome remodeling the diagram to remove the classification about "Alcoholic drinks" and "Non-alcoholic drinks" stating instead if a drink is alcoholic or not with an attribute about the alcoholism percentage. This new classification would look like the following picture:

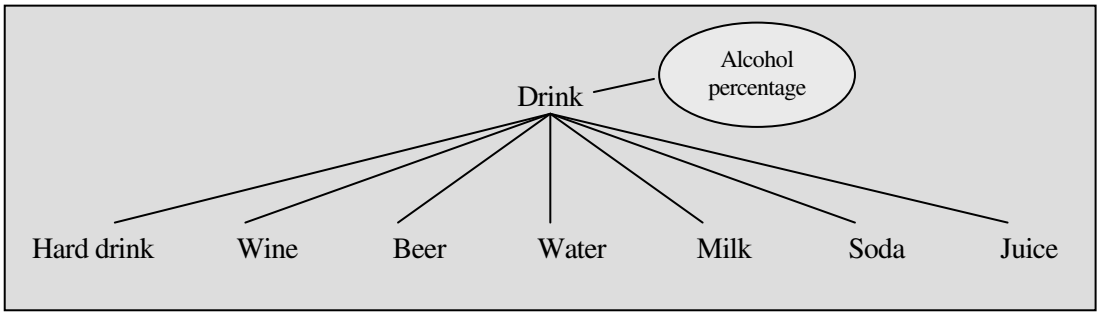


Figure 23 – Nixon Diamond Solution

This is the everlasting problem of the knowledge representation: whether a characteristic should be modeled as a class or as a property. There is no a foolproof method to state the way a certain characteristic should be modeled, so each part of the taxonomy has to be carefully studied to get a classification that makes sense and avoids such kind of problems.

21.2.2.3 How to transform from multiple-inheritance to simple-inheritance

- Duplicating the boundary entities

This is a very simple way of transforming multiple into simple inheritance, duplicating the entities that inherit from more than one entity.
For example, in the case of the *butter* multiple inheritance classification showed in picture 24, the result tree classification will look like the one showed in picture 25

- Remaking the schema by swapping the troubled criteria to attributes instead of a classification criteria

For example, in the case of the butter, a possible solution is to eliminate the fat group and put an attribute (*fat percentage,* for example) in all the ingredients (also an attribute stating if an ingredient is a dairy product or not would be possible, all depends in the system purpose as explained above)
The simple-inheritance classification will look like picture 26:

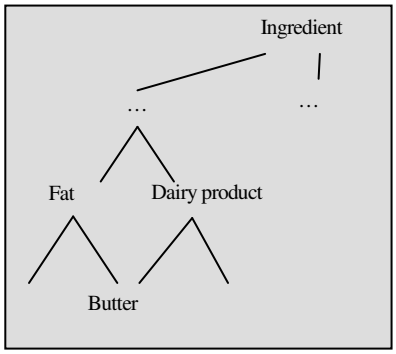


Figure 24 - Multiple-inheritance classification

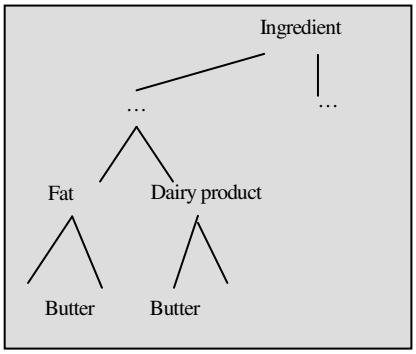


Figure 25 - Tree-classification duplicating the boundary entity

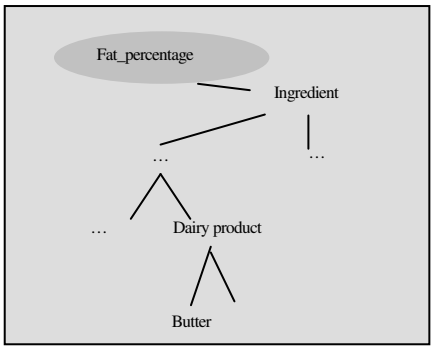


Figure 26 - Tree-classification swapping one classification criteria to an attribute

Other examples of how to transform multiple-inheritance into simple-inheritance are shown in [Appendix-8]

21.3 Selected classification

A decision has to be made in order to make the final classification: which criterion/criteria is the most suitable one/s for the project purpose?

It has to be clear what the purpose of the project is: extract (will guide the IE process) and structure information (will state the database where the extracted information is going be structured in) from recipes web pages. As long as the ontology is the main structure of these tasks, it has to fit the IE purposes.

As explained all along this chapter, there are several possible classifications to describe the ingredients context. All of them are valid, but they are useful for different purposes. So, the classification has to select focusing on the system requirement specifications. [Chapter 13]. The important thing this classification has to focus on is:

- The kind of information the system will manage
- The database constrains
- The information extraction constrains.

The final classification is not presented here. Several classifications were selected and afterwards discarded as the database and IE tools characteristics were found out. I would like to firstly define these characteristics and present afterwards the final model. In order to make this project more understandable and easy to read.

.

IX System Definition

This chapter will explain the choice of the technologies that fulfill the Information Extraction task basing on the Ontologies approach.

22 Introduction

The analysis and the requirements specification phase have already stated the scope of the project, the domain, the objectives and functionality. So it is time now to design the system; to make a planning to accomplish this task.

23 Define the system functionality

As briefly introduced in chapter [12.1.2] there are several utilities of the Ontologies in the semantic web context. This chapter will study each one highlighting the kind of system it is going to be implemented.

The web resources integration [12.1.2] is not a feasible task within the recipes domain. It can be done within restricted domains that state some rules or standards about their contents. For example: in the World Heritage domain, if different pages about the same site are found, they can (and should) be integrated in only one, using the Ontology if a middleware approach is followed, or consolidating the database looking for duplicates if the data warehousing approach is implemented. However in the recipes context the duplicates can not be treated as exactly the same thing. If more than one recipe is found with the same title, they can not be fused in only one, because the probably have different ingredients, different way of doing, cooking times, etc as different people have their particular way of doing the same recipe. The only possible duplicate management in the recipes context is to provide the user a list of the recipes with the same title, or the same kind of ingredients, or the same cooking time, etc, and he/she will decide which one fits his/hers better.

Restructure current sites: [12.1.2]. This feature has been implemented in this project in somehow. Once the relevant information has been extracted and structure, the user can make several queries, so different views of the same page can be presented as the result. It can be said that this utility is a consequence of the main purpose of this project: query the web.

This master thesis project focuses on the Web Querying use of the Ontologies. It aims is to extract information from several webs.

Now that the main task has been stated as extracting and structuring relevant information about the recipes context, what should be thought next is how to accomplish this mission. There are two different ways stated in the next chapter.

24 Define the Kind of system

24.1 Warehousing vs. Middleware

There are two ways of designing an IE system. The first one refers to a *middleware*: the information is extracted from the Web each time the user request it. The second approach is to create a *data-warehouse*: the information extraction is made only once; all these information stored in some structure way for its subsequent use (each time the user makes a request, the system fetch the desired information from the storage device)

This project follows the second approach: the data-warehouse: the Ontology will be used to guide the information extraction from some web pages, structure it and populate a database.

The two first approaches have been taken into account. Finally the warehousing approach has been chosen because of some reasons:
The middleware approach looses a lot of computational time each time a request is made. The entire web (or a certain domain if the searching is restricted) has to be parsed looking for the desired information.

In the warehousing approach the web pages are only parsed once when creating the database. Once they are stored into the database the information extraction is faster and it can be made by normal queries. But this approach also presents some inconveniences; some flexibility is lost, and some problems about inconsistency may appear: if a web source changes the database may still have obsolete information. Also if a web page disappears or new interesting ones appear the database will be behind the times. This should be taken into account and the web should be tracked every so often following some defined criteria.
Besides this inconvenience, the data warehouse approach is the most common method to implement the information extraction from the web (as I could see in all the articles consulted)

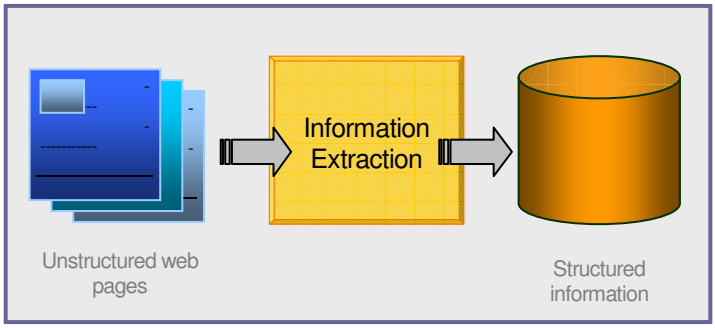


Figure 27 - Warehousing IE Approach

Now the way of acting is clear: extract information, structure it, storage and retrieve it. The next step is to design how to accomplish these tasks using an Ontology. Can it help guiding this process? Next chapter will explain how to fulfill this task:

24.2 Kind of database: Relational databases vs. semi-structured databases

At the present time, there are two ways of storing knowledge: the classical approach of a relational database or the one is to use a semi-structured database.

24.2.1 Relational database

A classic relational database is always composed by the same structure: A set of tables, each table is a set of records, each record is a set of fields and each field consists of a pair: name/value.

Every record of the same table has the same number and type of fields. The relational databases have a fixed structure given by the ER-diagram. There resides its main inconvenience, in their rigid structure.

In a relational database approach the data integration among different sources might be a difficult task, because of its rigid structure.

24.2.2 Semistructured database

The Semistructured databases are the newest generation of databases.

In the Semistructured databases the structure is more flexible as they have more freedom than the relational ones, because their only partial structure specification. A relational database might be incomplete; some records can have some missing or incomplete fields, which is not possible in a relational database.

Another nice feature of the semistructured databases that the relational ones do not have is that they are web-oriented. Their main purpose is to ease the exchange of information through world area networks, especially through the Internet. Because of this, the data are stored together with the structure to easily exchange it trough the net. In this way, the data embeds its meaning.

With Semistructured data it is easy to integrate different documents, since the data are coupled together with information as regard their meaning,

The weak point of the semistructured data approach is its immaturity. The standardization is still in the definition phase, there are some standards but there is still a lot of work to do in this field. Thus the available tools for this incoming approach do not always follow the same specifications. Another problem is that, since all the available tools are quite new, the performance parameters (like time response) are worse than the ones obtained with relational database tools.

24.2.3 Selected model: XML

The Semistructured databases have several nice features, like their flexibility, or integration facilities, but the most important one for this project purpose is the web orientation. Storing the information in a semistructured database, it can be easily exchanged all over the Web. The data and its meaning can be traveling around helping to constitute to the next Web generation.

XML language is the one chosen to represent the information of the database. This tagged and flexible language has several nice properties that make it more useful in some applications.

24.2.3.1 Pros of XML

Besides being a database language it is also a web-based language, so the database it describes can be shown in a browser with the help of a style sheet
("*a style sheet is a definition of a document's appearance. It describes how data sent over the Web using the Extensible Markup Language (XML) is to be presented to the user",*
[http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci213062,00.html]

24.2.3.2 Cons of XML

XML presents some disadvantages like the lack of constrains in the tables and the operations on them. To fulfill this needs some schemas (DTD, XMLSchema) and query languages (XQuery) have appeared.

Here is a fragment extracted from [Reconstructing DTD Best Practice, by Thompson, a member of the W3C]: "*XML Schemas 1.0 would not include multiple-inheritance, as the Working Group is keen to produce a strong design for a single inheritance model first"*. This means that the current versions do not support multiple-inheritance schemas. The W3C group is putting his emphasis on getting the first version of XML Schemas complete and maybe afterwards they will extend this feature in next versions.

So, the multiple inheritance has to be removed from the model, and put it like attributes or duplicate the entities as seen in chapter 21.2.2.3

24.2.3.3 How to represent a Database in XML

A whole database can be represented by means of several XML documents. Each document represents each table of the DB. Inside each table, within a nested structure are represented the records and finally are the fields nested inside the records.
Next figures will clarify this structure:

Database	***table***	***record***	***field***
<!doctype *name* "*url*"> <*name*> ***table*$_1$** ***table*$_2$** **...** ***table*$_n$** </*name*>	<*name*> ***record*$_1$** ***record*$_2$** **...** ***record*$_m$** </*name*>	<*name*> ***field*$_1$** ***field*$_2$** **...** ***field*$_m$** </*name*>	<*name* type="*t*"> *value* </*name*>

Defining a Semistructured database is a very systematic and intuitive process. The developer only has to pay attention to the nesting structure.

The order of the tables within the database, the order of the records within the tables or the fields within the records does no matter. Also some elements can be missing.

24.2.3.4 XML Validation: XML Schema vs. DTD

An XML document with correct syntax is a Well Formed XML, but to get Valid XML, it has to be validated.

Their purpose of the validation is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

The validation can be made against a Document Type Definition (DTD) or an XMLSchema (XSD).

A DTD can be declared inline in your XML document, or as an external reference.
With DTD the XML files can carry a description of its own format with it.
With a DTD, independent groups of people can agree to use a common XML document for interchanging data. An application can use a standard DTD to verify that the data it receives from the outside world is valid and also to verify its own data.

This is the alternative to the DTD that is supported by W3C. It became a W3C Recommendation in May 2001.
XML Schema is written in XML (quite the opposite than DTD) which is much more consistent. It is also more extensible.
DTD can only handle 10 datatypes while XMLSchema has 44 plus the ones defined by the user. XML schema also supports data types and namespaces

These validation methods preserve the structure and the semantics the developer wants to give to the documents. This is an easy way to create, transport, import and interpret documents in a consistently way; contributing to make the transactions and communications more trustable and valid.

24.2.3.5 Query a Semistructured database

If the semistructured database is implemented in XML language, then querying it is a very easy task. There is only need to make the desired queries in the correspondent query language. XML has its own query language called XQuery [see Glossary]

24.2.3.6 XML example of database

The Ontology can be used to model the XML skeleton (DTD or XMLSchema). It can be done because XML can describe documents of all kind of fields and purposes. Afterwards the XML skeleton can be populated with the recognized instances, obtaining then the K.B. modeled as the XML populated knowledge.

The first attempt of describing the Ontology model in XML was made, and XMLSchema is the one chosen to validate it . All the database documents can be found in APPENDIX-25

25 Theory: How does an Ontology guide the IE warehousing process?

Having in mind what an Ontology is (the specification of a conceptualization), what it is going to be used for in this project (to extract data from semi-structured web pages and structure within a database), and what kind of system is going to be implemented (data warehouse); the next chapter provides an overview about the way of achieving this target by means of an Ontology.

25.1 Get the input corpus

The first step is about identifying the input pages the user wants to extract information from.

25.1.1 What is a corpus?

Firstly some definitions about the notion corpus are given. A corpus can be defined as:

"A collection of linguistic data, either written texts or a transcription of recorded speech, which can be used as a starting-point of linguistic description or as a means of verifying hypotheses about a language" [David Crystal, A Dictionary of Linguistics and Phonetics, Blackwell, 3rd Edition, 1991.]

But we are interested on a computer-based use of the corpus, so these definitions were found:

"A very large collection or a body of words, usually stored in computer format" Lancaster University:[http://www.ling.lancs.ac.uk/]

"*In principle, any collection of more than one text can be called a corpus, (corpus being Latin for "body", hence a corpus is any body of text). But the term "corpus" when used in*

the context of modern linguistics tends most frequently to have more specific connotations than this simple definition. The following list describes the four main characteristics of the modern corpus. Sampling and representativeness, Finite size , Machine-readable form, A standard reference " [Corpus Linguistics. Part Two: What is a Corpus, and what is in it? written by: Tony McEnery and Andrew Wilson. Department of Linguistics Lancaster University UK Bailrigg Lancaster LA1 4YW]

This project is focused on HTML pages, which do not have any data structure or semantic meaning. As explained in the project scope in chapter 14 this project focuses on the IE task, considering the first stage of the domain definition out of its scope. The corpus was manually selected from the web, choosing a representative amount of HTML recipes from several web sites.

The corpus used to perform the IE task is attached in the CD due to space limitations

If done automatically, the corpus should be retrieved like the following:

The searching of the corpus can refer to a particular domain (besides the recipes do not have any official site, this would be restricting the web so much) or query the whole web.
Each time it visits a web page, the system should have some heuristics to recognize if that web is a pages of interest for the system.

When the URL has been found it has to be followed to get to the right web source that satisfies the query made by the user; fetching the page just making an HTTP connection to retrieve data. Once having the desired Web page it can be added to the input corpus.

The implementation of this part is an easy task that can be done trough a simple routine that access to internet to fetch the pages via HTTP. The problem of the corpus acquiring is to distinguish among a great number of web pages, which ones refer to the specific domain we are looking for. The Ontology should guide this process, parsing each single page, looking for concepts that match with it. The corpus acquiring can not be done using keyword searching because a lot of undesired pages will be received then (as explained in the problematic of the keyword searching) which is exactly what we are trying to avoid with the semantic query. If doing so the corpus should be preprocessed by the Ontology removing the undesired pages from the corpus.

25.2 Develop the Ontology

The next task alter defining the web pages the system is oriented to, is to develop an Ontology according to the page structure (not the web page structure but the information structure).

An Ontology can attempted to be modeled by means of an ER relationship or other kind of traditional domain modelling (like the Object oriented class diagrams). But with these approaches only the entities, their relationships and attributes can be modeled,; an Ontology is

much more than that. It also comprises some other elements described in detail in next section. With a traditional model we would miss a lot of information.

25.3 Parse the Ontology

The third task is to parse the ontology to get the schema of the database, which is going to format the extracted data. The Ontology is consists of an Object-relationship model and also for some data frame.

The picture below shows the ontology development and parsing. These steps are only made once (for each context; when the application's subjects change, the Ontology has to be modeled and parsed again). All the following steps will remain the same.

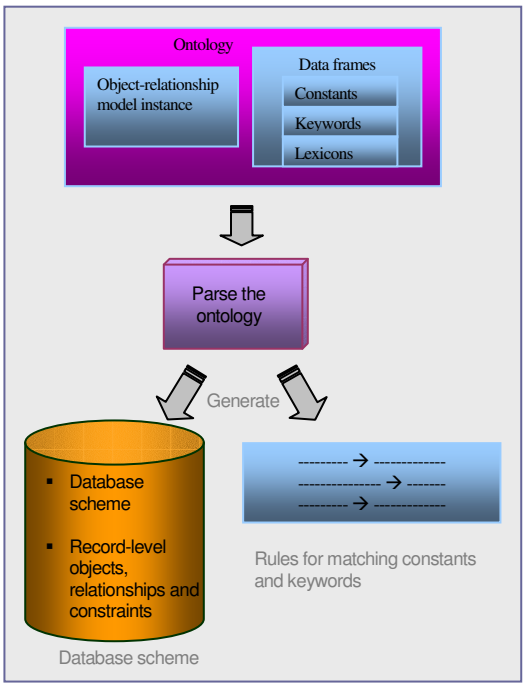


Figure 28- Ontology Parsing

25.4 Preprocess the input corpus

At the same time we need to separate the information into records. If the web pages have more than one recipe per page the texts have to be preprocessed, using some heuristics to identify the record separators.
When parsing a text with an Ontology it looks for every recognizable entity, constant, relationship, etc in the whole text. It does only rely on the data, so it does not take into account any separator or braking symbol in the text, so one input text is always going to generate one instance of the Ontology.

This is why some pre-processing has to be done before treating the input corpus. It requires carefully studying the page layout and the HTML tags that separate records to discover the record-boundaries.

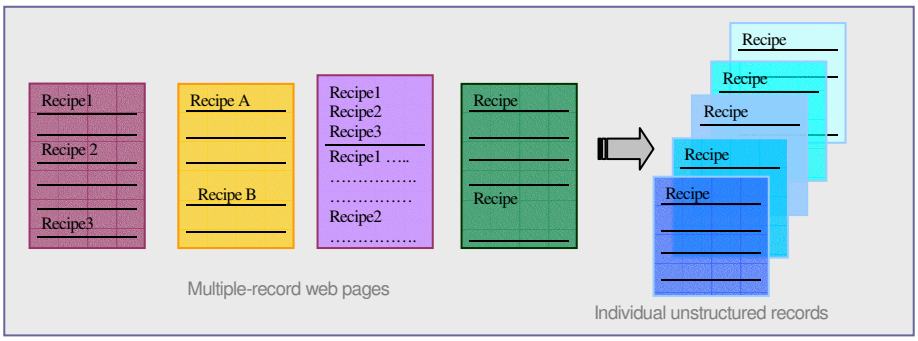


Figure 29 - Input Corpus Preprocessing

25.5 Invoke routines

Once having the rules for matching constants and keywords and the individual unstructured records, the next step is to create a recognizer able to extract the objects expected to populate the model instance.

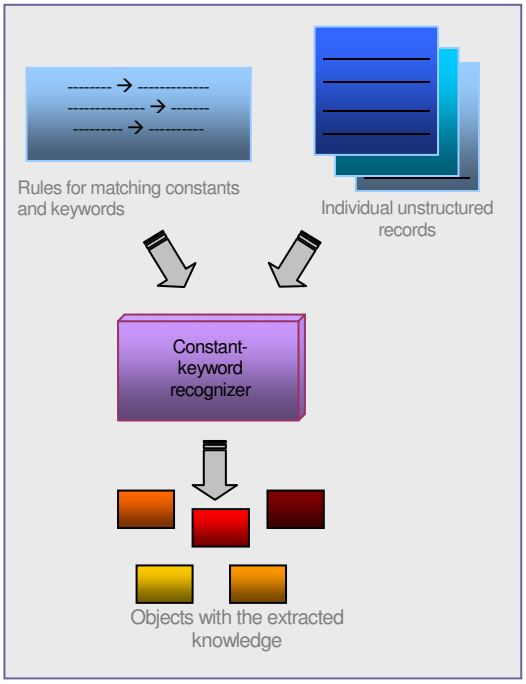


Figure 30 - Routines to Extract Information

25.6 Populate the database: creating the knowledge-base

Knowledge base definition: *"… a knowledge base is a centralized repository for information: a public library, a database of related information about a particular subject […] is a machine-readable resource for the dissemination of information, generally online or with the capacity to be put online […] is used to optimize information collection, organization, and retrieval for an organization, or for the general public."*

Another definition more focused on the Artificial intelligence field is:
"… a dynamic resource that may itself have the capacity to learn, as part of an artificial intelligence (AI) expert system … According to the World Wide Web Consortium (W3C), in the future the Internet may become a vast and complex global knowledge base known as the Semantic Web." [http://whatis.techtarget.com/whome/0,289825,sid9,00.html]

With the extracted records of information the intelligent agent will populate the database, which will turn into the Knowledge Base. The agent will use some heuristics (based on the constant keywords), the relationships of the database and their cardinality to know how to construct the records to populate the database.

The next pictures will help to visualize this step:

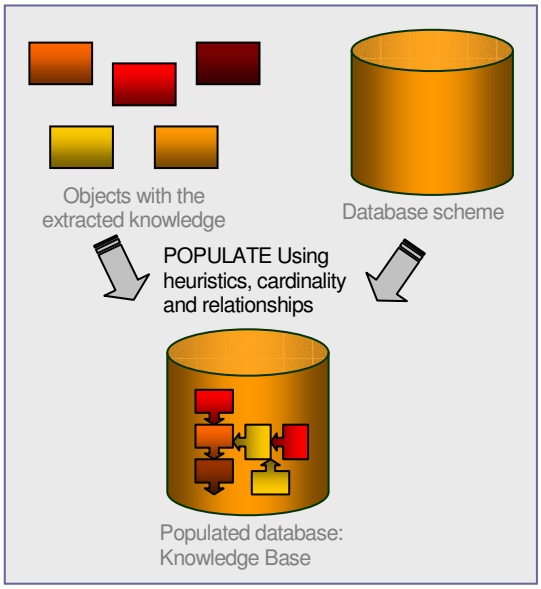


Figure 31 - Database Population

This picture shows how to obtain the knowledge base. The database is populated with the pieces of information recognized form the texts.

25.7 Query the knowledge base

Finally, once having all the desired knowledge properly structured and related in the knowledge base, the unique step left is to query it with a proper query language. The way of

performing the queries will depend on which kind of database is chosen; if it is a relational database it can be queried with a normal query language. If it is another kind of structured or semi-structured storage (e.g.: XML), it will have to be done with the appropriate queries (e.g.: XQuery), as explained in chapter 24.2.3.

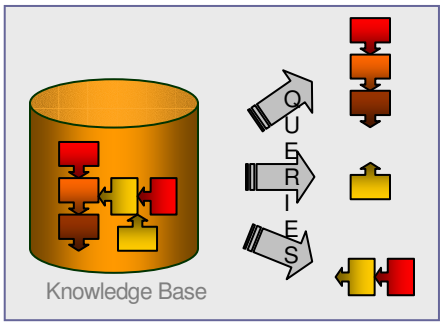


Figure 32 - Knowledge Base Query

The user will obtain the data he/she desires in and its relationships with other data.

26 Tool-based vs. Program-based

Two ways of design and implementing the ontology-based information extraction approach from semi-structured web pages were considered. Both of them were carefully studied and the most suitable one was chosen. They are described in the next sections.

26.1 Program-based Design

One possible approach to develop the ontology-based application it to make a program that analyzes the input texts following an Ontology. The steps are these ones:

- Make a program that parses lexically and semantically the HTML input documents following the Ontology, identifying instances and their relationships with the help of the data frames.
- Create a database following the Ontology schema.
- Extract and store the identified knowledge in the database.

26.2 Tool-based design

When building an Ontology from the first time, a deep study of the several ontology tools and ontology development languages has to be made. Because of the immature of this field, several problems had to be faced. Let me quote a sentence that perfectly defines all these problems:

"Various methodologies exist to guide the theoretical approach taken, and numerous ontology building tools are available. The problem is that these procedures have not coalesced into popular development styles or protocols, and the tools have not yet matured to the degree one

expects in other software practices. Further, full support for the latest ontology languages is lacking" [XML.com: Ontology building: A survey of Editing Tools]

That is why before choosing any Ontology tool; I studied very carefully the table with the ontology survey results. [Appendix-4]

26.3 Selected approach: Tool-based design

After an in-depth study of both approaches, the tool-based approach was chosen.
The reasons are the following:

- With the program-based approach all the steps have to be done manually, having to start from scratch. No auxiliary data, structures, storage, etc can be used as support.
- Many people experienced in this field have been working a lot of time on their tools, and they are not even finished. So, if I began to program another tool from the very first stage, no new results will came up; and no new contributions will be made to the Semantic Web.
- Instead, surveying and studying the available tools on the semantic web field and IE field, if I manage to combine them properly, and improve some of their features, new results and approaches can be obtained to contribute to the AI applied to the Semantic Web.

That is why, after thinking about it long and hard, the program-based approach was finally discarded. Anyway, this approach was regarded and carefully studied in depth. An example of how to implement this approach is explained in detail in the [Appendix-11]. It may be useful for guidance in future Ontology-based extraction projects.

Next chapters are a detailed explanation of all the reasoning process followed to select the different tools. This will clarify the decisions made, and the reasons of why the chosen tools where selected among all the other available tools.

27 Ontology Editor Selection

The first step to design the Ontology is to find the most suitable Ontology editor. This chapter presents an overview of the current Ontology editors, a comparison of their characteristics and the selected editor.

27.1 Ontology Editors Overview

First of all, an Ontology editor needs to be chosen. There are several Ontology editors available nowadays. Some of them are commercial programs; others are Universities initiatives and investigation projects. Those were the ones surveyed. These are free and sometimes open source, but have the disadvantage that they are still under development and therefore may have some lacks and bugs.

27.1.1 Ontology Editors Characteristics (Survey)

The task of choosing the right tool is a big effort, because there are several different editors (Ontolingua, Ontosaurus, WebOnto, Protégé2000,OilEd, OntoEdit,WebODE, etc…) also tools for merging ontologies (Chiamera, PROMPT), tools to translate Ontologies into ontology languages (Ontomorph) and to annotate web pages with ontological information (OntoMat, SHOE knowledge annotator, COHSE,etc). The problem is to choose the one that has the best functionalities for this project.

A detailed survey I found on the net is shown in [Appendix-4]. This survey, along with all the articles I read about several tools [6,22,23,31,32,33,34,37] and some research of the functionalities of each one, was the way of selecting the most suitable Ontology editor for my project.

A dozen of tools were firstly selected in the first overview. Their performance, importing/exporting features, degree of reliability, easiness of use, documentation available and web-orientation, information extraction and merging were some of the important features taken into account when surveying these tools.

After a careful study of all these features, most of these tools were discarded for several reasons and only two were left. These are: *Protégé* and *WebODE*

- Both support multiple-inheritance
- Both allow more or less the same features about the relationships between concepts (classes and instances), their attributes, the taxonomy, etc.
- WebODE allows multi-user support while Protégé does not.
- Information extraction is allowed in WebODE while not in protégé
- WebODE has an online database in a server available via API or Web browsing, while protégé is a standalone although some plug-ins can be added to it.
- The other characteristics can be consulted in the survey [Appendix-4]. and are almost the same.

27.1.2 Selected Ontology Editor: WebODE

The main reason of selecting WebODE is the online access. It is not a standalone application, but an online, multi-server environment. These are the main characteristic I was looking for in this project: the possibility to store the Ontology in a server, populate it and access to the subsequent knowledge base via a server.

The Ontology model is stored in a server in a relational database in Oracle DBMS. This server is located in: http://webode.dia.fi.upm.es/webode/jsp/webode/frames2.jsp?ontology_name=recipes .Besides this characteristic, WebODE editor is a very complete tool, with a lot of interaction possibilities and an own methodology approach called METHONTOLOGY (which is deeply described in [Appendix-3])

Until now, a few methodological approaches for the Ontology context have appeared. The most important ones are the Uschold's methodology (Uschold & Gruninger 1996), Grûninger and Fox's methodology (Grüninger & Fox 1995 and 1994) and METHONTOLOGY (Fernández, Gómez-Pérez & Juristo 1997 and Gómez-Pérez 1998)

METHONTOLOGY [Appendix-3] is the methodology created for WebODE to facilitate the creation of Ontologies though all their life cycle. It states which activities should be performed to get a complete and correct Ontology, rather than leave it to de developer criteria, which can lead to chaotic designs.

This project follows this methodology, due to the Ontology design has been made with the Ontology editor WebODE. Some intermediate representations of the knowledge in the Ontology can be generated which allows a better comprehension of the model. Some of them are presented below.

27.2 Interaction with other systems

With the Ontology editor the Ontology skeleton can be designed and stored. Afterwards the Ontology has to be populated with the desired instances. The population can be done manually trough the web browser, but this is not the objective of this project, otherwise the database has to be automatically populated. Furthermore, the instances that are going to populate the database can not be chosen at random, otherwise this project pretends to extract this information from unstructured web pages in an automatic way. The instances automatically recognized have to be automatically introduced into the database to populate the Ontology. This is the biggest challenge of this project.

Once the Ontology editor features have been stated, it is time to seek out a compatible way for extracting the desired information. This has been done having in mind the compatibility features of WebODE (import/export features):

Import from: XML, RDFs, DAML+OIL, UML, OWL
Exports to: XML, RDFs, Prolog, X-CARIN, OIL, Java/Jess, DAML+OIL, UML, OWL.

It also incorporates an API through which other applications can access to the Ontology.

28 Extract information from the web

This phase corresponds to the information extraction that will populate the database

28.1 Different IE Algorithms used for IE

There are two kinds of methods to extract information from texts:

- Probabilistic IE methods, which base on probabilistic.
- Symbolic IE methods, which base on the context.

28.1.1 Probabilistic IE methods

Some examples of the probabilistic methods are the hidden Markov models (HMMs) and the maximum entropy models (MEMs)

- maximum entropy models (MEMs) [A Survey of Methods and Results in Maximum Entropy Models, Viren Jain, CIS 520, Fall 2002, University of Pennsylvania]

The maximum entropy models are based on the next principle: *"The best distribution of some set of events is the one that maximizes the entropy (uncertainty or randomness) of all the distribution, basing on previous know information about the distribution"* [Jaynes, 1957]
The next formula formalizes this concept, being p the distribution and H the entropy and *x* represents events in a particular model.

$$H(p) = -\sum_{x} p(x) \log p(x)$$

This distribution is the one that does not make any implicit assumption (which can be incorrect) about the data of the distribution. It selects the most ambiguous model not to make incorrect assumptions.

These models can be used to select the most suitable set of characteristics that model a certain data, which can be applied to *document classification* and *part of speech tagging* [consult glossary]

This model has the disadvantage that is very likely of data over-fitting [consult glossary]. Some other complementary techniques have to been complemented.

- Hidden Markow models (HMMs) [A Survey of Methods and Results in Maximum Entropy Models, Viren Jain, CIS 520, Fall 2002, University of Pennsylvania]

Is based on probabilistic finite state machines. It infers knowledge which makes the system to transition to another state, where new knowledge can be infered.
They can be applied to language comprehension problems (NLP)

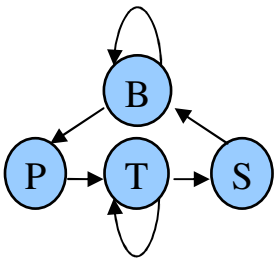


Figure 33 - Finite State Machine for IE

The Finite State Machine (FSM) consists of a set of states and transitions. It goes from one state to another one each time a word is generated.

Each state has a transition distribution (a function with the probabilities of the next state) and a word generation distribution (word probabilities)

It has algorithms that determine the probability of text generation and which is the most probable sequence of states to generate that text.

There are four kinds of nodes (states) in the machine:
B= Background nodes, which generate words of no interest for the domain, for example all the HTML tags (it is comparable to the pool of negative examples in the LP2 algorithm)
T= Target. These nodes generates the words we want to extract from the texts (it is comparable to the positive pool)
P=Prefix. These nodes generates the typical words that precedes the target in the text
S=Suffix. It is like the prefix nodes, but after the target. (Prefix and suffix are comparable to the contextual rules in the LP2 algorithm)

Some results obtained over 100 annotated texts with this method, extracted from http://www.cs.vsb.cz/dis/prispevky/20040122/ie_hmm_dis04.pdf show the following performance rates:

Recall varies from 69.0 to 99.1 and precision from 63.5 to 93.7 (these concepts are explained in detail in Appendix-9)

28.1.2 Symbolic IE methods

These are other kind of algorithms that induce the extracted information following context-based rules; some examples are the LP2 algorithm or Rapier although more methods exist in this field.

- Rapier

  Bases on candidate classification.Combines the bottom-up(restrict the *contents*) and the top-down(add restrictions based on the *context*) learning. Uses part-of-speech tagging [see glossary]. Induces rules with three patterns: pre-filler, filler, and post-filler patterns. A rule can match many times some elements in a document.

  It starts with one rule for each example and then it generalices over them. Then it goes the other way specializing the rules.
  This algorithm expects high-precission and low-recall [see Appendix-9] :
  Over a set of 150 examples it obtains rates of: 90% of precision, and 65% of recall.

- Learning Pattern by Language Processing(LP^2)

Its author, Fabio Ciravegna, claims that the LP2 algorithm out forms many NLP-approaches (Rapier, and Whisk) and also many non-NLP approaches (BWI, HMM) in terms of accuracy

This algorithm obtains the following rates:
Precision over a set of (only) 24-40 examples: between 90 and 100
Recall over a set of 100 examples: between 50 and 80

These rates were obtained from an study over a corpus of 250 annotated documents.
The parameters of precision and recall are higher and stable when we annotate more than 20-40 documents, while the other algorithms expoded need around 100 texts in the intial corpus to reach a good performance.

As this method seems to be the most suitable for the IE in the web context, next chapter will explain how this algorithm works:

28.2 Learning Algorithm: LP^2 algorithm (Learning Pattern by Language Processing)

28.2.1 Overview

The Information Extraction Tool (Amilcare) uses a symbolic IE method: the LP^2 algorithm. This LP^2 algorithm uses NLP to deal with data sparseness while treating free texts and it also gives effective results while parsing highly structured texts.

It is a web-related IE algorithm. Induces symbolic rules from a corpus annotated with SGML (Standard Generalized Markup Language) tags [see Glossary]

The creator of the algorithm (Favio Ciravegna) claims that it overcomes any other known algorithm tested on the same corpora.

28.2.2 How it works

The explanation of how this algorithm works is a little dense, so it has to be separated in the [Appendix-10] not to cloud the explanation of the project.

28.3 Selected IE tool

The tool that implements the LP^2 algorithm is called Amilcare. That is why this tool has been selected. It is one of the most renowned tools for the purpose of the IE

Amilcare relies on Gate (University of Sheffield) for preprocessing the input texts. It makes tokenization [see glossary], sentence identification [see glossary], and part of speech tagging [see glossary] and gazetteer lookup (it defines by default its own gazetteer with common elements of the real world)

The IE process is made in three stages: Training, Testing and Production.

28.3.1 Training mode

This phase needs the following inputs:

- A corpus already annotated with the information we want to extract (this information is annotated via SGML tags). This is the only language dependent part. The rest of the stages are language independent.

- A scenario, which can be a list of recognizable tags or an Ontology.

Amilcare provides as output a set of rules that can be applied to new texts of the same type in order to annotate them. These rules are induced by means of the LP^2 algorithm explained before.

28.3.2 Testing mode

Uses the test rules produced in the training mode on an untagged corpus of documents and checks how well they perform the annotation.

This is done by removing all the annotations from the training corpus, and then applying the inferred rules to the un-annotated corpus. The IE process re-annotates the corpus again comparing it with the initial annotated corpus.

The output of this phase is the corpus re-annotated along with some statistics obtained when comparing both corpuses: Precision, recall [see Appendix-10], and mistakes made by the program.

The user can interfere and revise the testing; this is why this IE approach is a supervised learning.

28.3.3 Production mode

This is the phase when the application is released. It takes the documents the user wants to annotate as the input, and gives as output: the documents annotated, several performance rates (precision and recall) [see Appendix-9], and a list with the extracted information (pairs of constants and values)

29 How to Annotate the Training Corpus

29.1 Annotation Tools Overview

The most know annotation tools are listed below: Annotea*, Annozilla, Melita, GATE, Semantic Markup Plug-In for MS Internet Explorer, Briefing Associate, OntoMat Annotizer,*

SMORE, Yawas, Semantic Word, SHOE Knowledge Annotator.

Some of these annotation tools pretend to annotate web pages with external annotations, like explained in chapter 10.1.1 (Annotea and Annozilla for example). This is not the need of this project. The annotation is only an auxiliary method to mark the input text in such a way the IE tool is able to recognize the relevant instances the user wants to extract.

The selection of the annotation tool is a more restricted task, because it has to meet the necessities of Amilcare. The selected annotation tool is a mechanism to help the user in the task of manually annotating the input training corpus with SGML tags.

Some freeware annotation tools from the Sheffield University (like Amilcare) were surveyed for this task. Those are: Melita, MnM, Gate, Mitre's Alembic, and S-Cream.

MnM and Melita are similar except of MnM displays some additional features (but not so relevant). MnM was up to recently unable to save the annotations, so it was unusable. S-Cream is very similar to MnM. Gate is also an annotation tool pretty solid and mature.

All these annotation tools are developed by Sheffield's University. They were regarded but finally Melita was the chosen one. The advantages of this one are that it interacts directly with Amilcare. Due to this connection it can be considered more that Annotation tool; it serves as an interface for the IE process, where the recognized entities are presented to the user and the he/she can interacts and improve the IE task.

29.2 Selected Annotation Tool: Melita Overview

This Annotation tool is supposed to allow the user to make annotations on the texts in an easy and intuitive way. These annotations are made highlighting the relevant words (the elements the IE tool should extract from the text).

29.2.1 Interaction with the IE tool

Besides annotating the texts with XML tags, the annotation tool also interacts with the IE tool. The process is related below:

After setting the annotation tool, the user will be ready to annotate the texts using the annotation GUI.
The IE tool is running all the time in the background, examining the documents. Each time the user makes an annotation, the IE tool detects it and it will use that new annotation to induce new annotation rules (As explained in chapter 28.3)

This is done during all the annotation cycle. The interaction with the IE tool is not optional, it is done each time the annotation tool is executed and it can not be set to not perform this action.

29.2.2 Different ways of extracting Information: Automatically, or semi – automatically

The chosen approach is a semi-automatic annotation and extraction of information.

I have manually annotated a significant set of web sites following the developed ontology. Afterwards an Information Extraction tool can infer rules from these annotated texts in order to automatically extract information later on.

As long as the user has to annotate the texts and can revise and supervise the learning this is called a semi-automatic approach.

In the automatic approach the user does not have to supervise the learning, but much more number of tests has to be done in order to get good extraction results. In the semi-automatic approach we get earlier to the desired levels of accuracy, just because of the user corrections that make the learning retrained.

29.3 How to populate the Ontology with the Extracted Information?

Finally, after running the IE process, we will obtain the entities it was able to recognize. Then these entities have to be structured and then the database will be populated with them.

After retrieving the recognized entities, the program hast to relate them following the Onotology and then it populate the database with this knowledge, getting then the knowledge base. The details of these methods are explained in the implementation chapter.

30 Final Overview: Tools Interaction

Table

Purpose (Function)	Tools	Developed by	Purpose
Ontology editor	WebODE	Technical University of Madrid (UPM)	Create the ontology
Information Extraction tool	Amilcare	Sheffield University	Extract information from any kind of documents
Annotation tool	Melita	Sheffield University	Annotate texts

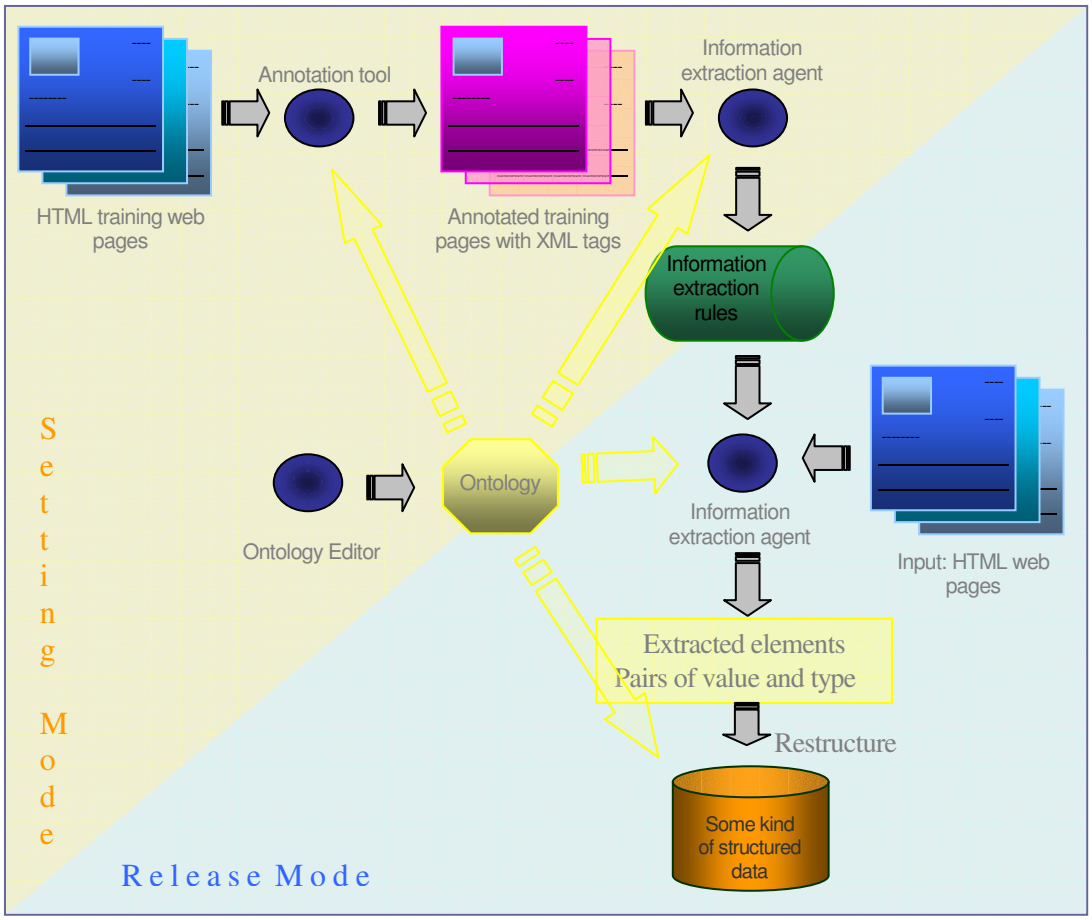


Figure 34 - Final IE Overview

The picture above tries to explain graphically the parts of this project:

Train the system

- Annotate an unstructured set of HTML documents with semantic metadata
- Infer rules from these annotated documents

Release the system

- Extract the information from the texts with the annotation rules
- Store this information in a structured way in the Knowledge base.

The ontology guides all the steps of the information retrieval and structuring.

X System Design

The design of the system is explained in this chapter, divided in several main parts: the initial part of configuring the system, the part of releasing the system, then query the system and finally consolidating the database.

31 Configuring the system

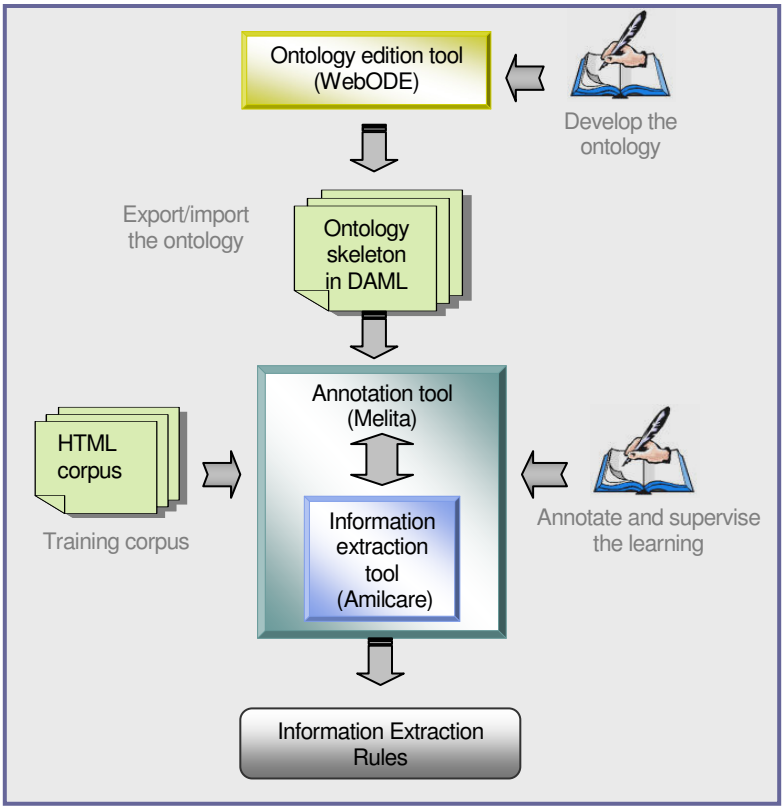


Figure 35 - System Configuration

Steps:

1. Edit the ontology with the Ontology editor and export it to DAML (this language has been chosen because both tools support it)
2. Annotate the training set of HTML web pages with the help of the annotation tool, and give this corpus as the input for the IE tool.
3. Import the DAML Ontology into the IE tool (it works with NLP and also following an Ontology it is given) and the annotated pages.
4. Get the information extraction rules

31.1 Edit the ontology with the Ontology editor and export it to DAML

31.1.1 Choose the most appropriate ingredient taxonomy

Now that it is time to implement the Ontology, the general ER diagram shown in chapter 19 has to completed with the proper ingredient taxonomy (this task had to be postponed until knowing some implementation features)

As I have explained before, the annotation tool and the IE tool do not support the attributes in the model, so they had to be removed from the model and then swapped into entities, in order these tools can recognize all the important features and work with them. The final Ontology skeleton (modeled in an ER diagram) is shown in Appendix-15 together with an explanation of the conversion.

This classification has an intermediate level of detail, just to make the annotation and IE processes feasible. The final ingredient description is shown in [Appendix-7].

31.1.2 Edit the Ontology

The Ontology is edited in WebODE. This tool has a graphical interface that facilitates this task. Firstly all the entities are modeled, and then the relationships among them are inserted. The next picture gives an idea of the process:

Figure 36 - Ontology edition in WebODE

31.1.3 Intermediate Representations of the Ontology

This chapter describes the implementation of the Ontology in WebODE editor, following its methodology (methontology)

31.1.3.1 A Concept Dictionary

Contains all the Ontology concepts, their instances, their attributes (class and instance attributes), the relationships where the concept is the source element, and the synonyms and acronyms defined by the developer.

31.1.3.2 Binary "ad-hoc" relationship description

Shows the classification tree of each "ad-hoc" relation. It shows its name, the name of the source concept, the name of the target concept, etc….

31.1.3.3 Instance Attribute Table

It lists all the instance attributes of the Ontology.

31.1.3.4 Class Attribute Table

It is a list of all the class attributes that exist in the concept dictionary.

31.1.3.5 Logical Axioms Table

All the axioms are described showing its name, natural language description, the concept it belongs to, and the expression of the axiom in FOPC (First Order Predicate Calculus)

31.1.3.6 Constants Table

Lists the name, natural language description, value type, constant value, measurement unit (only for numeric constants) and attributes that can be inferred of each constant.

31.1.3.7 Formula Table

It shows all the formulae of the attributes of the Ontology. It lists the name, mathematical expression, natural language description, attributes and constants used to calculate the formula, its accuracy and the when the formula can be used.

31.1.3.8 Attribute Classification Trees,

It related attributes and constants and their formulas.

31.1.3.9 Instance Table

It lists the instance name, its attributes and their values for all the instances in the Ontology.

Some of the intermediated views of the developed Ontology are shown in the Appendix – 17.

31.1.4 Improving the Ontology: Merging the developed Ontology with new ones

Introduction

To improve the quality of this project, the developed recipes Ontology has been extended merging [see merging theory in chapter 10.3] it with a wide wine's Ontology. This has been found in the DAML library [http://www.daml.org/cgi-bin/hyperdaml?http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml] this initiative to share knowledge over the Web has been described in detail in chapter 23.3.3.1 Reusing Ontologies: Merging

All the list of available Ontologies has been carefully studied looking for some that could complement the recipes domain. The wines Ontology is a good one; it does include a long knowledge description for wines and meals. Both Ontologies can be merged because they have several concepts (about the same subject) in common.

The wines and the recipes Ontologies can be considered as Local Ontologies [see chapter XXX] as they both refer to specific domains. Merging them (and maybe someone else) a Regional Ontology can be formed, referring to a wider context (*drinks and food* for example).

Also some upper Ontologies have been found in [http://www.daml.org/cgi-bin/hyperdaml?http://reliant.teknowledge.com/DAML/SUMO.owl] (Standard Upper Merged Ontology proposal to the IEEE Standard Upper Ontology effort), but an in-depth investigation has to be made in order to refer the recipes Ontology to an Upper level Ontology, this is a very complex task. As the merging process is out of the scope of this project, only one merging has been made. This project-extension pretends to show how the merging process is and how important is to reuse and merge existent information.

The source code of this Ontology can be found in the Appendix-18
It is written in DAML but it does not present any problem, as the chosen Ontology editor can import and merge Ontologies in a wide range of languages (being DAML+OIL one of those)

Merging method

The merging process has to be guided and supervised by an expert, because it is necessary to provide additional semantic information besides the two Ontologies to merge.
This is because the Ontologies normally do not have the same structure and not the same words are normally used to design the same concepts.

The merging process of two Ontologies in WebODE (with ODEMerge [explain or reference] service) is based in additional semantic information provided by the developer. It is necessary to provide some mechanism to compare related concepts. It is necessary to provide as input:

- Two Ontologies to merge
- Additional semantic information:
    - A synonym table, which contains the synonyms between both Ontologies.
    - A hyperonym table,[see Glossary] SEE GLOSARY (Hyponym) which contains the hyponyms (subclass relationship) concepts among both Ontologies.

This additional information guides the merging service, stating how to connect both Ontologies. The resulting output is a new ontology with the concepts of both related.

All the related work, the recipes Ontology skeleton, the wines Ontology skeleton and explanations, the synonym table, the hyperonym table and the result Ontology can be found in [Appendix-5]

With this new Ontology the one developed before for the recipes purpose is extended and enriched. Some entities missing or incomplete in both Ontologies can be complemented with the merging. So merging Ontologies is a good method to make it easier to make a complete Ontology.

The merging service currently merges only concepts, not the other components of the ontology (like the attributes, axioms, synonyms, etc).These knowledge is added to the new ontology, as long as the element/s they belonged to is also present in the new taxonomy.

The last step is to fill the Ontology with the instances of each entity. It could be done by hand by the developer, but the main objective of this project is to populate the Ontology automatically by retrieving online information. After filling the Ontology with all the instances the IE system is able to recognize, it will finally be the knowledge-base, where all the information is perfectly defined and related.

31.1.5 Ontology Evaluation

The Ontology editor incorporates a mechanism to evaluate the designed Ontologies. It is called OntoClean. OntoClean is an evaluation methodology that analyzes other Ontologies stating whether they are correct or not. It was developed in the UPM by N. Guarino and C. Welty. It is based on philosophical notions that are used to analyze the conceptual model of an Ontology.

The recipes Ontology has been evaluated with this methodology and the following information was obtained: "***Synchronous Evaluation OntoClean for recipes 0 ERRORS FOUND"***

This means that the Ontology is well formed following *Methontology* and has no errors in its design.

31.1.6 Conclusions

METHONTOLOGY is a methodology that guides the creation of Ontologies in an incremental way trough all its life cycle. It states to firstly create the terms of the Ontology (concepts, instances, properties, etc) structuring them afterwards with the relationships, axioms and formulas to create the taxonomy. It also verifies and validates the ontology.

31.1.7 Export the Ontology skeleton to DAML

It is just automatic to convert the developed Ontology to DAML. The Ontology editor incorporates an option to export it. This language has been chosen as the intermediate one, because both the Ontology editor and the IE tool support it. In Appendix-16 this document is shown.

31.2 Annotate the training set of HTML web pages with the help of the annotation tool, and give this corpus as the input for the IE tool.

31.2.1 Annotate the input corpus with Melita

To set this annotation tool is necessary run two files, a Melita client, which comprises the GUI, where the user can interact with the application, and a server file, which is the one that connects with Amilcare.

The inputs necessary to run the annotation tool are the following:

- A training corpus: These are some representative files of the domain. This is not included in the Appendix due to its big volume. Please refer to the enclosed CD to consult these files. Melita specifications state that these files can be HTML files, but a lot of bugs were found when working whit them, from a bad performance to loading errors. So the input files given had to be plain text documents to avoid this inconveniences.

- An Ontology: The skeleton of the Ontology that is going to guide the annotation process. It has to be a very simple ontology, without attributes or relationships, because the tool does not cope with them.

  It only annotates instances of the Ontology concepts. This is why the Ontology had to be remade several times to adapt it to this tool (and the IE tool as well). All the attributes had to be remodeled into concepts related to the one they belong. The new Ontology model is shown in [Appendix-13]. This is finally the last domain model made in this project.

  This tool has another inconvenience about the Ontology language, it is not any standard one (quite the opposite than the IE tool that can import a DAML Ontology),

it does not support any web-oriented language (as it would be desired), and so the Ontology had to be remade by hand into a specific Melita language. The picture below shows the Ontology:

Melita's Ontology
things(X) ==> concept(X) v relation(X). concept(X) ==> general_features(X) v ingredient_description_part(X) v way_of_doing(X) v nutritional_value(X) v course(X). general_features(X) ==>name(X) v retrieved_from(X) v posted_by(X) v price_person(X) v difficulty(X) v cooking_time(X) v number_of_servings(X). nutritional_value(X) ==> fats(X) v carbohydrates(X) v cholesterol(X) v proteins(X) v calories(X) v good_for(X). ingredient_description_part(X) ==> ingredient_description(X). ingredient_description(X) ==> ingredient(X) v quantity(X) v measure(X). ingredient(X) ==> food(X) v drink(X). food(X) ==> fish(X) v dairy_produt(X) v vegetable(X) v fat_oil(X) v cereal_grain(X) v egg(X) v meat(X) v fruit(X) v miscellaneous(X) cereal_grain_based(X). vegetable(X) ==> spice(X).

In this notation the only features that can be specified are the entities and their relationships (it is not possible to specify the kind of relationship between two entities). No attributes can be defined.
Relationships can be specified in this notation, the picture below shows an example:

Relationships in Melita's Ontology
things(X) ==> concept(X) v relation(X). relation(X) ==> IS-A(X) v is-part-of(X) v is_made_of(X) v consists_of(X) v etc …

Although the relations can be defined in this language and they appear in the left part of the screen, and moreover; although it is stated in the User Manual and other documentation; after several attempts it was found out that it is not possible to annotate the texts with the relationships.

After several investigations, Melita's developers admitted that is impossible to perform this action with the annotation tool. As they explained, the first developer of this tool thought it would be easier to make a heuristic to annotate any kind of relationship, but afterwards he realized of the difficulty of this task and he dropped it. Unfortunately it is still said in the documentation provided with the tool and so, confusing the user. This is a mistake that should be immediately removed from the documentation.

- Accuracy levels:

These parameters can be set up by the user wants. They have two different scopes; they can be defined globally, for all the entities of the Ontology, or can be set locally giving different values of accuracy to each element in the Ontology.

- Annotations: This is the third input needed to run the annotation tool. The annotations are made manually by the user, who highlights the words he/she wants to identify in the text. This is done very easily dragging an Ontology concept from the left part of the screen and dropping on the word/s to be identified.

The picture below shows an annotated recipe with Melita's annotation tool to clarify this process:

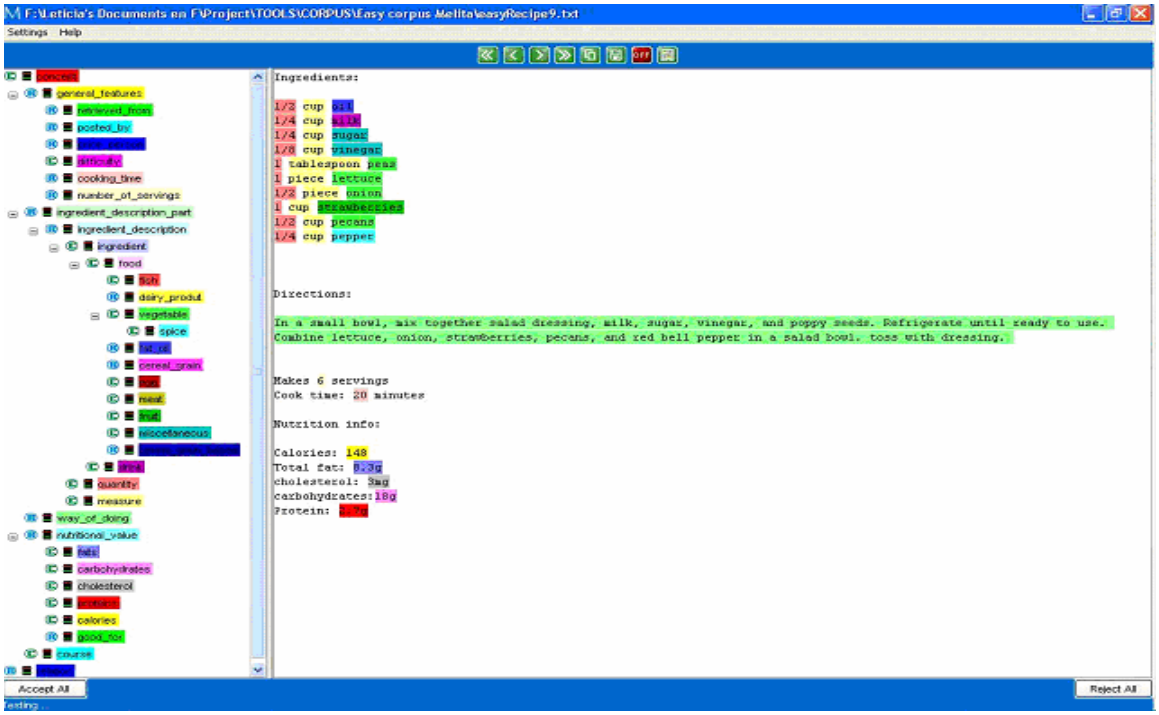


Figure 37 - Annotation tool

As seen in the picture, the GUI is very intuitive and allows users without experience to highlight concepts in the texts.

The Onotology is displayed in the left part of the screen, and the documents to be annotated can be consecutively displayed in the right part.

31.2.2 Output

31.2.2.1 Annotated texts

Each time concept is annotated by the user, the annotation tool performs this action: It places XML tags just before and after the concept in the recipe's text. The first tag is an XML opening tag (<example_tag>) and the second one is a closing XML tag (</example_tag>).

The text of the tag is just the name of the Ontology's concept the user wants to relate the highlighted text with.

For example, is the user wants to identify the word *milk* as an instance of the concept *dairy_product*, he/she will drag and drop the (dairy) concept on this word. Automatically the tool will place this tags in the recipe's source:

<dairy_product>milk</dairy_product>
Next picture shows the annotated source of the:

Annotated recipe's source
Ingredients: <quantity>1/2</quantity> <measure>cup</measure> <fat_oil>oil</fat_oil> <quantity>1/4</quantity> <measure>cup</measure> <drink>milk</drink> <quantity>1/4</quantity> <measure>cup</measure> <miscellaneous>sugar</miscellaneous> <quantity>1/8</quantity> <measure>cup</measure> <miscellaneous>vinegar</miscellaneous> <quantity>1</quantity> <measure>tablespoon</measure> <vegetable>peas</vegetable> <quantity>1</quantity> <measure>piece</measure> <vegetable>lettuce</vegetable> <quantity>1/2</quantity> <measure>piece</measure> <vegetable>onion</vegetable> <quantity>1</quantity> <measure>cup</measure> <fruit>strawberries</fruit> <quantity>1/2</quantity> <measure>cup</measure> <vegetable>pecans</vegetable> <quantity>1/4</quantity> <measure>cup</measure> <spice>pepper</spice> Directions: <way_of_doing>In a small bowl, mix together salad dressing, <drink>milk</drink>, <miscellaneous>sugar</miscellaneous>, <miscellaneous>vinegar</miscellaneous>, and poppy seeds. Refrigerate until ready to use. Combine <vegetable>lettuce</vegetable>, <vegetable>onion</vegetable>, strawberries, <vegetable>pecans</vegetable>, and red bell <spice>pepper</spice> in a salad bowl. toss with dressing. </way_of_doing> Makes <number_of_servings>6</number_of_servings> servings Cook time: <cooking_time>20</cooking_time> minutes Nutrition info: Calories: <calories>148</calories> Total fat: <fats>8.3g</fats> cholesterol: <cholesterol>3mg</cholesterol> carbohydrates:<carbohydrates>18g</carbohydrates> Protein: <proteins>2.7g</proteins>

The XML tags is just a way of representing the annotations, that both the annotation tool and the IE tool understand (the IE tool is capable to recognize SGML tags in the input texts). But it is only a kind of representation. Although there are many other ways to represent annotations in texts this is a good way of representing semantic information, because one of XML's purposes is to represent semantic meaning in the web.

31.2.2.2 Annotations vs. suggestions

These are the results the IE tool (Amilcare) provides to the annotation tool. The IE tool infers rules from the user annotation, and then it is able to apply that rules to the texts.

The difference between annotations and suggestions depends on the accuracy levels the user sets. If a rule gives a good performance, above the certainty level, then all the words recognized by that rule are automatically annotated in the training corpus without even asking the user. The rules with a performance between the suggestion level and the certainty level are presented to the user as suggestions. Then the user can decide whether that annotations are correct or not; this information is used again by the IE tool to retrain getting to be more precise (this is the advantage of a semi-automatic approach, the user can interact correcting the IE process; so it learns faster and with less number of training corpus). The words that fit with IE rules with a lower performance that the suggestion level, are obviated and not presented to the user.

Figure 38 – Annotation Intervention Level

31.2.2.3 Gazetteers

Below is shown examples of the gazetteers generated by the annotation tool. A gazetteer is a list of the concepts recognized in the texts; along with the number of times that concept has appeared in the texts. It corresponds to the lexicons in the Ontology theory (see chapter 25).Each time a new sequence of words is annotated as an instance of a concept; this is automatically entered into its gazetteer. If this instance is found again, the number of occurrences is augmented.

The utility of the gazetteers is to recognize words in the texts. Each time a word or sequence of words is recognized by the gazetteer this is annotated by the tool.

Measure's gazetteer

```
<xml version="Melita">
 <concept name="measure">
  <element occurence="2">pound</element>
  <element occurence="1">ounce can</element>
  <element occurence="5">teaspoon</element>
  <element occurence="1">cloves</element>
  <element occurence="1">can</element>
  <element occurence="1">cloves</element>
  <element occurence="2">teaspoons</element>
  <element occurence="2">can</element>
  <element occurence="1">bell</element>
  <element occurence="2">tablespoon</element>
  <element occurence="3">tablespoons</element>
  <element occurence="3">glass</element>
  <element occurence="3">cup</element>
  <element occurence="2">ounces</element>
  <element occurence="1">tspns</element>
  <element occurence="1">tspn</element>
  <element occurence="0">kilo</element>
  <element occurence="0">kilogram</element>
  <element occurence="0">gram</element>
  <element occurence="0">liter</element>
  <element occurence="0">deciliter</element>
  <element occurence="0">centiliter</element>
  <element occurence="0">milliliter</element>
</concept>
</xml>
```

The rest of the gazetteers generated for all the Ontology's concepts are shown in the [Appendix-12] there are listed the gazetteers for the reduced domain, as well as some gazetteers generated for the complete domain.

Although the gazetteers are generated by the annotation tool, the user can create new ones or edit the existent. It is possible to add or remove concepts from any gazetteer to improve the performance of the IE task.

31.2.2.4 Final decisions about the annotation process

Not using the annotation tool because:

- ✓ It only supports little Ontologies with small levels of relationship.
- ✓ Although the Ontology was cut down to make it smaller and easier to use, it neither performs well with it.

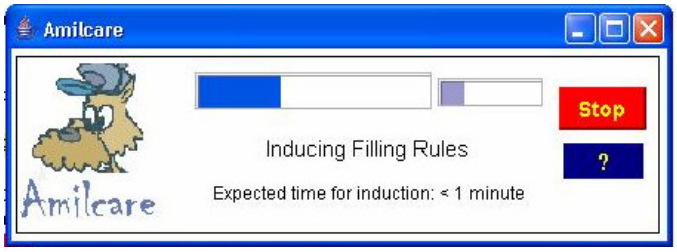
- ✓ The important process for this project is the Information Extraction task. The IE process can be performed alone, provided that, a correct training annotated corpus is given as an input.

31.3 Import the DAML Ontology into the IE tool and the annotated pages.

The recipes DAML Ontology is imported in the IE tool.

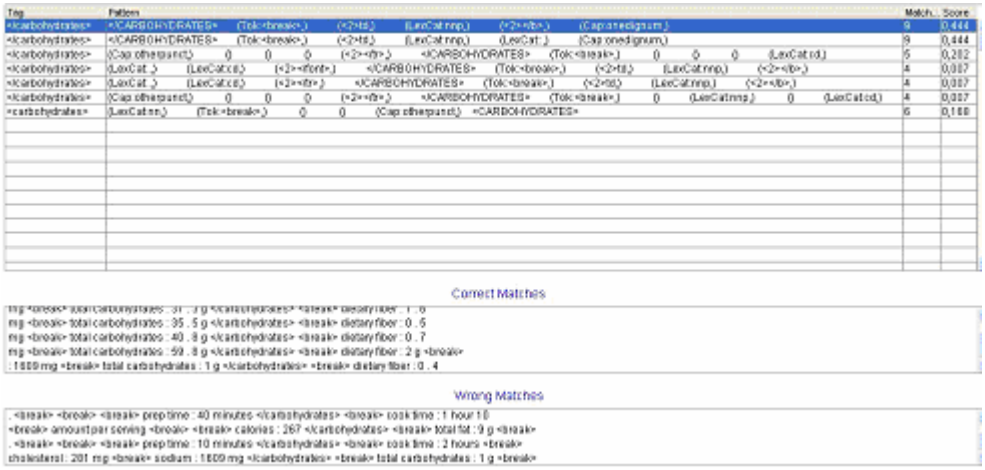
31.4 Get the information extraction rules

After setting the tool with the annotated corpus, the Ontology and some other parameters the system is run to extract the tagging rules.



This process is very slow. After several hours it finished (sometimes one day was needed to perform this action over a corpus of 25 ingredients, other times it never finished)

Then the tagging rules are induced by the IE tool and presented to the user. The next picture shows an example of these induced rules:



These tagging rules are automatically inferred by the IE tool using the LP2 algorithm already explained. The user can modify them in order to get more accuracy. Also with larger training sets and more use of the gazetteers the results are improved. The rules are internally stored in the IE tool, and can be accessed by the API in order to apply them to new texts.

32 Running the system

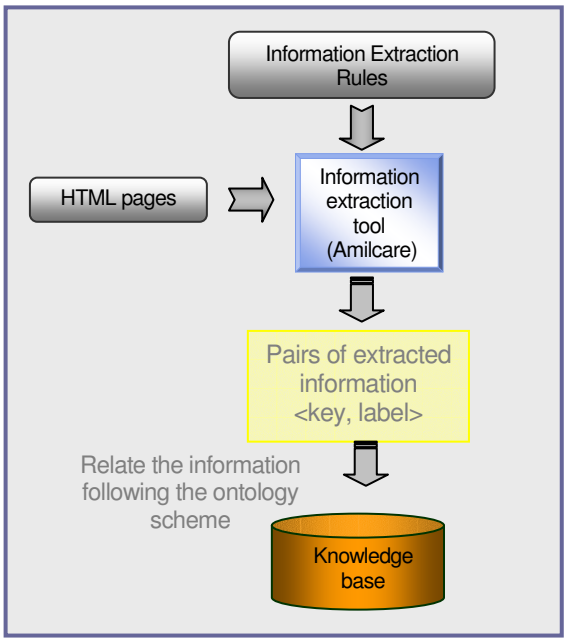


Figure 39 - System Running

Steps:

1. Input the HTML definitive corpus of web pages.
2. Run the IE system in the running mode, to extract the information from the corpus
3. Retrieve the information and then relate it following the ontology
4. Populate the database

Run the IE tool on another group of pages (the input corpus) the user wants to parse.
The output of the Information Extraction system is given in a text format. The entities it was able to recognize and extract are given in the format of pairs like: <key, value>
The system will automatically retrieve the output values, and it will relate them following the Ontology structure
The constructed records of related information will be automatically entered into the database in order to populate it.

32.1 Input the HTML definitive corpus of web pages

The new corpus is given to the system in order to extract information from it. It is not explained in detail as it is a very trivial process

32.2 Run the IE system in the running mode, to extract the information from the corpus

The IE tool now is trained and has learnt some IE rules. Then it is released on the new unseen corpus and it annotates the elements it is able to understand. An example of these new annotations is shown in the next picture:

This is the graphical output of the IE tool. The GUI shows the new texts highlighting the annotations the system was able to make. Some of them can be correct, some missing and some others incorrect or incomplete, it just depends on how well the rule performed.

32.3 Retrieve the information and then relate it following the ontology

The interesting feature is not to see the annotations; it is to retrieve the extracted information. Trough the API, the system developed can connect to the Annotation tool, train it, release it and then access to the recognized information. This information is given in a matrix of objects. The application has to be able to treat all the extracted entities, related them following the Ontology and then input them into the database.
This is a difficult task; all the entities have to be studied carefully. They are provided as a combination of two characteristics: filler and the tag. The tag is the annotation tag that highlighted the entity in the training phase; this is the class of the Ontology it corresponds to. The filler is the word/s the IE process was able to recognize as an instance of that class. Example: filler: tomato, tag: vegetable.

The main program has to study all the results, identify the tag, and be able to relate instances with others as it is stated in the Ontology.

32.4 Populate the database

Once the instances are treated, they can be inserted in the database. The “filler” is inserted as an instance of an element in the Ontology; this element is recognized by the “tag” that accompanies the element.

This operation is also made trough WebODE's API.. First of all a connection to the server has to be made, afterwards the program has to set some parameters to connect to the desired Ontology (as there are many Ontologies stored in this server). Then all the entities, relationships and other features in the Ontology can be accessed trough the API.

The program introduces the instances and also their relationships with other instances in the Ontology.

33 Consolidating the Database

As explained 19.3.2 the data extracted from the text is not normalized. It is provided in a text format by the IE tool. The types of the database should be the ones that reflect the reality (*quantity*: decimal, *ingredient*: string, *carbohydrates*: decimal, *number of servings*: integer, and so on). It has the additional problem of the non-standardize way of describing recipes in the net (explained in detail in chapter 19.3.1). These problems can be sorted in two ways:

Transform the type of the data before populating the database

After extracting the desired data from the Web, the program can transform it into its suitable type, so it can be entered into the well-formed database. But this is a very tedious task. Each time an instance is extracted, the program has to analyze which entity of the database it belongs to, and so transform it into its suitable value with the help of some auxiliary data.

Example: In the particular case of the entity *quantity* the program would have to deal with different formats like: 1, 2, ½, 1 and ½, 1.5, 1 1/3, etc. besides the data type problem, so a little parser is needed inside the program to analyze all the extracted elements.

Input the data as it is extracted and normalize the knowledge-base afterwards

The data is extracted as plain text, so all the fields of the database are stated as string type. The data is straight imputed into the database. After having it populated some methods to normalize everything are needed.
This is the approach followed in the project implementation. Some techniques have been invented to normalize the database in an easy way, avoiding the awkward task of studying and transforming each kind of information before entering into the database.

This approach consists of taken advantage of the XML characteristics. As long as the populated database is going to be transformed into a semistructured database implemented in XML, some additional files can be added then. The idea is to make some *transformation-files* that will contain all the data types and their conversions. This XML files can be contrasted with the Database files to transform and then normalize and convert all the data at once.

An example of these auxiliary files is shown in the Appendix-23.

34 Query the system

As the culmination of the project, the knowledge base can be treated in order to query it.

There are two possible ways to do this step. One is access to the data in the knowledge-base trough the Ontology editor API. Then all the desired queries can be made against the DB and retrieve this information. The information can be displayed to the user trough a web page (it can be written in many languages, JSP is the most suitable because of the API characteristics)

Another way to access this information is using the export module of the Ontology editor. In this way the whole knowledge base can be exported to some languages like UML, prolog, RDF, XML, OWL, etc.

If exporting the knowledge base to some unstructured web-oriented language, it will be easy to query, also these knowledge will be in a suitable form to be delivered trough the internet, and moreover, the structured database will be transformed into a semistructured database.

As long as this project is focused on the Semantic Web, this has been the selected approach: Transform the whole knowledge base to the XML semistructured, web-oriented language.

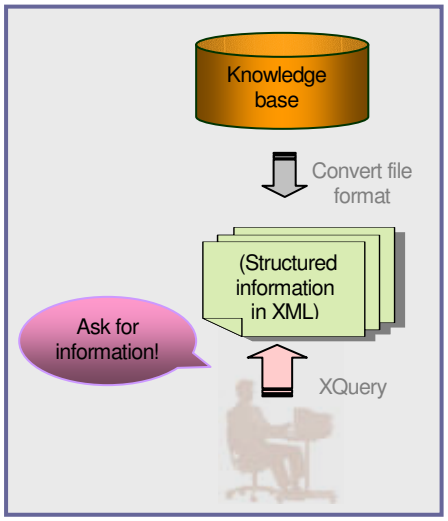


Figure 40 - System Querying

The XML editor exports directly to XML, referring to a DTD [see Appendix-21] they have design to make the structure of all their XML files. It is no possible to export the data to XML referring to another DTD or an XMLSchema.

There is another way of exporting to XML referring to a desired schema, but it is much more tedious. This is about creating the XML document by the program. The program can make the xml tags while treating the extracted elements. Instead of check the "tag" element and then introduce the "filler" in this Ontology's concept, it can create its own document. It can add the xml tags with the "tag" information and then fill it whit the "filler" information.

Example:

Recognized elements: tag: vegetable, filler: tomato
XML construction: <vegetable>tomato</vegetable>

This way of creating the XML file is much more beneficial as the developer can structure the knowledge in the way he/she considers appropriate. It also makes the query process easier as the document structure is much more logical than the one created with the Ontology editor. An example of how the final Ontology would look like is shown in Appendix-25

In spite of these advantages the final structure was made introducing the elements in the database and the exporting the files to XML, due to time limitations.

Once having the knowledge base in this appropriate semistructured format, the information can be accessed via a suitable query language (XQuery is XML's query language). The example queries made against the XML documents are shown in Appendix-22.

35 Problems faced -Connectivity problems

As many of the articles about the Ontology context reflect (and as I have experienced myself), the Ontology developers are concerned about the Ontology portability and interoperability between the different tools.

There is a need of a workbench to support the tree stages of the ontology life-cycle:

- Ontology Development tools to create, manage and populate the ontology.
- Ontology middleware to easily integrate the ontology in the information systems.

As I remarked before there is a lot of work to do in this field, as several problems occurred while developing this project. We are in need of:

- More interoperability between Ontology editors, some standards to export and import Ontologies between different ones.
- Tools to merge Ontologies developed with different ontology editors.
- The tools give support to design and implement Ontology, but support to test, maintain and evaluate Ontologies is needed.
  The objective is to create a workbench that helps the ontology developers in all the stages of the ontology design (ontology creation and edition, knowledge acquiring following this ontology, browsing the results, integration with other tools, import and export from/to different languages and formats, and merging with other ontology tools)
- Generic domain Ontologies to reuse in different domains and easily create new ones (Ontology tools that include ontology libraries)
- An Ontology methodology that guides the development of all the life stages of the ontology life cycle, along with scheduling, documentation, etc.

- An Ontology methodology to evaluate the ontology once we have created it.
- Middleware services to help with the use of the Ontologies. Software to decide which the best ontology for a certain project is, functionalities to query the Ontologies, integration with current databases systems, remote access to an ontology library and administration services.
- Formal metrics to compare different Ontologies and measure their similarities.

XI Implementation

36 What I have implemented

The system design also shows the implementation of the project. It has been made firstly running manually the tools to see how the performed and what kind of features, inputs and outputs they had. Afterwards all the tools where connected manually trough the APIs.

The main program is a Java process. This is a process codified in Java, because both APIs are written in this programming language as well.

The JRE (Java Runtime Environment) and the JDK (Java Development Kit) were needed to perform these actions. The JRE enable to see and execute Java-based programs. The JDK enables the developer to create, compile and run his/hers own Java programs.

This is a small routine that connects the IE tool and the Ontology edition tool (the online-database). As explained in the design part, the tasks of this process is to connect to the IE tool, makes it learn new tagging rules from the training corpus. Then applies the rules to the real corpus. Afterwards it copes with all the extracted information. Connects to the database where the Ontology is stored, and then automatically introduces the instances in their correspondents places.

37 What I did not have the time to implement

This is a very wide project, which should be carried out with the time and the proper technical resources. Although I was very confident at the beginning of this Master Thesis, I realized afterwards that this is a very ambitious project to be developed form scratch by an only person within seven months. Not all these stages could be handled, mainly because of the required theoretical content (many time was spent in analyzing and making the formal models to represent the domain)

Now that all the analysis and design is made, and many technical problems have been solved and documented, it would be easier for another Master Thesis to complete the missing stages and fulfill all the process.

The missing part is the Web pages retrieval, which can be considered as another complementary project.

Although the query system has been implemented, it is not completed and some more queries should be added to finish this part. A prototype of an HTML page where the user can make queries to the system has been designed and it is shown in Appendix-14.

Also the knowledge base consolidation was impossible to implement due to time limitations. But these tasks have been carefully studied and complete guidelines of how to perform this task have been explained in this report.

I have also experienced a lot of problems with the Annotation and IE tools. It was a hard task to run them in my computer. Some of the tests last days and others never ended. The next table shows the technique features of the computers, this project was run on:

	Computer 1	Computer2
CPU	AMD Athlon™ Processor	AMD Athlon™ Processor
RAM Memory	640 MB	384 MB
Frequency	1.66 GHz	807 MHz
Operative System	Microsoft Windows XP Professional. Version 2002	Microsoft Windows XP Professional. Version 2002.

The main problem was the lack of CPU. This was the only reason that stopped the project to get good results with the annotation and IE tools.
Each time the annotation tool or the IE tool were run, the CPU usage rose to the hundred per cent. All the CPU power was consumed by these processes and it was impossible to perform any other action. With small inputs the tools (luckily) finish, but with medium-size ones they never finished

XII Test

Three corpuses with different characteristics have been tested in order to see how well the IE performed.
A detailed explanation about how this corpus is, and the results obtained with them and different settings can be found in Appendix-26.

XIII Conclusion: What would be done differently if I could do it all over again

First of all, I would like to remark that my knowledge about Ontologies and Semantic Web can not be compared now with the knowledge I had about this subject when I began this project. Neither my technique skills, which have improved all over this months.
If I had to start the project all over tomorrow I would really do things different, but this is because I have now much more knowledge about the subject than I had before.

Anyway, I will state what could have been done different, and which improvements can now be made. It can be maybe useful for future projects about this subject.

Firstly, I would have chosen an easier **context** than the recipes one. Although it looked very interesting and not so difficult at the beginning, then it tuned up to be an incredible wide and complex, with a lot of different possible points of view. Other difficulty of the recipes context is that it does not exist any official site or any rules or criteria to design recipes web page. Due to this it is so spread out and ambiguous. In spite of this, this is the real challenge of the semantic web; to deal with non-official pages.

Secondly, whatever context is selected, I would not **spend so much time in** the **analysis** part. Although it is very important and is the base for a good design an implementation, a lot of time was spent in this phase, taking up a lot of time for the design and implementation phase.

Thirdly, now that I now that the state-of-the-art of most of the Ontology-based tools is no so advanced yet (and the problems and lacks that it carries) I would chose the **program-based approach** if I had to begin again (a deeply study of this approach has been already made. See Appendix-11. Although it looked more complex, this approach would have been easier to implement, because then I would have relied on myself, not on other people tools; avoiding delays, problems about the installation, bad specifications, incomplete versions, etc.

Finally, supposing the tool-based approach is chosen again, and more time is given, I would have export the knowledge base into the new W3C Semantic Web standard language: **OWL**, which allows to define more properties than XML. This could not be done because time limitations. I had to make it in XML because it was the language I already new.

XIV Possible extensions

1) Make some suggestions of menus, grouping together some dishes, following some criteria, for example:

   - As the starter put some light dish, maybe salad, soup, or a cold dish or vegetal food.
   - As the main plate serve meat, fish…or a heavy vegetarian food (for vetgetarian people)
   - The dessert can be one of some fixed dishes (cakes, biscuits, ice-cream, pudding, creams, fruit, nuts...etc) mainly sweet things, fruit or cheese.
   - Look after the nutritional pyramid while making the menu suggestion. Take care of having all the nutritional groups in the correct dairy-recommended proportions.
   - Maybe combine hot and cold dishes in the same menu, or base on the season to suggest cold or hot dishes.
   - Make a menu with all the four flavors in it
   - Make some suggestions of typical Italian menus, Spanish, Arabic, Chinese…etc

   (One possible approach to make this suggestion is shown in Applendix-19)

   - Make some special menus for vegetarian people, fat people, and other people with special necessities.
   - Suggestions about if a dish is for lunch or for dinner, are very complex to make because that depends on the person and on the country. (For example, in Denmark people have a light lunch and a heaviest dinner, quite the opposite than in Spain for example)

2) Say if a recipe (or an ingredient) is willing to be frozen or if it looses all the taste and the nutritional facts (some additional information is needed for this feature)

3) Make some Christmas recipes, Easter recipes, summer recipes etc. (information about the season of the ingredients should be added to the model)

4) Add the medical properties of each ingredient, what kind of diseases they are good for. (I have already implemented the vitamins each ingredient has)

5) With more complex dietetic and medical knowledge nice features can be added. Suggestions about a dish, basin on the sex, age, height and weight (*the percentage of fats has to be less than 30% each day in order to not get weight*)can be made, but much more information of other fields is needed (one approach could be to merge with different medical and nutritional Ontologies)

6) Etc, etc…

1. What did I gain doing this project?

Despite all the problems experienced, I have found a lot of positive points when developing this project.

With this project I have learned how the state of the art in this AI field is, and how Ontologies, regardless of being a very theoretical structure, can be applied and implemented in real projects.

During these months I have been in touch with many Ontology and IE experts. It has been really beneficial for me to see how they work in the real world.

I have also helped them reporting the problems and bugs their tools had. Maybe I gave some little idea to move forward in this field

XV References

Below are listed all the books, web sites and articles that I have consulted during the development of my Master Thesis. They are listed in alphabetical order for an easier use.

1. ***AGENTC: A compiled Agent Programming Language.*** Henrik Lauritzen , IMM-THESIS-2002-61
2. ***Agents and the Semantic Web.*** James Hendler, Department of Computer Science, University of Maryland, College Park, MD 20742. (Published in the IEEE Intelligent Systems Journal, March/April 2001)

3. ***An XML-enabled data extraction toolkit for web sources.*** Ling Liu, College of Computing, Georgia Institute of Technology, Atlanta; Calton Pu, Wei Han. Information Systems 26 (2001) 563-583

4. ***Automatic information extraction from semi-structured web pages by pattern discovery;*** Chia-Hui Chang, department of Computer Science and Information Engineering, National Central University, Chungli, Tauyuan 320, Taiwan; Chun-Nan Hsu, Institute of information science, Academia Sinica, Nankang, Taipei 115, Taiwan ;Shao-Cheng Lui, ChungHwa Telecommunication Laboratories, Yangmei, Tauyuan 326, Taiwan. Decision Support Systems 35 (2003) 129-147

5. ***A proposal of infrastructural needs on the framework of the semantic web for ontology construction and use;*** Asunción Gómez-Pérez. Facultad de informática, Universidad Politécnica de Madrid. Campus de Montegancedo s/n. Boadilla del Monte, 28660. Madrid. Spain

6. ***Building Ontologies at the Knowledge Level using the Ontology Design Environment.*** M.Blázquez, M. Fernández, J.M. García-Pinar, A.Gómez-Pérez. Laboratorio de Inteligencia Artificial, Facultad de informática, Universidad Politécnica de Madrid. Spain.

7. ***Conceptual-model-based data extraction from multiple-record Web pages***. D.W.Embley, D.M.Campbell, Y.S.Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K.Ng, R.D. Smith. Data Extraction Group, Brigham Young University, Provo, Utah, UT 84602, USA

8. ***El futuro de internet tiene un nombre: La web semántica (The future of internet has a name: The semantic web)*** By Javier Castañeda, Posted on http://www.baquia.com/com/20011115/art00016.html

9. ***Entity-Relationship Models as Grammars and Lattices – a Foundational View***. Hans Bruun & Jørgen Fischer Nilsson. Informatics and Mathematical Modelling, Technical University of Denmark.

10. ***How to structure and access XML documents with Ontologies***. Michael Erdmann and Rudi Studer. Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) Universät karlsruhe (TH) , karlsruhe (Germany)

11. ***Introduction to Orders and Lattices***. Jørgen Fischer Nilsson & Nikolaj Oldager. Course of Knowledge-based Systems, Informatics and Mathematical Modelling, Technical University of Denmark.

12. ***La web inteligente (the intelligent web)*** ; published on http://www.baquia.com/com/20011115/art00016.html

13. ***(LP)^2, an Adaptative Algorithm for Information Extraction from Web-related Texts***, Favio Ciravegna. Department of Computer Science, University of Sheffield. Regent Court, 211 Portobello Street, S1 4DP Sheffield, UK.

14. ***"Reconstructing DTD best practice"***,Henry Thompson. Human Communication Research Centre ,2 Buccleuch Place, Edinburgh EH8 9LW, Scotland. Published on http://xml.com/pub/2000/06/xmleurope/schemas.html

15. ***Semi-structured Data Extraction from Heterogeneous Sources****, Xiaoying Gao and Leon Sterling. Intelligent Agent Laboratory. Department of Computer Science and Software Engineering. The University of Melbourne, Parkville 3052, Australia*

16. ***Some Ideas and Examples to Evaluate Ontologies*** Asunción Gomez-Pérez; Knowledge Systems Laboratory; Stanford University

17. ***Summary of: How to structure and access XML documents with ontologies;*** Michael Erdmann and Rudi Studer. Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) Universät karlsruhe (TH) , karlsruhe (Germany)

18. ***The relationship between recall and precision***. Michael Buckland and Fredric Gey. School of Library and Information Studies, University of California, Berkeley, Berkeley, CA 94720

19. ***The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities***. By Tim Berners-Lee, James Hendler and Ora Lassila

20. ***Methontology***. Mariano Fernández-López, Asunción Gómez-Pérez, Artificial Intelligence Laboratory, Technical University of Madrid (UPM), Spain

21. ***Next generation information technologies*** ; published on http://jornada.enredando.com/

22. ***Ontology building: A survey of editing tools****;* published on XML.com http://www.xml.com/pub/a/2002/11/06/ontologies.html

23. ***Ontology editor survey results***; published on http://xml.com/2002/11/06/Ontology_Editor_Survey.html

24. ***Puede hacerse Internet más fácil? (Can internet become easier?)*** Published on http://www.ukinspain.com/NewsSC/UKSC_NEWS_detail.asp?IdNews=2505

25. ***Semi-structured Data Extraction from Heterogeneous Sources*****,** Xiaoying Gao and Leon Sterling. Intelligent Agent Laboratory. Department of Computer Science and Software Engineering. The University of Melbourne, Parkville 3052, Australia

26. ***Six different kinds of composition***; James J.Odell

27. ***Some Ideas and Examples to Evaluate Ontologies*** Asunción Gomez-Pérez; Knowledge Systems Laboratory; Stanford University **??**

28. ***Spinning the Web: How to provide information on the internet.*** 1996,Tim Dixon, (ISBN 1-85032-290-2).

29. ***The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities,*** By Tim Berners-Lee, James Hendler and Ora Lassila. Published on Science and Technology at Scientific American.com in May 2001. http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

30. ***Weaving the web - the original design and ultimate destiny of the world wide web by its inventor***. Tim Berners-Lee and Mark Fischetti. IEEE transactions on professional communication, Vol 43, No 2, June 200.

31. ***WebODE: an integrated workbench for ontology representation, reasoning and exchange***. Óscar Corcho, Mariano Fernández-lópez. Laboratory of Artificial Intelligence, School of Computer Science, Technical University of Madrid.

32. ***WebODE: a Scalable Workbench for Ontological Engineering;*** Julio C. Apírez, Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez. Laboratory of Artificial Intelligence, School of Computer Science, Technical University of Madrid.

33. ***WebODE 1.0 User's Manual.*** Julio César Arpírez Vega, Laboratory of Artificial Intelligence, School of Computer Science, Technical University of Madrid.

34. ***WebOQL, Restructuring Documents, Databases and Webs.*** Gustavo O. Arocena, Alberto O. Mendelzon, Department of Computer Science, University of Toronto

35. ***What is an Ontology?*** Tom Gruber. Presented at the Padua workshop on Formal Ontology, March 1993 and posted in http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

36. ***Why Use DAML?*** Published on The Darpa Agent Markup Language Homepage http://www.daml.org/index.html

37. ***WonderTools?A comparative study of ontological engineering tools***. A.J.Duineveld, R. Stoter, M.R.Weiden, B. Kenepa and V.R. Benjamis.Department of Social Science Informatics (SWI), University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands.

38. ***Wrapper generation for semi-structured internet sources;*** N.Ashish, C.Knoblock. Information Science Institute and Department of Computer Science. University of Southern California.

39. ***W3C-XML representation of a relational database****;* published on http://www.w3.org/XML/RDB.html

40. ***XML representation of a relational database;*** published on http://www.w3.org/XML/RDB.html

41. ***XML Tutorial;*** published on http://www.w3schools.com/xml/default.asp

38 Recipe's web sites consulted

1. http://allrecipes.com/
2. http://1stholistic.com/Nutrition/nutrition.htm
3. http://www.betapure.com/index.htm
4. http://www-2.cs.cmu.edu/~mjw/recipes/
5. http://frenchfood.about.com/cs/cookingutensils/
6. http://www.foodsubs.com/
7. http://www.txbeef.org/dictionary.php3
8. http://eos.cnice.mecd.es/mem2001/nutricion/program/apli/alitip.html
9. http://www.nutrition.org
10. http://www.uia.mx/ibero/oacademica/licencia/salud/nutricio/sisequiv/
11. http://www.ar-revista.wanadoo.es/
12. http://www.deliaonline.com
13. http://health.yahoo.com/health/nutrition_fitness/recipe_search/
14. http://www.dianaskitchen.com/index.htm

15. http://www.kangaroobrands.com/textonly_recipes.htm

39 Development groups and interesting projects all around the world

1. Advanced Knowledge Technologies (AKT) http://www.aktors.org/akt/
2. Defense Advanced Research Projects Agency http://www.darpa.mil/
3. OntoWeb http://www.ontoweb.org
4. Semantic web agents project (Maryland Information and Network Dynamics Lab) http://www.mindswap.org
5. Semantic Web & Ontology Related Initiatives http://ontobroker.semanticweb.org/
6. Sheffield natural language processing group http://nlp.shef.ac.uk/
7. SWSI (Semantic Web Services Initiative) http://www.swsi.org/
8. SWWS (Semantic Web Enabled Web Services) http://swws.semanticweb.org/
9. The semantic web community portal http://www.semanticweb.org/
10. The Semantic Web group of the W3C http://www.w3.org/2001/sw/
11. University of Madrid, Artificial Intelligence Department of the Computer Science of the UPM http://www.dia.fi.upm.es/
12. University Of Maryland http://www.umd.edu/
13. Web-Ontology (WebOnt) Working Group, part of W3C (World Wide Web Consortium) http://www.w3.org/2001/sw/WebOnt/

40 Languages related to the Semantic Web

1. DAML (The DARPA Agent Markup Language) http://www.daml.org/
2. DAML+OIL http://www.w3.org/TR/daml+oil-reference
3. OIL (Ontology Inference Layer) http://www.ontoknowledge.org/oil/
4. OWL (Web Ontology Language) http://www.w3.org/TR/2004/REC-owl-features-20040210/
5. SHOE (Simple HTML Ontology Extensions) http://www.cs.umd.edu/projects/plus/SHOE/
6. RDF (Resource Description Framework) http://www.w3.org/RDF/
7. RDFS (RDF Vocabulary Description Language 1.0: RDF Schema) http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
8. XML (Extensible Markup Language) http://www.w3.org/XML/

41 Consulted dictionaries

1. Cambridge Advanced Learner's Dictionary
2. Collins Pocket. Spanish-English English-Spanish dictionary
3. Foreingword.com http://www.free-translator.com
4. Longman Dictionary of Contemporary English
5. Thesaurus.com http://thesaurus.reference.com/

I. Glossary

Bellow is shown an alphabetical-ordered list with a brief explanation of technical terms that might be difficult to understand.

- **Data mining:** It is a technique that allows people to search for unexpected relations in large data collections and can be applied to structured collections, such as databases, as well as unstructured collections such as documents written in natural language.
- **DAML: DARPA Agent Markup Language.**
- **HTML** (Hypertext Markup Language): **Language used to create documents on the** World Wide Web. HTML defines the structure and layout of a Web document by using a variety of tags and attributes. Its tags allow the creator to structure the text into headings, paragraphs, lists, hypertexts links, etc. These tags are standard and understandable from any kind of browser. [http://www.webopedia.com/TERM/H/HTML.html]
- **HTTP (Hypertext Transfer Protocol).** It is a protocol to access to the web pages located in the servers connected to the WWW. It states how the browsers and the servers have to communicate between them, and how they should react against different commands (to post, maintain, retrieve information…). The HTTP states also the way these messages are formatted and transmitted. It is a stateless protocol, it means without any memory of what the previous commands. To fulfill this lack other methods like cookies, JavaScripts and so forth have appeared[http://www.webopedia.com/TERM/H/HTTP.html]
- **Hyponym:** A subordinate word that is more specific than a given word. Technical term for included meaning. A is a hyponym of the main B if all A's are B's, but not all B's are A's
- **Information extraction:** Is the process of pulling certain kinds of information out of documents automatically. This information can be treated afterwards in many ways (populate a Database for example)
- **Intranet:** An intranet is a private network that is contained within an enterprise. It may consist of many interlinked local area networks and also use leased lines in the wide area network [http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci212377,00.html]
- **LAN (Local Area Network):** A computer network that spans a relatively small area. [http://www.webopedia.com/TERM/l/local_area_network_LAN.html]
- **Markup:** sequence of characters or other symbols inserted at certain places in a text file to indicate how the file should look when it is printed or displayed or to describe the document's logical structure. The markup indicators are often called "tags." [http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci212527,00.html]
- **Natural language processing** (NLP) is a branch of computer science devoted to manipulating documents written by people with computers. [http://www.alias-i.com/lingpipe/]
- **Ontology**
- **Over-fitting** (data). This term is used in machine learning techniques to refer to the inconvenience of just fit the data of the training set and so loose generalization. It is a very common lack of the machine learning techniques.
- **OWL: W**eb **O**ntology **L**anguage**.** "*OWL is used to publish and share sets of terms called Ontologies, supporting advanced Web search, software agents and knowledge*

management" [W3C definition] .It became a W3C Recommendation the 10th of February of 2004.

- **Part of speech tagging** (POS) Also called grammatical tagging, is the commonest form of corpus annotation, and was the first form of annotation to be developed. [http://www.comp.lancs.ac.uk/ucrel/]
- **RDF: R**esource **D**escription **F**ramework. "*RDF is used to represent information and to exchange knowledge in the Web" [W3C definition]* .It became a W3C Recommendation the 10th of February of 2004.
- **Sentence detection or identification** is an early processing step common to many natural language processing systems in which paragraphs of text are divided into sentences based on punctuation and other information contained in each paragraph. [http://www.comp.lancs.ac.uk/ucrel/]
- **SGML** (Standard Generalized Markup Language): Standard for how to specify a document markup language or tag set. SGML is metadata. It is not itself a document language, but a description of how to specify one.
- **Summarization:** is the natural language processing task of producing an abstract from a document by machine.
- **Taxonomy:** A system for naming and organizing things, especially plants and animals, into groups which share similar qualities (Cambridge dictionary definition)
- **Thesaurus:** A type of dictionary in which words with similar meanings are grouped together (Cambridge dictionary definition)
- **Tokenization:** Is an early processing step common to many natural language processing systems in which text is divided into tokens—most of which are words but some of which are punctuation, numbers or other symbols—in order to make additional processing easier.
- **URI (Uniform Resource Identifier)**: The generic term for all types of names and addresses that refers to objects on the World Wide Web. A **URL** is one kind of URI. [http://www.webopedia.com/TERM/U/URI.html]
- **URL (Uniform Resource Locator):** Is a schema that provides every page a unique address. It has two parts: the protocol to be used (http, ftp…) and the IP address of the desired resource. It is the only web addressing technology nowadays
- **XML**: **E**xtensible **M**arkup **L**anguage. It is an open language to defining web data along with its structure and able to provide also semantic.
- **XPath**: Xpath is composed by syntax rules that allow us to address of an XML document, by using paths. With XPath we can manipulate string, numbers and Booleans and provides a library of standard functions. As well as XQuery, XPath is not written in XML On the contrary of XQuery, Xpath is a W3C recommendation. It is an important element in XSLT.
- **Xpointer:** XPointer is a language for locating data within an Extensible Markup Language (**XML**) document based on properties such as location within the document, character content, and attribute values. [http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci345240,00.html]
- **XQuery**: XML Query (XQuery) is a query language for querying XML data developed by W3C. It is not yet an standard, it is still working draft.

It is based on XPath and XML Schema datatypes. It is a newest version based on XML-QL, YATL,Lorel and Quilt. XQuery is not an XML language. (an XML version of XQuery is XQueryX)

- **XSLT:** XSLT began as XSL (eXtensible Stylesheet Language).

XSL was developed by W3C because there was a need to display the XML data, and the browsers needed more information to display it (HTML uses predefined tags that are understood by a browser, but not the XML tags, which meaning is not understood by a browser) is composed by three other languages:

XPath: Language that defines parts of XML documents (Described before)

XSL-FO: Language that formats XML documents

XSLT: A language that transforms XML documents in other XML, XHTML (the most common) or HTML documents.

It can also perform more actions like add or remove elements, sort them or decide whether to show them or not.

XLST uses XPath to define a part of a document, looks for in the XML source document and then transforms it into the result document.

Most of XML applications use Document Type Definition (DTD) to define their structure.

XIV. Appendices

APPENDIX_1 Some examples of online recipes and their most interesting parts

These are some examples of the surveyed recipes. I have searched in many different web sites, in order to obtain a global idea of how can a recipe look like.

These are some of the most interesting examples, because their differences in the format as well as in the contents. I have highlighted the main parts of a recipe: The ingredient description (highlighted in yellow), the cooking instructions (highlighted in green), the nutrition information (highlighted in blue), the time it takes to prepare this dish (highlighted in gray) and the amount of resulting food (highlighted in red). Notice that not all the recipes contain the same parts, nor in the same order.

Bolognese Sauce
Submitted by: Kimber

"An excellent chunky pasta sauce with beef, pork, lots of vegetables and tons of flavor. Freeze any unused portions for later use. If you have fresh herbs, you may substitute 2 teaspoons chopped fresh basil for the dried basil in this recipe."
Yields 8 to 10 servings.

Prep Time: 10 Minutes
Cook Time: 1 Hour 25 Minutes
Ready In: 1 Hour 35 Minutes
Avg. Member Rating:
*****
VIEW
- 22 Ratings
- 11 Reviews

INGREDIENTS:

2 tablespoons olive oil
4 slices bacon, cut into 1/2 inch pieces
1 large onion, minced
1 clove garlic, minced
1 pound lean ground beef
1/2 pound ground pork
1/2 pound fresh mushrooms, sliced
2 carrots, shredded
1 stalk celery, chopped
1 (28 ounce) can Italian plum tomatoes
6 ounces tomato sauce
1/2 cup dry white wine
1/2 cup chicken stock
1/2 teaspoon dried basil

1/2 teaspoon dried oregano

salt and pepper to taste

1 pound pasta

DIRECTIONS:

1. In a large skillet, warm oil over medium heat and saute bacon, onion and garlic until bacon is browned and crisp; set aside.
2. In large saucepan, brown beef and pork. Drain off excess fat. Stir in bacon mixture, mushrooms, carrots, celery, tomatoes, tomato sauce, wine, stock, basil, oregano, salt and pepper to saucepan. Cover, reduce heat and simmer one hour, stirring occasionally.
3. Bring a large pot of lightly salted water to a boil. Add pasta and cook for 8 to 10 minutes or until al dente; drain.
4. Serve sauce over hot pasta.

Makes 9 servings

Nutrition Info
Servings Per Recipe: 9

Amount Per Serving

Calories: 478

Total Fat: 20.5g

Cholesterol: 58mg

Sodium: 418mg

Total Carbohydrates: 46.7g

Dietary Fiber: 3.4g

Protein: 23.1g

Powered by ESHA Nutrient Database
About our nutritional information

This is one of the most complete online recipes I have found all over the Web (restricting the searching to the English language).
It has all the recipes features very detailed and clearly defined. It is also one of the most overloaded one, with pictures and banners all over the page. This has been retrieved from the web site: http://allrecipes.com/

BAKED CORN AND TOMATOES

Mix:

2 c. corn
2 c. tomatoes (I used canned)
1 tsp. salt and a little pepper
1 tbsp. sugar (or more to taste)

Put into a greased baking dish. Spread 1 cup fresh bread crumbs over the top. Dot with 3 tablespoons butter. Bake in moderate oven, 350-400 degrees for 1/2 an hour.

Caesar Salad

From: leclair@skatter.usask.ca (Don Leclair)

Date: Thu, 22 Jul 1993 19:11:23 +0000 (GMT)

```
Here is my recipe for Caesar Salad.  Hope you enjoy it. Caesar Salad


1 head Romaine lettuce

4 cloves garlic, minced (or diced or mashed or whatever)

3 tbsp vegie oil (I use Canola)

1 tbsp olive oil

1 tsp lemon juice

1 egg

2 or 3 shakes of Tobasco sauce

2 or 3 shakes of Watsdishere sauce (worchestershire)

2 or 3 grains of salt (or more if you're so inclined)

1/4 tsp dry mustard

1/4 tsp ground black pepper


Mix all ingredients except lettuce in a shaker and shake until well
mixed.

Break lettuce into bite sized pieces and put it into a bowl.

I find that a wooden bowl works well.  Pour the sauce over

the lettuce and toss to coat.  Enjoy
```

These are two examples of the least detailed recipes I have found on the Web. They only have two well defined parts: the ingredient description and the way of doing part. The first one has the time feature but is embedded in the description of the recipe, so all the text has to be carefully processed to find this feature.
The first one was found on the recipes web site: http://www.cooks.com/

The second one was found in an educational program of the school of computer science:
http://www-2.cs.cmu.edu/~mjw/recipes/

Quattro Formaggi (Four Cheese) Pizza

This is the classic version of one of the most wonderful combinations of bread and cheese imaginable. You can, of course, vary the cheeses, but the ones I've chosen here are a truly magical combination. Vegetarians might like to know that a vegetarian parmesan-style cheese is available from Twineham Grange Farms, tel: 01444 881394; enquiries@twinehamgrangefarms.co.uk

Sufficient for a 10 inch (25.5 cm) base pizza – serves 2

For the pizza base:

6 oz (175 g) plain white soft flour

1 level teaspoon salt

1 level teaspoon easy-blend dried yeast

1/2 level teaspoon golden caster sugar

1 tablespoon olive oil

2-3 level tablespoons polenta (cornmeal) to roll out, plus a little extra

For the topping:

21/2 oz (60 g) ricotta

2 oz (50 g) Mozzarella, cut into 1 inch (2.5 cm) slices

2 oz (50 g) Gorgonzola Piccante, cut into 1 inch (2.5 cm) slices

1 oz (25 g) Parmesan (Parmigiano Reggiano), grated (see recipe introduction)

* Click an ingredient to find out more

You will also need a pizza stone or solid baking sheet measuring 14 x 11 inches (35 x 28 cm).

Begin by warming the flour slightly in the oven for about 10 minutes, then turn the oven off. Sift the flour, salt, yeast and sugar into a bowl and make a well in the centre of the mixture, then add the olive oil and pour in 4 fl oz (120 ml) hand-hot water. Now mix to a dough, starting off with a wooden spoon and using your hands in the final stages of mixing. Wipe the bowl clean with the dough, adding a spot more water if there are any dry bits left, and transfer it to a flat work surface (there shouldn't be any need to flour this). Knead the dough for 3 minutes or until it develops a sheen and blisters under the surface (it should also be springy and elastic). You can now either leave the dough on the surface covered by the upturned bowl or transfer the dough to a clean bowl and cover it with clingfilm that has been lightly oiled on the side that is facing the dough. Leave it until it looks as though it has doubled in bulk, which will be about an hour at room temperature.

Having made the dough and left it to rise, pre-heat the oven to gas mark 8, 450°F (230°C), along with the pizza stone or baking sheet. The next stage is to tip the dough back on to a work surface that has been sprinkled generously with polenta to prevent it from sticking. Knock all the air out of the dough and knead it for a couple of seconds to begin shaping it into a ball. Then dust your rolling pin with polenta and roll the dough out to a circle that is approximately 10 inches (25.5 cm) in diameter. Then finish stretching it out with your hands working from the centre and using the flat of your fingers to push the dough out; it doesn't need to be a perfect round, but you want it to be a fairly thin-based pizza, with slightly raised edges.

Pre-heat the oven to its lowest setting.

Then, using a thick oven glove, very carefully lift the baking sheet or pizza stone out of the oven and sprinkle it with a little polenta (cornmeal). Now carefully lift the pizza dough on to the stone or baking sheet and quickly arrange teaspoonfuls of ricotta here and there all over. After that, scatter the Mozzarella and Gorgonzola pieces in between and, finally, scatter the Parmesan over.

Bake the whole thing on a high shelf for 10-12 minutes, until the crust is golden brown and the cheese is bubbling. You can lift the edge up slightly to check that the underneath is crisp and brown. Carefully remove the baking sheet or pizza stone from the oven, again using a thick oven glove, and serve the pizza on hot plates straight away.

This recipe is taken from How to Cook Book One and Delia's Vegetarian Collection.

This recipe is very complete and has almost all the relevant features the program will look for, and also some additional information (about the utensils' size) which can be added as an extension of the current features. In this web page the description of the ingredients is spitted in two, depending of the part of the recipe. The time is also difficult to find because it is embedded into the description. It does not relate to a general time, but to a partial times

Easy Pasta with Tomato Sauce

From FoodFit

FoodFit Original Recipes use seasonal ingredients to create dishes that are healthy, delicious and easy to make. Our chef follows FoodFit's nutrition standards for recipes that are lower in fat, but full of flavor!

Ingredients

2	teaspoons	olive oil
1	-	small onion, minced
2	cloves	garlic, minced
2	cups	chopped, diced tomatoes
2	tablespoons	freshly chopped basil
1/4	cup	chicken stock
-	-	salt to taste
-	-	freshly ground black pepper
1	pound	angel hair pasta, (cappellini)

Preparation - Estimated cooking time: 25 minutes -

1. In a large sauté pan over medium-high heat, add the olive oil and onions.
2. Cook the onions for about 5 minutes until they turn golden, then add the garlic and cook until the garlic begins to soften. Add the tomatoes, basil and stock and simmer for about 5 minutes for the flavors to blend.
3. Meanwhile, bring a large pot of salted water to a boil. Add the pasta and cook until al dente, about 2 minutes. Drain.
4. Transfer the pasta to a warm bowl and serve with the sauce.

Nutritional Analysis

Number of Servings: 4

Per Serving

Calories	139	**Carbohydrate**	23 g
Fat	4 g	**Fiber**	5 g
Protein	5 g	**Saturated Fat**	1 g
Sodium	265 mg		

This last recipe is very complete as well. It details the nutritional values, the number of servings and the time separately. This would be one of the most easily understandable recipe (as well for a human as for a machine), because of the clarity of the description.

What can be concluded from these examples is that a recipe has no fixed structure. It has not a predefine structure, nor in the way of formatting the information, nor on the way of expressing the information. Some parts can be missing, as some recipes are more detailed than others.

What is clear looking trough all of them, is that there are two main parts that are always present: the ingredient description (highlighted in yellow) and the cooking instructions (highlighted in green). These are mandatory fields within a recipe. If any is missing it can not be considered a recipe (as it would not be either understandable for a human).

Regarding the way of describe the sections, they do in many different ways. Some of them introduce the different parts of the recipe. For example: *ingredients, for the pizza base: For the topping, mix,* or just nothing are the words to introduce the ingredients description part. *Preparation, begin with, mix, put, directions …* are the way of introducing the way of doing in these recipes. *Nutritional analysis* and *nutrition info* is a way of introducing the nutritional values description. The system has to cope with all these differences owing to the freedom and the lacks of standards of the web.

APPENDIX_2 W3C Annotation project: Annotea

The W3C open annotation project is called Annotea. These annotations have two parts:

- The annotations are described as metadata by means of a RDF Schema [see Glossary]. This data provides information about the user who has made it, the date, and the type of the annotation and the URI of the annotated web.

- The position of the annotation in the web document it is posted on is showed using the **Xpointer** protocol [see Glossary]. The browser puts a visual icon in the place indicated with the Xpointer.

The annotations are structured documents such as HTML, XML or another kind of annotation.
Each annotation has its own URI (Uniform Resource Identifier) [see Glossary] since each annotation is a resource itself, and they are stored in another special annotation server, which stores RDF [see Glossary] generic databases.

When a user visits a web page with annotations, he downloads the original web page (without changes) and automatically he also downloads the positions of the annotations; so he can see how many annotations the web page has. (For this purpose the user has to have installed an annotation client in his computer; the clients currently available for Annotea are: *Amaya, Annotea bookmarklet, Annozilla, Snufkin* and *Annogates)*

It is then when the user can decide whether or not to download the annotations.
If he decides to do it, an annotation request (via HTTP GET method) is made to the server and that single annotation is downloaded and shown to the user in a window in the browser.

APPENDIX_3 WebODE's Methodology: Methontology

Introduction

Up till now, there is a big lack in the field of Ontology building methods. There is no standard method to guide the construction and design of Ontologies, which makes very difficult to design and implement an Ontology, and much more to share it and reuse it by other systems. It is necessary to create a standard methodology (set of methods, techniques and standards) that guides the developers while making an Ontology.

The aim of a methodology for the Ontological engineering should be:

- Identify the stages of the Ontology's creation and development
- Standardize the Ontology life cycle
- Standardize the techniques to acquire the knowledge of a domain and model it with an Ontology

Steps of the Ontology development

3.1.1 Identify relevant information

The first step when developing an Ontology is to acquire all the relevant knowledge of the domain that is desired to be represented

In the recipes context a big amount of recipes have been carefully studied. The relevant information is modeled as an Ontology.

3.1.2 Conceptualize the information

METHODOLOGY conceptualizes this knowledge in a set of intermediate representations (IR), which afterwards will generate the Ontology model. (Instead of conceptualize into formal languages like other methodologies)

3.1.2.1 Create the taxonomy

First of all the methodology states to build a Glossary *of Terms,* which is a list of the terms (concepts, instances, attributes, verbs, etc.) of the domain and their description

Afterwards the terms shown above have to be related to create the Ontology taxonomy. This can be done with these hierarchical relationships:

Subclass-of *(Class C is a subclass of parent class P if and only if every instance of C is also an instance of P.)*

Mutually-disjoint-subclass-of: *(Is a set of subclasses of a class C whose objects have no elements which belong to different sets)*

Exhaustive-subclass-of *(A subrelation-partition of a class C is a set of mutually-disjoint classes (a subclass partition) which covers C. Every instance of C is an instance of exactly one of the subclasses in the partition)*

These three relationships corresponds to the general concept of inheritance (IS-A), with some additional characteristics. They are the main backbone of the Ontology.

3.1.2.2 Improve the Ontology structure

After providing the skeleton of the Ontology, more relationships can be added. These are transitive-part-of, intransitive-part-of and "ad-hoc" relationships defined by the user (they can be provided these characteristics: Reflexive Irreflexive Symmetrical Asymmetrical Antisymetrical Transitive)

3.1.2.3 Complete the Ontology skeleton

After having the concepts and their taxonomy, it is time to complete the knowledge model with the other elements: attributes synonyms, acronyms, description, axioms and rules.

Instance attributes are attributes that are defined in the concept but that take values in its instances.

Class attributes describe concepts, not concept instances.

Axioms are used to define the concepts by means of logical expressions that are always true.

Formulae can infer attribute values (only numerical) from another instance attributes or constants values. (These are made in Prolog (Horn clauses)).

The knowledge inferred by the formulae and axioms is not entered in the knowledge base, because if some of the knowledge used to infer these new concepts changes, then inconsistency problems will arise.

The last two features can be modeled on the Ontology editor, but they are not useful for the aim of this project; because the current semantic web languages do not still support these features.

3.1.2.4 Improving the Ontology: Merging the developed Ontology with new ones

The Ontology can be improved by means of merging it with others. This can be done with this editor. This is not an automatic project, as the user has to supervise and guide the merging with some additional knowledge

3.1.2.5 Evaluation of the Ontology

An evaluation method is incorporated to evaluate the Ontology and check its consistency.

APPENDIX_4 – Ontology editor survey results

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Apollo	Knowledge Media Institute of Open University (UK)	Classes with slots plus relations; functions; hierarchical views.	OKBC model	No, but {server is planned}.	CLOS; OCML	No, but planned.	Yes	No	No	No	No	None	http://apollo.open.ac.uk/index.html
CIRCA Taxonomy Administrator	Applied Semantics, Inc.	Maps designed taxonomies to built-in general lexical ontology using weighted concept clusters ("gist"). No definable relations.	Proprietary	No	(RDFS planned)	Browsing of ontology (not for editing).	Yes, limited.	No	Yes, via common mapping.	Yes	Via other CIRCA tools.	Part of CIRCA Auto-Categorizer. A future (4Q'02) product may support relations and RDF import/export.	http://www.appliedsemantics.com/as_solutions_autocat_taxadmin.shtml
CoGITaNT	LIRMM CNRS (France)	Conceptual graph (CG) modeling with rules; nested typed graphs; projections.	CG model	{Web based client access}	BCGCT; CGXML; XML	Browsing of ontology.	Yes	No	No	No	No	Open Source server, client and underlying C++ library; also Java API.	http://cogitant.sourceforge.net/
Coherence	Unicorn Solutions	Roundtrip transformation of ontologies from XML Schema and RDB schemas. Class and property hierarchies; business rules.	XML	{Internet client for sharing ontologies}	XML Schema; RDB schema; XML; RDF(S); DAML+OIL (in 2.1 release)	No, but planned.	Schema synchronization and dependency (referential integrity) to show impact of changes.	(Yes, in 2.1 release)	(Planned)	Explicit mapping between lexicons is possible.	No, except as explicit mappings from RDB.	Ontology functions are part of an enterprise data integration product. Additional input/output: entity-relation diagrams, COBOL Copybooks, HTML.	http://www.unicorn.com/pr-overview.htm
Contextia	Modulant	Basic concepts and relations with datatypes are represented in schemas.	Express	Referenced ontologies (URLs); URIs	Entity relation diagrams; XML Schema	For editing single ontology (using FirstStep XG).	Express model (ISO 10303) validation; cross-ontology consistencies	No	Schema mapping including aggregation/generalization; "context" mapping.	Synonym mappings; term matching	No, except as explicit mappings from structured and semi-structured sources.	Ontology functions are part of an enterprise data integration product. Ontology editing supported by FirstStep XG included with Contextia.	http://modulant.com/products/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
COPORUM OntoBuilder	CognIT AS	Basic concepts and relations are represented with single inheritance. Representation of concepts and relations extracted from content may be extended with WordNet information.	RDFS	{Web based repository; Web services in development}	DAML+OIL; RDF(S)	Browsing of ontology.	RDF consistency via repository.	(Under development)	Flat merging via Sesame.	Yes, based on WordNet and RDF Query Language; also in Sesame	Yes, based on meaning and distribution.	Tool embedded in On-To-Knowledge project tool set and requires Sesame RDF repository. Focus on generating editable ontologies automatically from natural language documents.	http://ontoserver.cognit.no/
DAG-Edit	Berkeley Drosophila Genome Project (BDGP)	Mixed part-of and isa concept hierarchies are represented along with synonym and search facilities. No properties.	Directed (cyclic or acyclic) graph notation	{Read input via URLs}	Gene Ontology: RDF; Gene Ontology Postgres Database Schema (experimental); (DAML+OIL in GOET)	No, but tree view of flattened graph.	No	No	Yes, especially at the term level; also change history tracking.	Yes, for synonyms.	No, but allows regular expression search.	While intended for gene expression ontologies, it can be used for any taxonomy. Generic version - GOET - is under development (alpha).	http://sourceforge.net/projects/geneontology
DAMLImp (API)	AT&T Government Solutions	DAML+OIL constructs. Basic Java library for analysis and manipulation of DAML+OIL ontologies.	DAML+OIL	URIs	DAML+OIL; RDF	No	No	Possible	No, but Ontology Manager aids mapping.	No	No	DARPA DAML project	http://codip.grci.com/Tools/Components.html
Differential Ontology Editor (DOE)	National Audiovisual Institute - INA (France)	Creates lattice of concepts and lattice of relationships between concepts, plus a set of instances. Concepts cannot be defined intentionally with constraints. Only types of the domains of relationships can be specified. No axiom editor is provided.	XML & CGXML	Load ontology by URL	DAML+OIL; RDFS	No, but tree view.	Arity and type inheritance on relation domains; detects cycles in hierarchies.	No	No	Term definitions, synonyms and preference; methodology for differential definitions.	No	Supports methodology of Bruno Bachimont; to be used with other editors.	http://opales.ina.fr/public/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Disciple Learning Agent Shell	George Mason University, Learning Agents Laboratory	Semantic network representation with functions, extended to allow partially learned entities. A hierarchy of objects and a hierarchy of features, with their descriptions, are represented as frames. Also, general problem solving rules can be expressed with terms from the ontology.	OKBC-like	{Ontology summaries output in HTML}	Import: CYC ontologies	Browse classes, properties and individuals.	Syntactic consistency is always maintained; can commit multiple changes to persistent ontology in single operation.	No	Yes, two ontologies.	Search for terms	No	The shell is used by subject matter experts to rapidly form knowledge and reason about a specific domain. Users, via a set of task reduction rules, create Disciple-RKF agents that can be combined into a single knowledge base.	http://lalab.gmu.edu/
Domain Ontology Management Environment (DOME)	Btexact Technologies	Concepts, relations and constraints are mapped to ER-like specifications.	CLASSIC & FaCT	{Web access}	OKBC; XML	ER diagrams	Yes	Yes	(Under development)	(Under development)	Semi-automatic and rule-based extraction from RDBs and web pages.	Available externally by individual agreements with limited support.	http://more.btexact.com/projects/ibsr/dome/index.htm
DUET	AT&T Government Solutions	Represents only UML static constructs available on class diagrams.	UML	URLs and namespaces are preserved in UML package naming	DAML	Editing using UML class diagrams (via Rose or Argo products)	Valid UML diagrams will produce valid DAML+OIL and conversely.	Supports multi-user capabilities of Rational Rose.	Multiple ontologies may be imported for comparison and merging.	No	No	DARPA DAML project. Additional output: HTML views of UML models. Also under development for GentileWare Poseidon UML.	http://codip.grci.com/Tools/Tools.html

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Enterprise Semantic Platform (ESP) including Knowledge Toolkit	Semagix, Inc	Description models composed of hierarchical categories and attributes with named relationships. Type system for heterogeneous media content. Instances supported by simple constraints on entities (cardinality, range) and entity properties, as well as inferencing. Automatic assertion and maintenance of instances is possible.	Graph & XML	URIs; {partial HTML client; HTTP API}	XML; (RDF(S) is planned)	Connected tree browsing via TouchGraph	Yes, includes automatic and user interactive checks; dynamic content management.	Limited, user privileges prevent concurrent update of the same ontology parts.	No	Synonym based term normalization.	Automatic ontology directed classification and semantic annotation of heterogeneous content.	ESP is an application platform for semantic integration of heterogeneous content including media and enterprise databases. It includes the Knowledge Toolkit for building ontologies.	http://www.semagix.com/
EOR	Dublin Core Metadata Initiative	RDF models as sets of triples. Can be used to build, insert (infuse) and query instance knowledge bases for DAML+OIL, RDFS, etc. ontologies.	RDF	URIs	RDF	No	Validate RDF	No	Yes, by adding sets of RDF statements.	No	No	Developed by OCLC.	http://eor.dublincore.org/index.html
ExClaim & CommonKADS Workbench	National Institute for Research and Development in Informatics (Romania)	Description logic modeling plus primitive problem solving actions.	DL model	No	CML	Browsing of ontology.	Knowledge verification and model validation (for DL representation).	User roles	No	No	No	Uses the CommonKADS Workbench based on SWI-Prolog and the XPCE GUI.	http://www.ici.ro/ici/expoeng/prodici/prod_12_22/pag_excl0.htm

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
GALEN Case Environment (GCE)	Kermanog	Description logic terminological modeling without support for individuals. Composite concepts are automatically classified according to their criteria (relationships with other concepts). New concepts can be created interactively and according to user-defined rules.	GRAIL	No	GRAIL	No, but filtered tree views allow editing.	Explicit grammatical and sensible sanctions are enforced when combining terms.	No	Compiles differences in concepts, hierarchies and criteria (properties) between two ontologies.	GALEN concept identifiers can be associated with synonyms and word forms.	No	Although, developed primarily as a medical terminology model builder, the tool can serve as a general purpose ontology editor. GCE is part of the Classification Workbench with support to manage domain classification schemes.	http://www.kermanog.com/
ICOM	Free University of Bozen-Bolzano, Italy	EER (extended entity relations) modeling plus inheritance hierarchies, multidimensional aggregations and multiple schema relations.	Description logic	No	XML; UML (future)	Native editing of ER diagrams (UML diagrams planned).	Verify the specification via DL classifier (FaCT).	No	Supports inter-ontology mappings with graphical interface.	No	No	Graphically editing of native UML class diagrams planned for next release.	http://www.cs.man.ac.uk/~franconi/icom/
Integrated Ontology Development Environment	Ontology Works, Inc.	Distinguishes between properties and relations; allows contexts; default reasoning; temporal model relations; higher-arity relations; meta-properties and meta-relations.	OWL (based on KIF; not related to W3C WebOnt language of same name.)	{Web client - control panel}	KIF; UML; RDB; XML DTD	UML diagrams	Top-level ontology consistency per Guarino & Welty.	Yes	(slated for version 1.8 in Q2 2003)	Synonyms; English-language names	No	Supports OntoClean methodology (Guarino & Welty); supports relational and other databases	http://www.ontologyworks.com/
IsaViz	W3 Consortium	Supports RDFS level specifications. Can specify any model based on RDF such as DAML+OIL.	RDF model	URI namespaces	RDF; N-Triple; SVG	Native creation and editing of resources, literals and properties.	RDF model correctness.	No	Yes	No	No	None	http://www.w3.org/2001/11/IsaViz/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
JOE	University of South Carolina Center for IT	Basic concept and relations modeling ala ER.	KIF	No	ER (LDL++)	No	No	No	No	No	No	No current development. Available as an applet.	http://www.cse.sc.edu/research/cit/demos/java/joe/
KAON (including OIModeller)	FZI Research Center & AIFB Institute, University of Karlsruhe	Extends RDFS with symmetric, transitive and inverse relations, relation cardinality, meta-modeling, etc. Similar to F-Logic using axiom patterns. Editor currently only supports concept hierarchy.	KAON (proprietary extension of RDFS)	{Browsing ontologies via KAON Portal; Web services API under development}	RDFS	No	Yes, for evolution of ontology.	Concurrent access control with transaction oriented locking and rollback.	(Under development)	Explicit lexical representation in model. Synonyms; stemming; multilingual.	(Under development)	OIModeller is part of KAON tool suite for business applications that uses RDB persistence layer for scalability. The ontology editor is under development.	http://kaon.semanticweb.org/
KBE -- Knowledge Base Editor (for Zeus AgentBuilding Toolkit)	Institute for Software Integrated Systems, Vanderbilt University	Zeus ontology model of concepts, attributes and values; multiple inheritance; modularization within a closed world model. (Also defines agent interaction protocols.)	GME	No	Zeus ontology file (.edf)	UML-like diagrams for browsing only.	Yes	No	No	No	No	KBE is layered on top of the Zeus environment for building agents and extends the ontology editor functions. The underlying GBE model specification system could be used as the basis of other ontology builders.	http://www.isis.vanderbilt.edu/Projects/micants/Tech/Demos/KBE/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
LegendBurster Ontology Editor	GeoReference Online Ltd	Semantic network hierarchy of concepts, attributes, attribute values and explicitly represented truth-status flags. Inheritance within hierarchies with lateral links. Full reified relations; inverse relations (partial). Metadata for all entities (at node level). Separate tree list editor.	Proprietary (uses Prolog)	No	No, except across projects (proprietary).	No, except SVG export of instance and query graphs.	Partial, with strict attribute context checks but arities currently unchecked.	No	Yes, if from LegendBurster. (User must check semantic consistency.)	Term search and alphabetical sort.	Semi-automatically capture and import vocabulary present in attribute tables of maps of interest.	While LedgendBurster is principally a GIS application, the Ontology Editor is suitable for general purpose ontology development. Standalone editor with instance description and fuzzy query expected in early 2003.	http://www.georeferenceonline.com/
LinKFactory Workbench	Language & Computing nv	Description logic T-box (terminological) and A-box (assertional) model. Multiple inheritance over concepts and relationships; identification of necessary and sufficient criteria for concept definition. Manage multiple conflicting ontologies in one T-box. Versioning metadata.	Extended description logic	{LinKFactory Server supports Internet clients; WebInfo spider component.}	XML; RDF(S); DAML+OIL/OWL	No	Checks cover role restrictions, formal disjoints, sanctioning over subsumers, etc.	Yes, with author privileges and auditing specific to concept hierarchies.	Compares and links ontologies via a core ontology; related concepts matched on formal relationships and lexical information.	Strict concept/term distinction; lexeme-description; part-of-speech. Search with wildcards.	Yes, via text analyses and automatic linkage to ontology. WebInfo spider gleans domain-specific concepts/terms on Web.	LinKFactory Workbench includes a database server, application server and clients. Originally designed for very large medical ontologies. It has a Java beans API and optional Application Generators for semantic indexing, automatic coding, and information extraction.	http://www.landc.be/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Medius Visual Ontology Modeler	Sandpiper Software, Inc	UML modeling of ontologies with frame systems support.	UML with extensions for OKBC model	URI support in DAML generator; {read-only browser support from Rose}	XML Schema; RDF; DAML+OIL	Yes, as UML diagrams via Rose.	Limited	Yes	Native Rose model merging support	Search for terms and relations.	No	Operates as a Rational Rose plug-in.	http://www.sandsoft.com/products.html
NeoClassic	Bell Labs (Lucent Technologies)	Framework representation of descriptions, concepts, roles, individuals and rules. Concepts can be derived from necessary and sufficient conditions for individual membership. Subsumption and classification are inherent inference. (Command line editor only.)	DL model	No	No	No	Yes	No	No	No	No	This C++ implementation of the original CLASSIC system is the only currently supported version.	http://www-out.bell-labs.com/project/classic/
OilEd	University of Manchester Information Management Group	DAML constraint axioms; same-class-as; limited XML Schema datatypes; creation metadata; allows arbitrary expressions as fillers and in constraint axioms; explicit use of quantifiers; one-of lists of individuals; no hierarchical property view.	DAML+OIL	RDF URIs; limited namespaces; very limited XML Schema	RDFS; SHIQ	Browsing Graphviz files of class subsumption only.	Subsumption and satisfiability (FaCT)	No	No	Limited synonyms	No	None	http://oiled.man.ac.uk/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
OLR3 Schema Editor	Institute for Information Systems, University of Hannover	Instantiation and editing of external or custom schemas conforming to RDFS. Concept-specific filtering to present choice of legal properties.	RDFS	RDF URIs; {browser based}	RDF	No	Yes, for property constraints, etc.	No	No	No	No	Part of the Open Learning Repository Version 3 (OLR3) system for course specification.	http://www.kbs.uni-hannover.de/~tkunze (German only)
OntoBuilder	Institute for Medical Information, Statistics and Epidemiology University of Leipzig	Manages compilation of domain terms, their description, and contexts using natural language.	Natural language; (logical representation language planned)	{Web access}	No	No	Not automatically	Yes, with editor, moderator and administrator user group types.	No	Representation of synonyms; search on terms and descriptions; lexical rules for term input	No	Semantic analysis using a formal model based on a top level ontology and a logic-based representation language are planned. Domain focus is on medicine.	http://www.kompetenznetz-lymphome.de/KlinischeStudien/Qualitaetssicherung/DataDictionary/DataDictionary.html
Onto-Builder	University of Savoy; Ontologos	Distinguishes "what contributes to the essence of things and what describes them", defining concepts by their "specific difference". Thus, logical and set-oriented semantics are derived a posteriori.	LOK (Language for Ontological Knowledge) written in Smalltalk	{Web access}	Input: DAML-OIL; XML, LOK Output: DAML+OIL; XML; KIF; Conceptual Graph	Yes, for browsing.	Yes, based on logic and on the specific-difference theory.	User groups.	Yes, for ontologies based on the OK Model.	Lexicon management including synonyms	Extraction of lexicons from texts with OK lexical tools (based on Brill's tagger).	Part of Ontological Knowledge Station. Building OK ontologies is based on a dedicated methodology.	http://ontology.univ-savoie.fr/; http://www.ontologos.com

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
OntoEdit	Ontoprise GmbH	F-Logic axioms on classes and relations; algebraic properties of relations; creation of metadata; limited DAML property constraints and datatypes; no class combinations, equivalent instances.	F-Logic	Resource URIs	RDFS; F-Logic; DAML+OIL (limited); RDB	Yes, via plug-in	Yes, via OntoBroker	Transaction locking at the object and whole subtree levels.	Yes	Multiple lexicons via plug-in	No	Free and commercial (Professional) versions are available, with continuing development of the commercial version.	http://www.ontoprise.de/com/ontoedit.htm
Ontolingua with Chimaera	Stanford Knowledge Systems Lab	OKBC model with full KIF axioms.	Ontolingua	{Web access to service.}	Import & Export: DAML+OIL; KIF; OKBC; Loom; Prolog; Ontolingua; CLIPS. Import only: Classic; Ocelot; Protégé.	No	Elaborate with Chimaera; Theorem proving (via JTP)	Write-only locking; user access levels.	Semi-automated via Chimaera	Search for terms in all loaded ontologies.	No	Online service only (at http://www-ksl-svc.stanford.edu); Chimaera is being enhanced under DARPA funding in 2002.	http://www.ksl.stanford.edu/software/ontolingua/ http://www.ksl.stanford.edu/software/chimaera/
Ontology Builder & Server	VerticalNet, Inc.	Classes with slots, datatypes and cardinality constraints; node documentation; inclusion. No axioms.	Partial OKBC model	Fully qualified names; {HTTP browser and server}	RDFS & DAML+OIL (future)	No	Limited to term validity and graph cycles.	User roles and security; global locking.	Simple difference and merge process.	Yes	No	Currently available only as part of their enterprise solution product.	http://www.verticalnet.com/technology/components/process.html
Ontology Directed Extraction (ODE) Tools	XSB, Inc.	Multiple inheritance subsumption class hierarchies. Support for typed attributes of classes and relations between classes. Supports schema and object information.	Tabled Prolog	No	No	No	Yes	No	Yes, with limitations.	Yes	Yes	Tool supports construction of domain ontologies used to guide lexical classification and information extraction.	http://www.xsb.com/ode.asp

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Ontopia Knowledge Suite	Ontopia AS	Constraint modeling specifically and solely for Topic Map representations.	Ontopia Schema Language (OSL)	{Web access and Web API}	OSL; XTM; LTM (import only); HyTM	No, but tree view.	Validation against the OSL schema.	Full concurrency and transaction support when running with RDBMS.	For ontologies and instance data, but not (currently) for constraints.	Full-text search	No, but application framework allows this.	Although primarily an IDE for Topic Map applications, the framework supports ontologies.	http://www.ontopia.net/solutions/products.html
Ontosaurus	USC Information Sciences Institute	Rich KB browser with simple editing; contexts; same-class-as; metaclasses.	Loom	{HTTP browser}	KIF; Loom; OKBC	Browsing class hierarchy	Yes	Global locking	No	No	No	Online access to KBs hosted on CL HTTP server.	http://www.isi.edu/isd/ontosaurus.html
OntoTerm	University of Malaga	Concept and property hierarchies with concept instances; properties distinguished as attributes or relations. Metadata (natural language definitions).	n/a	{HTML output}	n/a	No, but cross-linked tree views indicate legal element associations or types, and allow editing.	No	No	No	Word lists	No	Although intended to be a terminology management system, OntoTerm can be used for general ontology development. Ongoing development and support of the software is unknown.	http://www.ontoterm.com/
OpenCyc Knowledge Server	Cyc Corp.	FOPC extended with contexts, equality, default reasoning, skolemization, quantification over predicates. (Basic ontology editing via KB Browser Create Term tool.)	CycL (& SubL)	{HTTP server}	DAML+OIL (native KB only)	No	Directed inferencing and queries; truth maintenance	Yes	No	Yes, via Cyc-NL with KB-linked lexicon for syntactic and semantic disambiguation.	English parsing possible with Cyc-NL.	Knowledge base subset and browser only. Future release of ontology building tools: Template-based knowledge entry, Index Overlap, Similarity Tool and Salient Descriptor.	http://www.opencyc.org/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
OpenKnoMe	University of Manchester Medical Informatics	Description logic terminological modeling without support for individuals or type system. Arbitrarily complex structures may be composed from primitive concepts and relations. Role hierarchy with inverses, and reasoning over relationships such as part-of. No formal negation, disjunction or conjunction. Limited support for cardinality. No reasoning over numbers or ranges. Toolset for managing intermediate representations.	GRAIL	Not as configured.	CLIPS; XML	No	Logical coherence ala DL and a meta-model system for declaring inherited semantic constraints and permissions. Also, declarative query language (GQL) can be used to author checks of modeling consistency.	User roles and read/write privileges; version control. Users see each other's changes only when they check modules back in.	Via explicit mappings (reifications) to GALEN Common Reference Model. Focus is on linking rather than mapping to reference model.	Can use GALEN language module that links its concept identifiers with synonyms and word forms, and provides segment grammar for semantic links.	No	Although, developed primarily as a medical terminology model builder, the tool can serve as a general purpose ontology editor. Currently requires OpenGALEN terminology server and CINCOM VisualWorks runtime environment.	http://www.topthing.com/
PC Pack 4	Epistemics Ltd	Knowledge acquisition and modeling. Multiple inheritance; n-ary relations; rules and methods. User definable templates for modeling formalisms like CommonKADS and Moka.	XML	{HTML output via XSLT}	XML	ER diagrams; class hierarchies; OO views	Only logically consistent models can be created.	Yes	No	No	No	Suite of many integrated KADS inspired tools.	http://www.epistemics.co.uk/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Protégé-2000	Stanford Medical Informatics	Multiple inheritance concept and relation hierarchies (but single class for instance); meta-classes; instances specification support; constraint axioms ala Prolog, F-Logic, OIL and general axiom language (PAL) via plug-ins.	OKBC model	Limited namespaces; {can run as applet; access through servlets}	RDF(S); XML Schema; RDB schema via Data Genie plug-in; (DAML+OIL backend due 4Q'02 from SRI)	Browsing classes & global properties via GraphViz plug-in; nested graph views with editing via Jambalaya plug-in.	Plug-ins for adding & checking constraint axioms: PAL; FaCT.	No, but features under development.	Semi-automated via Anchor-PROMPT.	WordNet plug-in; wildcard string matching (API only).	No	Support for CommonKADS methodology.	http://protege.stanford.edu/index.html
RDFAuthor	Damian Steer	Create RDF instance data against RDFS schemas.	RDF	URIs; {Web links; remote RDF query}	XML; RDF	Creating and editing instances as graphs.	RDF errors	No	No	No	No	Currently available for Mac OS X; also as a Java Swing application. Additional output: SVG, PNG, TIFF, PDF.	http://rdfweb.org/people/damian/RDFAuthor/
RDFedt	Jan Winkler	Textual language editor only.	RDF model	RSS	RDFS; DAML; OIL; Shoe	No, but tree view.	Writing mistakes only.	No	No	No	No	None	http://www.jan-winkler.de/dev/e_rdfe.htm
SemTalk	Semtation GmbH	Subset of RDFS and DAML extended with inverse relations and process modeling.	Visual Basic	URI namespaces; {distributed development}	XML; F-Logic; ARIS models; Bonapart models	Yes, for design and browsing.	Subsumption and name usage across multiple models; meta-model specific checks.	No	Yes, with simple filtering.	Synonyms; homonyms; stop words; some POS; glossaries via Babylon.	No, but interfaces to appropriate Ontoprise and TextTech products.	Microsoft Visio extension and SmartTags. Additional output include: Rational Rose UML class diagrams, RDF annotated HTML, MS Excel, MS Project, SAP IPC, HTML/VML.	http://www.semtalk.com/

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
Specware	Kestrel Technology	Logical and functional axioms. (Text based language editor only.)	Metaslang	No	None	No	Proofs via Gandalf and SNARK.	No	Yes, via composition operations (e.g., co-limits).	No	No	While primarily a tool for the formal, compositional specification of software, Specware can be used to define domain theories.	http://www.specware.org/
SymOntos	Institute for the Analysis of Information Systems - CNR (Italy)	XML Schema modeling constructs with subsumption of classes and relations; specified relation types of isa, part-of, similarity and predicate. Business-oriented predefined classes such as: actor, process, event, and message.	XML	{Web access}	XML; RDF(S)	(Planned release 4Q'02)	Concept hierarchy validity, range restrictions and graph cycles.	Simple user groups	Possible via XML encoding.	Word lists of synonyms; term query support.	No	Online service; academic level support; can support collaborative ontology building. SymOntoX version in progress with language for process, actor, event and goal.	http:www.symontos.org
Taxonomy Builder	Semansys Technologies	General taxonomy of elements assigned data types and substitution groups. Predefined XBRL relation types via links.	XML Schema	XML namespaces; {taxonomy browser; Internet client}	XML; XML Schema	No	Yes, relative to XBRL core schema.	No	Yes	No	No	Available separately or as part of the Semansys XBRL Composer Professional. Additional outputs include CSV, TXT and SQL.	http://www.semansys.com/about_composer.html

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
TOPKAT	AIAI, University of Edinburgh	Supports representation of the various models of CommonKADS (circa 1995). Underlying these models are dictionaries of concepts, properties, property values, inferences, and tasks. Production rules can be represented using a combination of these primitives.	HARDY and CLIPS	No	CML	Native graph view for editing.	Limited	No	No, except for models within a single ontology.	Term equivalence through the data dictionary.	Simple natural language parser can identify possible concepts and property values in a protocol transcript.	The Open Practical Knowledge Acquisition Toolkit (TOPKAT) supports CommonKADS knowledge acquisition techniques including: laddered grid, card sort, repertory grid, protocol analysis. Final diagrams also output in HTML. No support.	http://www.aiai.ed.ac.uk/~jkk/topkat.html
Visio for Enterprise Architects	Microsoft Corp.	Most object-role modeling (ORM) constructs, but imposes relational logical constraints on specification.	ORM	No	XML (via add-on); DDL	ORM class diagrams	Yes	Yes	Yes	No	No	ORM modeler may be effective for specifying domain ontologies; part of Visual Studio.NET Enterprise Architect	http://msdn.microsoft.com/vstudio/techinfo/articles/developerproductivity/orm.asp
WebKB	Distributed Systems Technology Centre (DSTC), Australia	Basic conceptual graph modeling and manipulation that includes contexts, constraint checking and querying. Can derive new statements (e.g., relations) from necessary and sufficient conditions.	FS (extended CGs)	URIs for element references; Web access of KBs; {CGI and HTML interfaces; Web script language}	Export (partial only): DAML/RDF; CGIF; KIF	Hyperbolic-like browsing (of taxonomies only) via KVO's OntoRama.	Syntactic and logical including transitive cycles, disjoint relations, relation signatures. Also lexical checking.	KB sharing restricts editing so each element has an associated author.	No, but separate ontologies can share the same KB "framework" including a WordNet based upper ontology.	WordNet nouns and adjectives; aliases; element searching by author or name.	No	On-line service (www.webkb.org); also source and binary code available.	http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/generalDoc.html

Tool	Source	Modeling Features/Limitations	Base Language	Web Support & {Use}	Import/Export Formats	Graph View	Consistency Checks	Multi-user Support	Merging	Lexical Support	Information Extraction	Comments	More Information
WebODE	Technical University of Madrid UPM	Concepts (class and instance), attributes and relations of taxonomies; disjoint and exhaustive class partitions; part-of and ad-hoc binary relations; properties of relations; constants; axioms; and multiple inheritance. Inference engine for subset of OKBC primitives and axioms.	Prolog translation of FOL and frames per OKBC model.	URIs as imported terms; {browser client}	DAML+OIL; RDFS; X-CARIN; FLogic; Prolog; XML	Native graph view with editing of classes, relations, partitions, meta-properties, etc.	Type and cardinality constraints; disjoint classes and loops, taxonomy style (OntoClean), etc.	Yes, with synchronization; authentication and access restrictions per user groups.	Unsupervised (ODEMerge methodology) using synonym and hyperonym tables; custom dictionaries and merging rules.	Synonyms and abbreviations; (EuroWordNet support under development)	Using WebPicker (UNSPSC, RosettaNet)	Supports Methontology methodology (Fern‡ndez-L—pez et al, 1999); offered as online service; successor to ODE; ontology storage in RDB.	http://delicias.dia.fi.upm.es/webODE/
WebOnto	Knowledge Media Institute of Open University (UK)	Multiple inheritance and exact coverings; meta-classes; class level support for prolog-like inference.	OCML	{Web service deployment site}	Import: RDF; Export: RDFS, GXL, Ontolingua, OIL	Native graph view of class relationships.	For OCML code	Global write-only locking with change notification.	No	No	(Available from OCML based tool MnM.)	Online service only.	http://kmi.open.ac.uk/projects/webonto/

NOTE	Concept	Instance	Relation
We frequently elected to retain the words of the software provider in these tool descriptions. Consequently, the alternative terms listed to the right may be used with roughly the same meaning.	Concept, class, category, type, term, entity, set and thing.	Instance, individual, resource, extension, description, object and entity.	Relation, relationship, property, function, role, slot, attribute, association, criterion, constraint of, feature and predicate.

Above is shown survey ontology table comparing all the current Ontology editors. This picture has been taken from: [*http://xml.com/2002/11/06/Ontology_Editor_Survey.html*]

APPENDIX_5 – Merging the recipes ontology with the wines Ontology

5.1 Wines Ontology description

First of all, the wine Ontology has been has been carefully studied. This chapter presents an explanation of the wines Ontology:

Each wine belongs to a region. There are several kinds of regions, following this schema:

WINE-REGION
- AUSTRALIAN-REGION
- FRENCH-REGION
  - BORDEAUX-REGION
    - MEDOC-REGION
  - BOURGOGNE-REGION
  - LOIRE-REGION
- ITALIAN-REGION
- US-REGION
  - CALIFORNIAN-REGION

The picture below shows this classification in a graphical way:

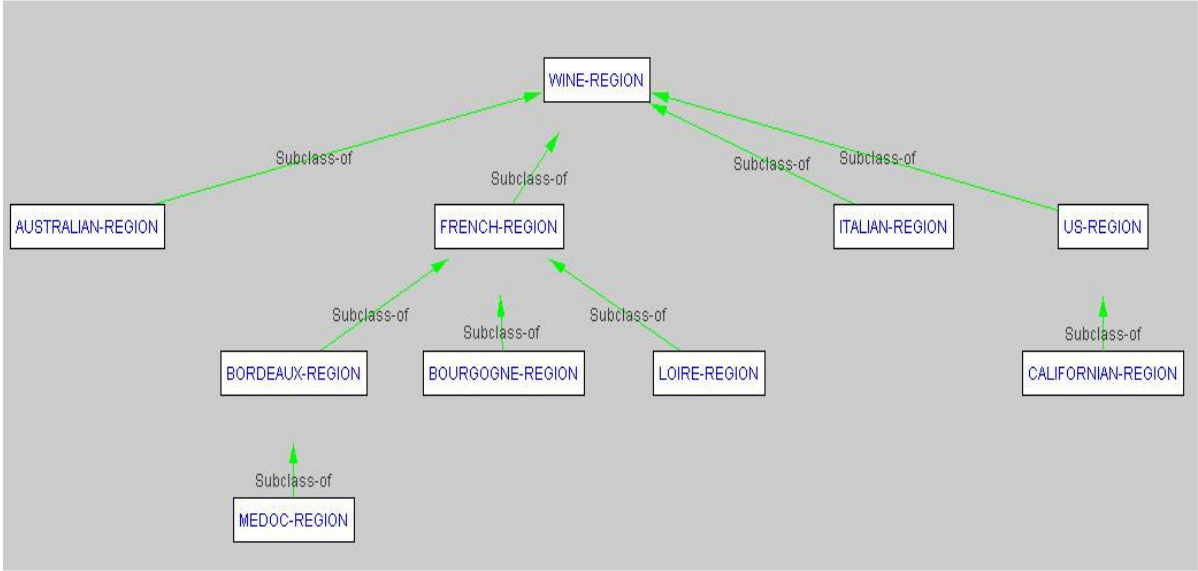


Figure 1 – Wines Region

There are defined several wines properties:

Color restrictions	Sugar	Flavor	Wine body
RED-COLOR-RESTRICTION	DRY-SUGAR	DELICATE-FLAVOR	FULL-BODIED-WINE
			LIGHT-BODY
ROSE-WINE	SWEET-OR-OFF-DRY-SUGGAR	MODERATE-OR-STRONG-FLAVOR	LIGHT-OR-MEDIUM-BODY
WHITE-COLOR			MEDIUM-OR-FULL-BODY

The next picture graphically shows these features:

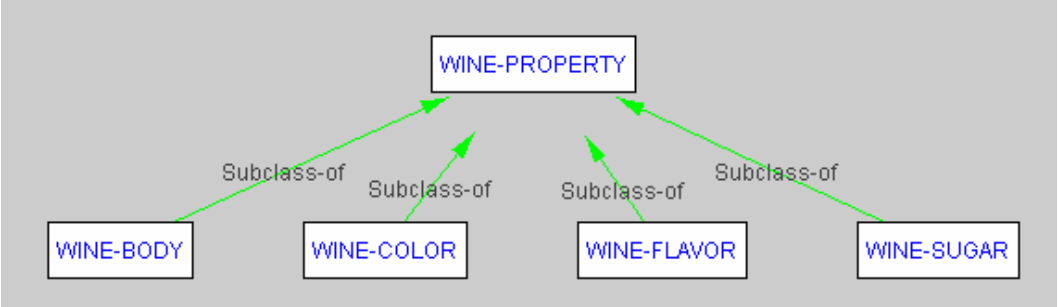


Figure 2 – Wines Properties

There are also defined other different wines characteristics, like the different kinds of wine (table-wine, dry-wine…) grape restrictions, wine origin (Italian, Californian …), names of the wine (Bordeaux, sauterne…), and some others.

5.2 Comparison of the wines the Ontology skeleton

As the recipes and the wines Ontologies have several different elements, these can not be compared. What is going to be studied and where the merging process resides is in the common or related parts of the taxonomy. The other features will be straight tipped out into the new Ontology. The common parts are the ones referred to the food taxonomy. Each classification is shown in the next table. This structure has to be carefully studied in order to find the related concepts among both Ontologies (whether to find synonyms or to find parents, children and siblings among all the concepts)

FOOD TAXONOMY IN THE WINES ONTOLOGY	FOOD TAXONOMY IN THE RECIPES ONTOLOGY

CONSUMABLE-THING

EDIBLE-THING

FRUIT

NON-SWEET-FRUIT

SWEET-FRUIT

GRAPE

WINE-GRAPE

DESSERT

CHEESE-NUTS-DESERT

SWEET-DESSERT

FOWL

DARK-MEAT-FOWL

LIGHT-MEAT-FOWL

MEAT

RED-MEAT

SPICY-RED-MEAT

NON-SPICY-RED-MEAT

NON-RED-MEAT

PASTA

PASTA-WITH-RED-SAUCE

PASTA-WITH-NON-SPICY-RED-SAUCE

PASTA-WITH-SPICY-RED-SAUCE

PASTA-WITH-WHITE-SAUCE

PASTA-WITH-HEAVY-CREAM-SAUCE

PASTA-WITH-LIGHT-CREAM-SAUCE

OTHER-TOMATO-BASED-FOOD

SEAFOOD

FISH

BLAND-FISH

ingredient

food

baking_supply

leaven

yeast

cereal_grain

cereal

wheat

grain

cacao

coffee

condiment

oil

sauce

tomato sauce

vinegar

dairy_product

butter

cheese

cream

milk

yogurt

dairy_product_substitute

butter_substitute

cheese_substitute

NON-BLAND-FISH

SHELLFISH

NON-OYSTER-SHELLFISH

OYSTER-SHELLFISH

MEAL

MEAL-COURSE

POTABLE-LIQUID

WINE

DESSERT-WINE

SWEET-RIESLING

EARLY-HARVEST

LATE-HARVEST

cream_substitute

milk_substitute

yogurt_substitute

egg

fat_oil

butter

margarine

oil

fish

caviar_roe

crab

fatty fish

lean fish

shellfish

smoked_dry_fish

flavoring

sweetener

cacao

chocolate

milk_chocolate

dark_chocolate

honey

jam

sugar

syrup

- salt
- spice
- herb
  - tea
- fruit
- fresh_fruit
- nut
- legume
- meat
- cured_precooked_meat
- mammal_meat
- poultry_meat
- reptile_meat
- pasta_bread
- bread
- pasta
- stimulant
- cacao
- coffee
- tea
- vegetable
- herb
  - tea
- sea_vegetable
- common_vegetable

soy

tomato

drink

beer

hard_drink

infusion

tea

coffee

juice

fruti_juice

vegetable_juice

milk

soda

water

wine

course

beverage

cocktail

infusion

juice

milk_shake

dessert

non_sweet_dessert

sweet_dessert

fish_course

fatty_fish_course

lean_fish_course

seafood_course

meat_course

pasta_course

pasta_with_red_sauce

pasta_with_white_sauce

pasta_with_cream

pasta_with_non_cream_white_sauce

rice_course

soup

vegetarian_course

As the names of the concepts are not the same in both Ontologies, and some of them are missing or incomplete in one of them, the following auxiliary knowledge has to be provided: synonyms and hyponyms. Next chapter shows both tables.

Some clues about the similarity: under the EDIBLE-THING classification in the wines Ontology groups all the ingredients and the courses together of the recipes Ontology. The match had to be done carefully not to loose any information, and to keep the structure coherent.

5.3 Auxiliary knowledge to fulfill the Ontologies merging

Table 1 –Synonym Table

Synonims	
fresh_fruit	FRUIT
mammal_meat	MEAT

SPICY-RED-MEAT cured_precooked_meat

LIGHT-MEAT-FOWL poultry

fish SEAFOOD

fatty_fish NON-BLAND-FISH

lean_fish BLAND-FISH

shellfish SHELLFISH

drink POTABLE-LIQUID

wine WINE

pasta_course PASTA

dessert DESSERT

sweet_dessert SWEET-DESSERT

fish_course FISH-COURSE

fatty_fish_course NON-BLAN-FISH-COURSE

lean_fish_course BLAND-FISH-COURSE

seafood-course SEAFOOD-COURSE SHELLFISH-COURSE

All the concepts in each line are synonyms

Hyperonym table

The first concept in each line is the upper concept of all the others (that are at the same level in the classification)

Table 2 –Hyponyms Table

Hyponyms:				
fish	FISH	SELLFISH	caviar-roe	smoked_dry_fish

course OTHER-TOMATO-BASED-FOOD

meat FOWL mammal_meat reptile_meat

course OTHER_TOMATO_BASED_FOOD_COURSE FRUIT-COURSE

pasta_with_red_sauce PASTA_WITH_SPICY_RED_SAUCE_COURSE PASTA_WITH_NON_SPICY_RED_SAUCE_COURSE

pasta_with_cream PASTA-WITH_HEAVY-CREAM-COURSE PASTA-WITH-LIGHT-CREAM-COURSE

non_sweet_dessert CHEESE-NUTS-DESSERT-COURSE

FRUIT-COURSE SWEET-FRUIT-COURSE NON- SWEET-FRUIT-COURSE

meat_course DARK_MEAT_FOWL_COURSE DARK_MEAT_FOWL_COURSE RED-MEAT-COURSE NON-RED-MEAT-COURSE

seafood-course NON-OYSTER-SHELLFISH-COURSE OYSTER-SHELLFISH-COURSE

5.4 Resulting Ontology structure

After all these settings, both Ontologies can be merged with the Ontology editor. The resulting Ontology has all the single features of the original Ontologies, and the next mixed classification for the course taxonomy (the common parts)

Table 3 – Final Course Taxonomy

Final course-taxonomy after the merging
course
Pasta_course
Pasta_with_red_sauce
PASTA_WITH_SPICY_RED_SAUCE_COURSE
PASTA_WITH_NON_SPICY_RED_SAUCE_COURSE
Pasta_with_white_sauce
Pasta_with_cream

PASTA-WITH_HEAVY-CREAM-COURSE

PASTA-WITH-LIGHT-CREAM-COURSE

Pasta_with_non_cream_white_sauce

soup

rice_course

vegetarian_course

dessert == DESSERT-COURSE

sweet_dessert == SWEET-DESSERT-COURSE

non_sweet_dessert

CHEESE-NUTS-DESSERT-COURSE

FRUIT-COURSE

SWEET-FRUIT-COURSE

NON- SWEET-FRUIT-COURSE

meat_course

DARK_MEAT_FOWL_COURSE

DARK_MEAT_FOWL_COURSE

RED-MEAT-COURSE

NON-RED-MEAT-COURSE

fish_course == FISH-COURSE

fatty_fish_course == NON-BLAN-FISH-COURSE

lean_fish_course ==BLAND-FISH-COURSE

seafood-course == SEAFOOD-COURSE == SHELLFISH-COURSE

NON-OYSTER-SHELLFISH-COURSE

OYSTER-SHELLFISH-COURSE

OTHER_TOMATO_BASED_FOOD_COURSE

beverage

milk_shake

cocktail

infusion

juice

Some new categories have appeared after the merging process. For example the FOWL category was missing in the recipes Ontology; it had into account only the poultry_meat (equal to the LIGHT-MEAT-FOWL), but the DARK-MEAT-FOWL was missing (this was not a mistake, but a way to simplify the Ontology for the IE purpose). But as long as the wines Ontology need this classification, it is added.

APPENDIX_6 – Kinds of relationships.

6.1 Topological inclusion

The container surrounds the parts spatially or temporally. It is not going to be used in the recipes context. No relationships of this kind were identified.

6.2 Classification inclusion

The objects are members of the set of objects. The only classification relationship found is the one between a single recipe and the whole amount of recipes treated.
Example: *"A single recipe is an instance of the recipes database collection"*

6.3 Attribution

Some properties can be considered as objects in the model, or as attributes of an object. If the concept is relevant for the domain description should be a component of the model, if not should be modeled like an attribute of another object. This is sometimes a subjective decision depending on the developer's point of view.
Example: *"shape is a property of a food"*

6.3.1 Attachment

Describes the things that are attached to one object, but they are not part of it and do not give any functional support for the object they are attached to. No example of this kind of decomposition was found in the recipe's domain

6.3.2 Ownership

The owner is in possession of the object (but the object is not part of the owner)
No example of this kind of decomposition was found in the recipe's domain

6.3.3 Composition

There are different kinds of composition, also called aggregation.
Composition: *"is the act of making up an object by putting together the parts of it"*
It reduces the complexity of the model by grouping different objects with similar characteristics in one object, and then treating all as only one object
The kind of composition depends on the value of the next three basic properties:

- Whether the parts have a functional or structural relationship to the object they compose.

- Whether the parts are made of the same thing as the whole
- Whether the parts can be broken up from the whole (extrinsic relationship) or not (intrinsic relationship), and the whole still exists

A brief explanation of each one along with some examples about the recipe's context is shown below.

6.3.3.1 Different kinds of composition-relationship characteristics and examples

Kind of composition	Brief definition	Intrinsic/ Extrinsic	Functional/ Structural	Same nature	Key word	Relation
Component-integral object composition	Defines the parts of the object – Parts can be tangible, abstract, organizational or temporal	Extrinsic	Both	No	Part of	*"The way of doing part is part of a recipe"* *"The nutritional information part is part of a recipe"* *"The ingredient description part is part of a recipe"*
Material-object composition	Defines what the objects are made of	Intrinsic	Structural	No	Partly or entirely	*"Cappuccino is partly milk" "Bread is partly flour"* *"Chocolate is partly cacao"* *"Tomato sauce is partly tomato"*
Portion-object composition	Parts and whole have the same things in the same proportion compounding them. Parts inherit the whole's properties. (Except from the quantities) Parts (portions) can be divided and measured (in liters, hours, minutes, kilograms, etc) Parts and whole can be added, subtracted, multiplied,divided..(Arithmetic operations)	Extrinsic	Structural	Yes	Portion of	*"A slice of bread is a portion of a loaf of bread"*
Place-area composition	It is often used to identify places and locations inside them. The peaces are similar in nature but they can not be separated from their area (the whole)	Intrinsic	Structural	Yes	Part of	No composition of this kind was identified in the recipes context.

Member-bunch composition	The parts are a collection that define the whole - The relationship is based on spatial, temporal or social connection	Extrinsic	Both	No	Part of	No composition of this kind was identified in the recipes context.
Member-partnership composition	Invariant collection of parts forming a whole – the parts can not be removed from the whole without destroying it	Intrinsic	Functional	No	form	No composition of this kind was identified in the recipes context.

6.3.3.2 Composition properties: Transitivity

What makes so important the composition relationships is the transitivity relationship it sometimes has. Additional knowledge can be inferred from a transitive relationship:

Definition: "If A has a relationship with B and B has a relationship with C then we can infer that A has that relationship with C."

Composition relationship is sometimes transitive. But we have to check carefully if both premises are based in the same kind of composition relationship.

- If they have the same kind of composition relationship they often produce a valid composition-related conclusion (but sometimes it may be wrong)

- Two different kinds of composition relationships in the premises often conclude a wrong sentence, but it can be sometimes right.

  Example1:

  An ingredient is partly vitamins (Material-object composition)
  An ingredient is part of a course (Component-integral object)

  Conclusion: A vitamin is part of a course. This is wrong conclusion, because although some vitamins are part of the final dish, some others are lost during the cooking.

  Example2:

  The loaf is partly flour (material-object)
  A slice of bread is part of a loaf of bread (portion-object)

  Conclusion: A slice of bread is partly flour. This conclusion is correct although it is different relationships

The transitivity property is very useful to propagate operations in composition relationships.
If some ingredients are part of the ingredient description,
And that description is part of a recipe,
It can be concluding by the transitivity property that ingredients are part of the recipe.

When the composition relationship is the same the propagation can be made to as many levels as we want. But if they are not the same kind, each level has to be studied carefully for validity.

When modelling the domain, the first step is to identify the different kinds of relationships related above. It will help to find out the utility of these different relationships (to identify whole-part associations and help us while developing the system)

6.3.4 Inheritance

The inheritance relationship is modeled in the diagrams as the IS-A relationship. This is normally the main bone of a diagram. Then the other kind of relationships can be added in order to complete this taxonomy. This approach, presented in [Andreasen et al., Nilsson 2001] is to divide the domain having the ISA structure as the backbone of the classification, and afterwards divide the generated groups basing on other kind of relationships, such as partitive, causative, locative, temporal, etc…

The parent elements have the common basic structure and characteristics that their children will inherit. The children elements will have their parents' characteristics plus their own ones (they specialize the parents' elements)

In this process the most general top category is analyzed and progressively specialized in more concrete concepts, ending up in the leaves of the ontology, which will be the instances of the lowest entity in the hierarchy they inherit from.

Example:

T == Ingredient → animal_origin → sea_animal_origin → fish → flesh → salmon

The sequence above represents a path down through the ontological ingredient structure, where the entity *ingredient* represents the top element in the hierarchy and *salmon* is an instance of the last entity *flesh*

But the ISA relationship has to be used carefully in order to make a consistent diagram, many a very common mistake while developing these kinds of diagrams is to inherit from a class with different characteristics like ours.

Summing up: the most important kinds of decomposition are the inheritance and the composition ones, and are the ones in which the Ontology model is based.

APPENDIX_7 – Course and ingredient classification

Course classification

course

- beverage
  - cocktail
  - infusion
  - juice
  - milk_shake
- dessert
  - non_sweet_dessert
  - sweet_dessert
- fish_course
  - fatty_fish_course
  - lean_fish_course
  - seafood_course
- meat_course
- pasta_course
  - pasta_with_red_sauce
  - pasta_with_white_sauce
    - pasta_with_cream
    - pasta_with_non_cream_white_sauce
- rice_course

soup

vegetarian_course

Ingredients classification
food

baking_supply

leaven

yeast

cereal_grain

cereal

wheat

grain

cacao

coffee

condiment

oil

sauce

tomato sauce

vinegar

dairy_product

butter

cheese

cream

milk

- yogurt
- dairy_product_substitute
  - butter_substitute
  - cheese_substitute
  - cream_substitute
  - milk_substitute
  - yogurt_substitute
- egg
- fat_oil
  - butter
  - margarine
  - oil
- fish
  - caviar_roe
  - crab
  - fatty fish
  - lean fish
  - shellfish
  - smoked_dry_fish
  - flavoring
  - sweetener
    - cacao
    - chocolate

milk_chocolate

dark_chocolate

honey

jam

sugar

syrup

salt

spice

herb

tea

fruit

fresh_fruit

nut

legume

meat

cured_precooked_meat

mammal_meat

poultry_meat

reptile_meat

pasta_bread

bread

pasta

stimulant

cacao

coffee

tea

vegetable

herb

tea

sea_vegetable

common_vegetable

soy

tomato

drink

beer

hard_drink

infusion

tea

coffee

juice

fruti_juice

vegetable_juice

milk

soda

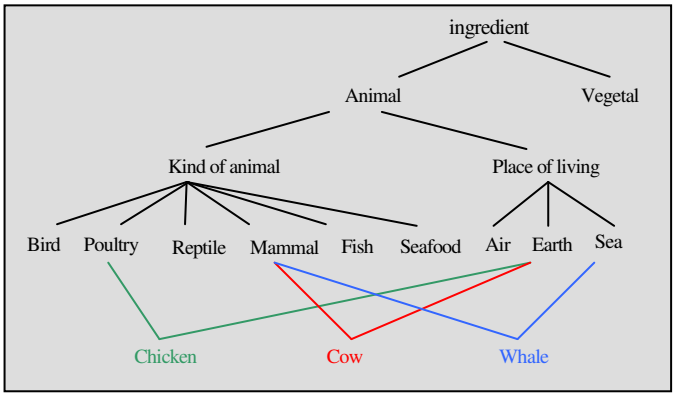
water

wine

APPENDIX_8 - Transforming multiple-inheritance into simple-inheritance.

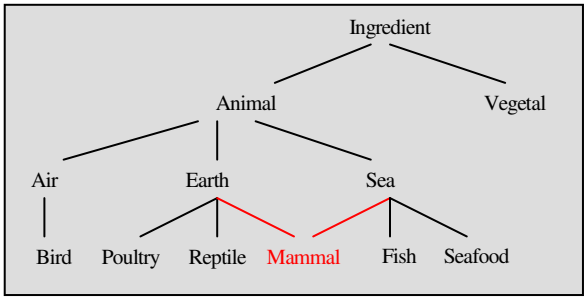
8.1 Transforming no-relevant features into attributes

The following example shows the ingredient taxonomy divided with the IS-A relationship *kind of origin (animal or vegetal).* Some knowledge about the place where the animal lives has been added as well, in order to improve the classification. The resulting taxonomy is shown in the next picture:



This classification has a poly-hierarchical structure. The leaf entities inherit from a *kind of animal* entity and from a *place of living* entity.

Some reorganization can be done to avoid some of the multiple inheritance:



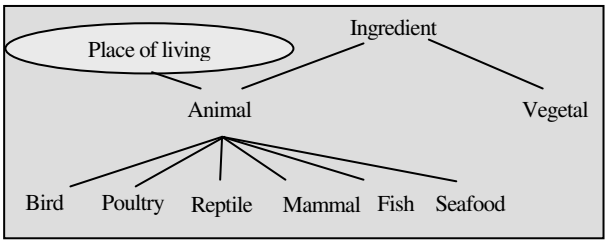
But the classification still has a poly-hierarchical structure due to the entity *Mammal*. Mammals normally live in the earth, but the instance *whale* lives in the sea (this is called a boundary case). The objective is to create a classification where the instances inherit from only one entity.

At this point two solutions are possible: to duplicate the entity whale in both earth and sea entities, or to transform one of the classification criteria (*place of living* or *origin*) into an attribute of the other.

When this classification was reconsidered, *place of living* was found not an important feature to bear in mind when studying a recipe content.

For example: also for the boundary case *whale*, it does not matter where it comes from. Although it lives in the sea, its taste resembles more to meat than to fish (which is the culinary point of view). (Moreover, the place of leaving of a vegetal ingredient has no sense)

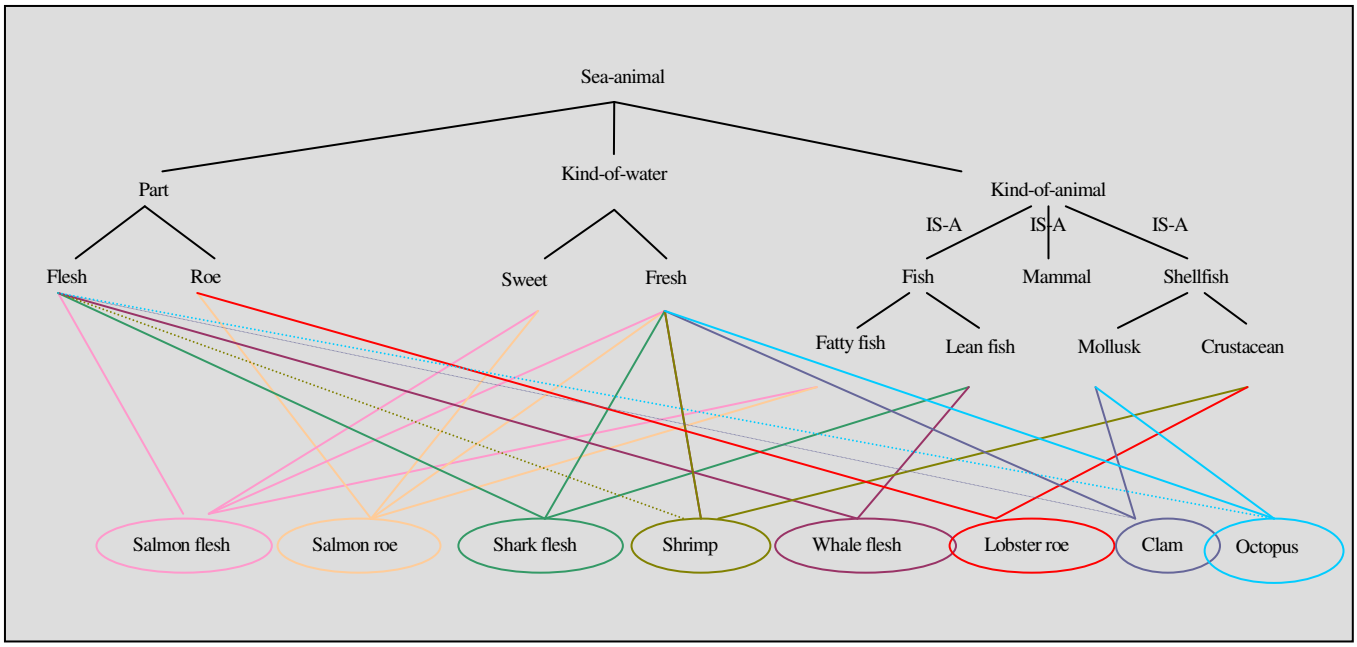
Due to the place of living is not a basic feature to bear in mind, it can be removed as a classification term and put as an attribute. The classification will now look like in the following picture:



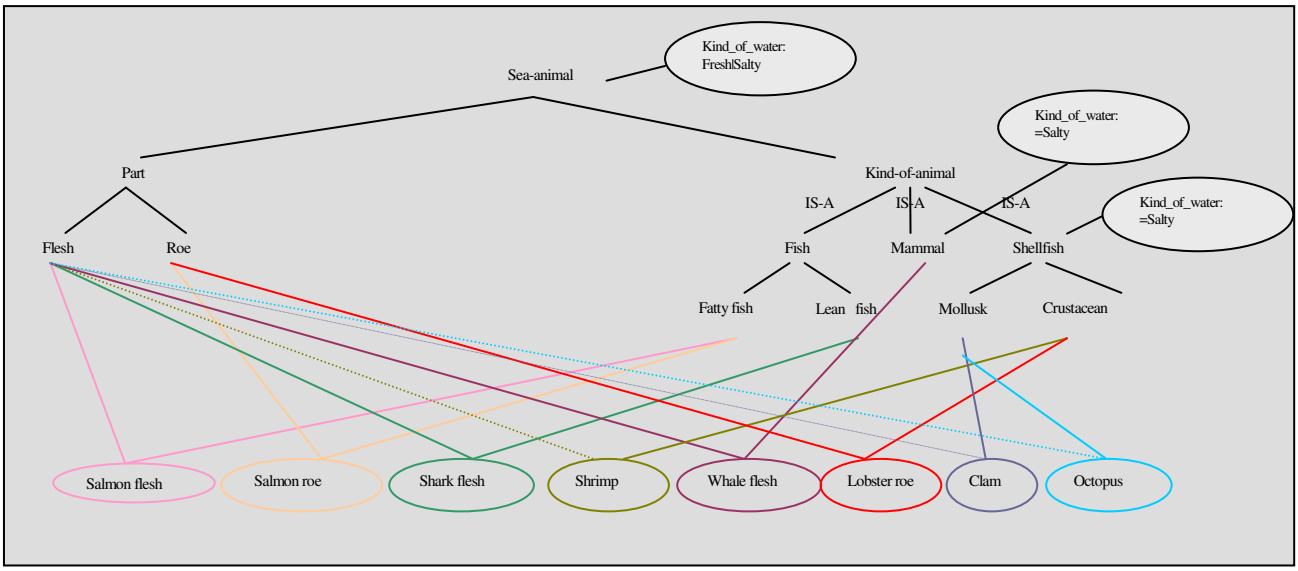
8.2 Transforming no-relevant features into sub-classifications

The next diagram shows an example of how to transform a multiple-hierarchical classification made in the *sea-animal* context.

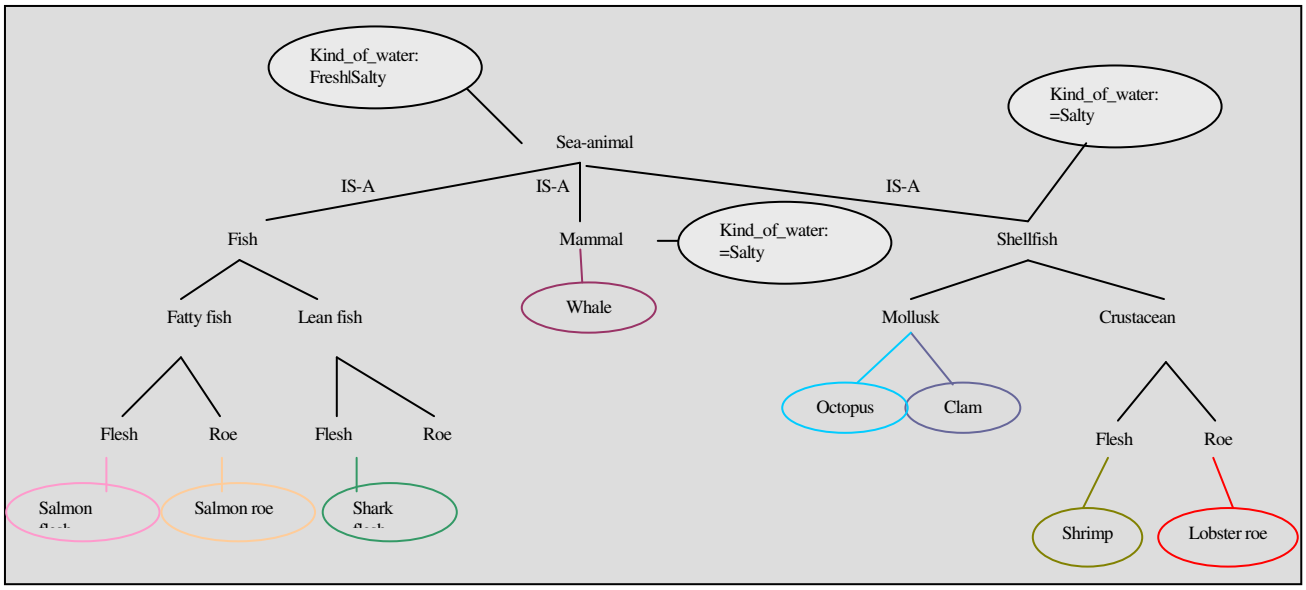
Imagine a first stage, when the classification is made following three different criteria. The next picture shows how the taxonomy would look like:



If the multiple-inheritance should be removed, the first step is to analyze the subdivision relationships and find-out the less important one. In this case the kind-of-water relationship is considered no crucial. In this first stage the modelling as an attribute has been chose (like in the first example), as shown in the next picture:



Finally, the remaining classifications are compared to find the least important one. As explained in the project, the part-of relationship varies depending of the kind of animal it is applied to. This time instead of swapping this characteristics to an attribute, it has been decided to put it as a sub-classification of the IS-A classification.


Kind_of_water:
Fresh|Salty
Kind_of_water:
=Salty
Sea-animal
IS-A
IS-A
IS-A
Fish
Mammal
Kind_of_water:
=Salty
Shellfish
Fatty fish
Lean fish
Whale
Mollusk
Crustacean
Octopus
Clam
Flesh
Roe
Flesh
Roe
Flesh
Roe
Salmon
Salmon roe
Shark
Shrimp
Lobster roe

APPENDIX_9 Relationship between precision and Recall [18]

9.1 Introduction

Retrieval performance is measured in terms of precision and recall.

First of all here are some definitions about Precision and Recall:

- ✓ Precision: "It is the measure of the purity of retrieval"
- ✓ Recall: "It is the measure of the completeness of retrieval"

It has been proved that precision and recall are inversely related: Precision seems to turn down as Recall augments. Although all the researchers would prefer high precision and recall at the same time, its relationship nature is a handicap for this objective. This relationship is a tangent parabola

It is also proved that if the information retrieval is made in more than one step it is possible to improve both parameters at the same time.

Some studies have been made in this field, ones considering precision and recall as continuous functions (Heine-1973, Robertson-1975, Gordon and Kochen-1989) and others considering them as a two-Poisson discrete model (Bookstein-1974)

9.2 Precision and recall overview

The precision and recall definitions are based on two traditional assumptions (although some authors question their veracity)

- ✓ Every retrievable item in a text of study is "relevant" or "not relevant"
- ✓ Information retrieval is an extensive process; the retrieving can be augmented in order to retrieve more items, and hence increasing the recall.

Basing on the first assumption, all the retrievable items are classified following the table below (they belong to one and only one cell)

Retrieval matrix	Relevant	Not relevant	TOTAL
Retrieved	N(retrieved∩relevant)	N(retrieved∩~relevant)	Nretrieved
Not Retrieved	N(~retrieved∩relevant)	N(~retrieved∩~relevant)	N~retrieved
TOTAL	Nrelevant	N~relevant	Ntotal

Table 4: The retrieval matrix

This matrix states two possible classifications for an item:
If it is retrieved or not retrieved and if it is relevant or not relevant.

Definition: **Generality**: it is the percentage of texts that are relevant among the whole texts collection to be retrieved information from.

After being familiar with these notions it is easier to understand the following definitions, which are more complete than the first ones:

- ✓ **Recall**: "Is the number of retrieved relevant items (N (retrieved∩relevant)) among the all relevant items in the texts (Nrelevant)".
  So it is calculated by the formula:

$$Recall = N(retrieved \cap relevant) / Nrelevant$$

Table 5: Recall formula

It is a measure of the *effectiveness* in retrieving relevant information from the texts. The number of relevant items in a given set is a fixed number, so it is clear that the higher recall the bigger the relevant retrieved set.

It can happen that the retrieved information set augments (more information is retrieved) but this information is wrong or non-relevant information, so the recall remains the same. But if the entire document is retrieved then a 100% recall is always achieved.

But this is clearly a non-sense, as long as the user is only interested in some relevant information, not the entire text. So a 100% recall is not necessarily good, there is a need to measure the Precision of the retrieved information.

- ✓ **Precision**: "Is the number of retrieved relevant items (N (retrieved∩relevant)) among the whole retrieved items (Nretrieved) that exist in the document"
  So it is calculated by the formula:

$$Precision = N(retrieved \cap relevant) / Nretrieved$$

Table 6: Precision formula

It is a measure of the *purity* in retrieving relevant information from the texts.
It measures the efficiency in extracting relevant information, nor to retrieve irrelevant one.

Like the Recall, a high rate of Precision is desired, but a rate of 100% Precision, does not mean that the information retrieval is necessarily good. Because it can be achieved retrieving just a very few items from the text but all of them relevant, and a lot of useful information can be being lost.
So the objective is to combine high rates of Precision and Recall at the same time. The ideal situation would be to obtain a 100% rate of both simultaneously.

Precision vs. Recall

The empirical cases show the inverse relationship between Precision and Recall. If one of them improves the other one tends to decrease. Here it is the big challenge all the information extraction experts deal with: evade this behavior and obtain good rates of precision and recall.

APPENDIX_10-LP^2 algorithm (Learning Pattern by Language Processing)

The input corpus is a set of texts annotated by the user. Each relevant element is surrounded by SGML (XML in fact) tags. For example:

<quantity> 5</quantity> <measure>kg</measure> <ingredient>rice</ingredient>

The algorithm introduces all the instances annotated by the user in what is called the **positive pool** (contains the positive or relevant examples). There is also a **negative pool**, which contains all the negative examples (the rest of the words in the text)

The algorithm covers all the training examples sequentially; when a new induced rule covers some positive examples, these are removed from the positive examples pool. The induction finishes when the positive pool is empty.

The algorithm makes use of different techniques: Lemmatization [see Glossary], Upper/lower case letters [see Glossary], POS tags [see Glossary], and the gazetteers (synonyms and acronyms)

The training is carried out two steps:

1. The first set of induced rules make no use of linguistic information

    1.1. First of all it induces **tagging rules**. These rules are the ones that will tag information (in order to annotate and then extract it) from new untagged texts.
        The tagging rules are of two kinds:
        1.1.1. **Initial tagging rules**: Are the rules that will annotate future texts
        1.1.2. **Contextual rules**: Complete and improve the initial tagging rules
    1.2. Afterwards **correction rules** are induced. These rules remove or correct mistakes and imprecitions that the previous rules could make while annotating.

2. The second set of induced rules make use of linguistical and aditional information

10.1.1 Induce no linguistic-based rules

10.1.1.1 Tagging rules

The algorithm iterates in the following way: it selects a positive example from the positive pool. Then it builds an initial tagging rule, then it generalizes the rule and finally keeps the k best generalizations (k is set by the user).
All the elements in the positive pool covered by this new rule, are removed from there.

An intial taggin rule have a pattern of conditions in the left side and the right hand is an action to insert XML tags in the texts.

Tagging rule *to insert annotations in the text*		
Left part		Right part
Pattern of conditions on some groups of connected words	→	Action that **insert**s an **XML tag** in the text

The left part of the rule is obtained by the following method:

For each positive example in the text, the algorithm takes the **w** words to the left and **w** words to the right, and makes a rule with this. For example:

One rule only inserts one simple tag (e.g: <kg>), no pairs of XML tags. (Opening tag and closing tag).

So the other kind of tagging rules, the contextual rules are applied afterwards to complete the initial rules, adding the closing tags for example. </kg>

10.1.1.2 Correction rules

This is another kind of rules that do not insert any tag on the corpus. They only correct mistakes and vagueness of the tagging rules.

These rules are induced like in a classic wrapper induction system (WIS), making no use of linguistic information at all. It is now when the algorithm improves the result with additional information:

10.1.2 Induce language-based rules

The algorithm adds additional linguistic information to these rules. Amilcare incorporates for that purpose a NLP module, which can not be changed by the user and has a language limitation (English)

The first rules are based on linguistic information only (like in a wrapper induction system). This kind of rules are very suitable for high-structured texts, but not for free texts because their lack of linguistic structure. That is the reason for adding more rules now. These rules make use of the results of the linguistic preprocessor, initial rule patterns, the Ontology associated with the text (user defined classes, its hierarchy (relations), gazetteer (dictionary)…), etc.

Example of some additional information:

Word	Lemma	Lexeme category	Case	Semantic
Chicken	Chicken	Noun	Lower	Ingredient

Incorporating NLP to the Wrapper induction

- Helps to generalise rules beyond the flat word structure. (it limits data sparseness and overfitting in the training phase)
- Rise the efectiveness
- Reduces the training time
- Reduces the corpus size

Summing up; LP2 is a supervised algorithm of WIS (Wrappers Induction Systems) that uses LazyNLP. That is why is good to analyze texts with different kind of information

APPENDIX_11 Rough guidance to create a program to extract information from texts following an Ontology

11.1 Choose the programming language

Fist of all it is necessary to choose the language to implement the program in. This decision is up to the programmer, but some characteristics do have to be taken into account; one important characteristic to have in mind is that the language should support multiple-inheritance if the Ontology schema has multiple-inheritance structure.

Some Object Oriented implementation languages were considered, like Java which can define classes and objects as well but does not support multiple-inheritance. C++ was considered as well but has the contrary problems; it does support poly-hierarchical structures but does not allow operation with the single objects.

Other not Object Oriented implementations should be considered as well. Logic languages like Prolog or regular expressions like Perl5 are discussed in some Ontology approaches papers, to perform well with the task of writing patterns.

These or some other implementation languages can be considered basing on the necessities of the implementation, the skills of the programmer and the available languages at that time.

11.2 Preprocessing the input texts

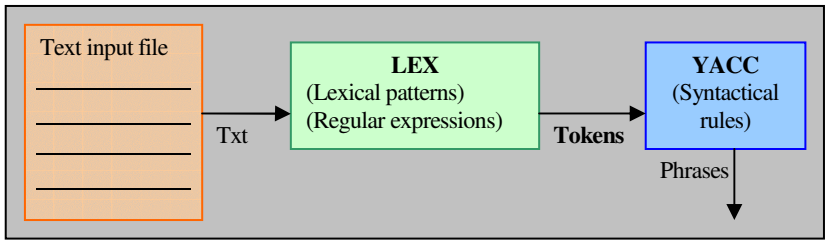
Some preprocessing can be made before treating the input texts.
If more than one recipe per page appears in the input files, it would be easier to use some heuristics to delimit each one and treat them separately.

While preprocessing also the commas, dots, and other punctuation can be removed. The HTML tags can be removed as well with a simple parser, getting plain text without marks, which is easier to extract information from.

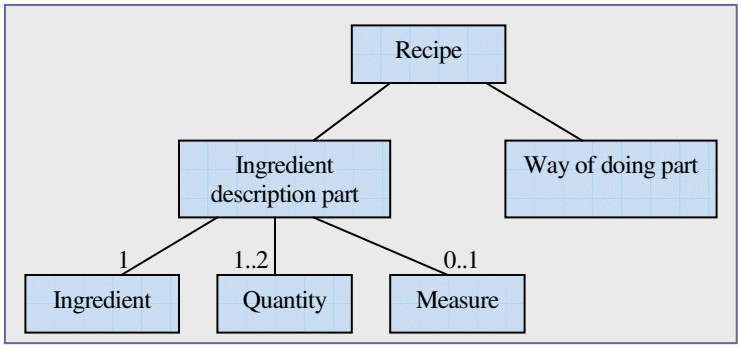
11.3 Parse the input texts to extract the information

Each input recipe has to be parsed to identify and extract the relevant information it contains. A program in the selected language should be made to parse the Ontology. The best way I thought about is to create one class per each entity of the Ontology. Each class will have its associated dictionary and synonyms and patters.

The main program will begin parsing the text. It should recognize the grammar and the lexicons. This parsing can be made easily in a semi-automatic way some lexical and grammatical parsers. Lex and Yacc for example are good tools to borne in mind.



The lexical analyzer parses the text looking for some lexical patters the developer has already defined and some regular expressions. Once identified the entities, then the grammatical analyzer will assure that they meet the grammatical. This grammatical is defined by some production rules. These production rules are made basing on the Ontology structure.
An example of the grammatical of the ingredient description part of a recipe is shown below in pseudo code.



The picture above shows an ER diagram with description of the *Ingredient description part*. It contains one and only one ingredient, cero, one or two quantities and cero or one measure. It is almost straightforward to translate this diagram into a BNF grammar.

<RECIPE>::=<INGREDIENT_DESCRIPTION_PART><WAY_OF_DOING_PART>
<INGREDIENT_DESCRIPTION_PART>

::=<INGREDIENT_DESCRIPTION>

|<INGREDIENT_DESCRIPTION><INGREDIENT_DESCRIPTION_PART>

;

<INGREDIENT_DESCRIPTION>::=<QUANTITY_PART><MEASURE_PART><INGREDIENT>;
<QUANTITY_PART>::= <QUANTITY><QUANTITY_PART>

|<QUANTITY>

```
                    |λ

                    ;
<MEASURE_PART>::= <MEASURE>

                    | λ

                    ;
```

The second production rule states that it can be only one ingredient_description, or more than one (but at least only one). Note that the λ symbol refers to the null value.

Some examples of these production rules in the recipes are shown below

Pattern	Example
(***Quantity***) (***Measure***) (***Name of the ingredient***)	5 liters of water ½ cup flour 1 ½ tablespoon oil
(***Quantity***) **a_separator** (***Quantity***) (***Measure***) (***Name of the ingredient***)	5 or 6 cups of rice 5-6 cups of rice 7 to 9 cups of flour
(***Quantity***) (***Name of the ingredient***)	6 tomatoes 2 egg
(***Quantity) (Name of the ingredient***)	A bit of salt Oil as needed

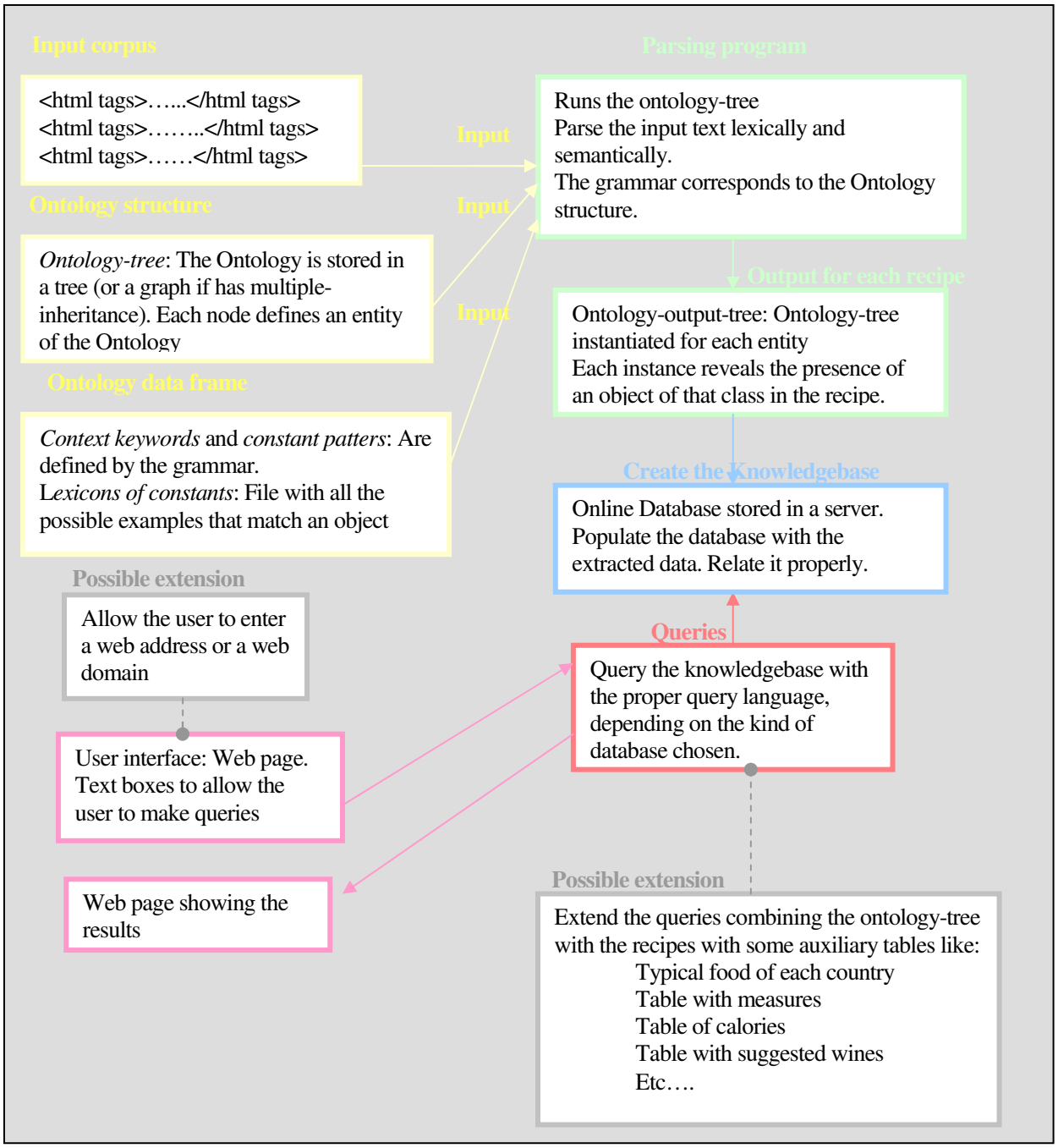
This grammar can be easily transformed into code. Each element of the production rule is transformed into a class in the source code. Each of these classes will parse the text and will call the other classes in the production rule, depending on the token it gets from the lexical analyzer.

An example of how to codify the grammar of this approach in pseudocode

```
Procedure ingredient_description ()

{

        ….

        If quantity()=true then

        {
```

```
                    ….
                    If measure()=true then
                    {
                            if ingredient=true() then
                                    {
                                    }
                                    else error (“not found the correct element”)
                    }
                    else
                    if separtator=true() then
                            {
                            ……
                            }
                            else
                            if ingredient=true() then
                                    {
                                    }
                                    else error (“not found the correct element”)
            }
            else error(“bad definition of ingredient”)
            …..
}
```

An overview of this approach is shown in the next picture:



All the objects that appear in the recipe are hierarchically stored in the tree (or graph) structure

In the running time: each time a class is called and it gets the token it expects form the lexical analyzer, it instantiates an instance of that class. This instance is created calling the constructor of the class which can store the element found (the token), its category (related to the class) the line where it appeared, etc…. Then this element has to be entered into some structured storage. It can be referenced to an instance-tree or instance-graph, where all the instances found in the text are stored and related among them following the Ontology.
Each recipe has its own instance-tree, just to distinguish between the instances of the different recipes.

Each node of the tree is instantiated each time the element it represents is found in the input recipe. Store the position in the text were it has been found.

The **ontology-tree** is a tree of nodes. Each node has a dictionary (a table with names) of words, synonyms, some attributes like found (true or false) and each node calls a method with their same name.

The result of all the recipes parsed can be made as a list of records. Each record is a recipe. Each record will be like following fields:

Recipe's input text (in html)
(file)
URL (string)
Ontology-output-tree with all
the objects filled

Store the identified knowledge in some structured way.

An auxiliary data structure needs to be implemented to store the information of each recipe. This data structure should store the entities, as well as the relationships among this data. This implementation has to be made following the Ontology structure.

The best data structure found for this purpose is a dynamic tree or graph (depending on if the Ontology structure is mono-inheritance or multi-inheritance), implemented with pointers (or some other kind of dynamic storage). As the static structures have a maximum predefined capacity, and the amount of data available on the Web is unpredictable, they were discarded.

Summing up, the program should define a sequence (a dynamic list) to store the information of each recipe (each one stored in a dynamical tree or graph). This means a list of trees or graphs.

Then this knowledge has to be emptied out into the database. Then the information can be queried with any suitable query language.

APPENDIX_12 Gazetteers

This appendix lists the gazetteers generated by the annotation tool, with some editions and modifications to improve them.

12.1 Reduced domain gazetteers

Here is shown a list of the gazetteers generated for the reduced domain of the project.

Table 7

Auto-increment gazetteer

```
<xml version="Melita">

       <concept name="good_for">

       </concept>

       <concept name="ingredient">

       </concept>

       <concept name="fish">

       </concept>

       <concept name="way_of_doing">

              <element occurence="1">In a small bowl, mix together salad
dressing, <drink>milk</drink>, <miscellaneous>sugar</miscellaneous>,
<miscellaneous>vinegar</miscellaneous>, and poppy seeds. Refrigerate until
ready to use. Combine <vegetable>lettuce</vegetable>,
<vegetable>onion</vegetable>, strawberries, <vegetable>pecans</vegetable>,
and red bell <spice>pepper</spice> in a salad bowl. toss with dressing.
</element>

       </concept>

       <concept name="cereal_grain_based">

       </concept>

       <concept name="carbohydrates">

              <element occurence="1">18g</element>
```

```
</concept>
<concept name="meat">
      <element occurence="2">chicken</element>
      <element occurence="1">veal</element>
</concept>
<concept name="difficulty">
</concept>
<concept name="ingredient_description_part">
</concept>
<concept name="drink">
      <element occurence="2">water</element>
      <element occurence="2">milk</element>
</concept>
<concept name="measure">
      <element occurence="2">tablespoon</element>
      <element occurence="5">pieces</element>
      <element occurence="1">pounds</element>
      <element occurence="2">tsp</element>
      <element occurence="1">spoons</element>
      <element occurence="3">g</element>
      <element occurence="1">cloves</element>
      <element occurence="2">cups</element>
      <element occurence="11">cup</element>
      <element occurence="6">piece</element>
</concept>
```

```
<concept name="fat_oil">
      <element occurence="3">oil</element>
</concept>
<concept name="number_of_servings">
      <element occurence="1">6</element>
      <element occurence="1">5</element>
      <element occurence="1">4</element>
      <element occurence="2">8</element>
</concept>
<concept name="cholesterol">
      <element occurence="2">134mg</element>
      <element occurence="1">3mg</element>
</concept>
<concept name="ingredient_description">
</concept>
<concept name="quantity">
      <element occurence="3">1/4</element>
      <element occurence="5">1/2</element>
      <element occurence="1">750</element>
      <element occurence="1">400</element>
      <element occurence="2">5</element>
      <element occurence="1">4</element>
      <element occurence="1">500</element>
      <element occurence="1">3</element>
      <element occurence="8">2</element>
```

```
        <element occurence="1">1/8</element>
        <element occurence="10">1</element>
    </concept>
    <concept name="relation">
    </concept>
    <concept name="retrieved_from">
    </concept>
    <concept name="dairy_produt">
    </concept>
    <concept name="price_person">
    </concept>
    <concept name="fats">
        <element occurence="2">8g</element>
        <element occurence="1">8.3g</element>
    </concept>
    <concept name="concept">
    </concept>
    <concept name="egg">
        <element occurence="2">eggs</element>
    </concept>
    <concept name="miscellaneous">
        <element occurence="2">vinegar</element>
        <element occurence="2">sugar</element>
    </concept>
    <concept name="general_features">
```

```
</concept>

<concept name="posted_by">

</concept>

<concept name="food">

</concept>

<concept name="nutritional_value">

</concept>

<concept name="proteins">

      <element occurence="1">2.7g</element>

</concept>

<concept name="calories">

      <element occurence="1">148</element>

      <element occurence="2">253</element>

</concept>

<concept name="fruit">

      <element occurence="1">orange</element>

      <element occurence="1">strawberries</element>

      <element occurence="1">pineapple</element>

</concept>

<concept name="cooking_time">

      <element occurence="1">20</element>

      <element occurence="1">30</element>

</concept>

<concept name="things">

</concept>
```

```
	<concept name="course">

	</concept>

	<concept name="vegetable">

		<element occurence="1">potatoes</element>

		<element occurence="2">carrots</element>

		<element occurence="2">pecans</element>

		<element occurence="1">garlic</element>

		<element occurence="2">mushrooms</element>

		<element occurence="5">onion</element>

		<element occurence="2">lettuce</element>

		<element occurence="1">peas</element>

	</concept>

	<concept name="spice">

		<element occurence="1">salt</element>

		<element occurence="3">pepper</element>

	</concept>

	<concept name="cereal_grain">

	</concept>

</xml>
```

This is the gazetteer automatically generated by the annotation tool. All the concepts are grouped together in the same gazetteer. It is a task of the user to organize it into each concept's gazetteer. This can be done trough some options provided by the GUI.

Cooking-time

```
<xml version="Melita">

	<concept name="cooking_time">

		<element occurence="1">5 minutes</element>
```

```
            <element occurence="1">1 minute</element>

            <element occurence="1">1 to 2 hours</element>

            <element occurence="1">20 seconds</element>

      </concept>

</xml>
```

Drink gazetteer

```
<xml version="Melita">

      <concept name="drink">

            <element occurence="3">water</element>

            <element occurence="1">milk</element>

      </concept>

</xml>
```

Egg gazetteer

```
<xml version="Melita">

      <concept name="egg">

            <element occurence="1">eggs</element>

            <element occurence="1">egg</element>

      </concept>

</xml>
```

Fruit gazetteer

```
<xml version="Melita">
```

```
	<concept name="fruit">
		<element occurence="1">lemon</element>
		<element occurence="1">peach</element>
		<element occurence="1">walnuts</element>
		<element occurence="1">almonds</element>
		<element occurence="2">pineapple</element>
		<element occurence="1">orange</element>
		<element occurence="1">strawberries</element>
	</concept>
</xml>
```

Measure gazetteer

```
<xml version="Melita">
	<concept name="measure">
		<element occurence="2">pound</element>
		<element occurence="1">ounce can</element>
		<element occurence="5">teaspoon</element>
		<element occurence="1">cloves</element>
		<element occurence="1">can</element>
		<element occurence="1">cloves</element>
		<element occurence="2">teaspoons</element>
		<element occurence="2">can</element>
		<element occurence="1">bell</element>
		<element occurence="2">tablespoon</element>
		<element occurence="3">tablespoons</element>
```

```
            <element occurence="3">glass</element>
            <element occurence="3">cup</element>
            <element occurence="2">ounces</element>
            <element occurence="1">tspns</element>
            <element occurence="1">tspn</element>
            <element occurence="0">kilo</element>
            <element occurence="0">kilogram</element>
            <element occurence="0">gram</element>
            <element occurence="0">liter</element>
            <element occurence="0">deciliter</element>
            <element occurence="0">centiliter</element>
            <element occurence="0">mililiter</element>
      </concept>
</xml>
```

```
<xml version="Melita">
      <concept name="meat">
            <element occurence="1">veal</element>
            <element occurence="5">chicken</element>
            <element occurence="1">beef</element>
      </concept>
</xml>
```

```
<xml version="Melita">
	<concept name="miscellaneous">
		<element occurence="3">vinegar</element>
		<element occurence="1">sugar</element>
	</concept>
</xml>
```

```
<xml version="Melita">
	<concept name="number_of_servings">
		<element occurence="1">8</element>
		<element occurence="1">6</element>
	</concept>
</xml>
```

```
<xml version="Melita">
	<concept name="spice">
		<element occurence="8">pepper</element>
		<element occurence="6">salt</element>
		<element occurence="1">celery</element>
	</concept>
</xml>
```

```
<xml version="Melita">

       <concept name="vegetable">

              <element occurence="2">peas</element>

              <element occurence="1">escarole</element>

              <element occurence="2">potatoes</element>

              <element occurence="1">pecans</element>

              <element occurence="4">garlic</element>

              <element occurence="1">tomato </element>

              <element occurence="2">cucumber</element>

              <element occurence="2">lettuce</element>

              <element occurence="2">carrots</element>

              <element occurence="2">mushrooms</element>

              <element occurence="8">onion</element>

              <element occurence="2">tomatoes</element>

       </concept>

</xml>
```

12.2 Complex domain gazetteers

This are the gazetteers generated by Melita for the big detailed domain. Notice that the

Quantity gazetteer

```
<xml version="Melita">

	<concept name="quantity">

		<element occurence="2">1/3</element>

		<element occurence="2">1 1/2</element>

		<element occurence="16">1/2</element>

		<element occurence="1">16</element>

		<element occurence="1">15</element>

		<element occurence="1">2/3</element>

		<element occurence="3">12</element>

		<element occurence="1">10</element>

		<element occurence="2">l</element>

		<element occurence="1">1 <quantity>1/2</element>

		<element occurence="2">8</element>

		<element occurence="3">5</element>

		<element occurence="1">d</element>

		<element occurence="5">4</element>

		<element occurence="7">3</element>

		<element occurence="5">1/8</element>

		<element occurence="25">2</element>

		<element occurence="56">1</element>

	</concept>
```

```
</xml>
```

Ingredient's gazetteer

```
<xml version="Melita">

       <concept name="ingredient">

              <element occurence="2">bay</element>

              <element occurence="1">(spring onion)</element>

              <element occurence="2">vinegar</element>

              <element occurence="3">ginger</element>

              <element occurence="8">onion</element>

              <element occurence="1">lettuce</element>

              <element occurence="1">chives</element>

              <element occurence="4">tomato</element>

              <element occurence="1">basil</element>

              <element occurence="1">seeds</element>

              <element occurence="3">sugar</element>

              <element occurence="1">mozzarella cheese</element>

              <element occurence="1">sesame oil</element>

              <element occurence="1">sauce</element>

              <element occurence="3">water</element>

              <element occurence="1">oregano</element>

              <element occurence="1">onions</element>

              <element occurence="1">almond</element>

              <element occurence="1">mint</element>

              <element occurence="9">tomatoes</element>
```

```
<element occurence="1">porkchops</element>

<element occurence="3">asparagus</element>

<element occurence="1">tomatoes;</element>

<element occurence="1">mushroom</element>

<element occurence="9">garlic</element>

<element occurence="2">beef</element>

<element occurence="1">pasta</element>

<element occurence="11">pepper</element>

<element occurence="1">mayonnaise</element>

<element occurence="3">noodles</element>

<element occurence="1">egg</element>

<element occurence="2">Chicken</element>

<element occurence="2">scallions</element>

<element occurence="7">chicken</element>

<element occurence="1">oil</element>

<element occurence="1">butter</element>

<element occurence="4">soy</element>

<element occurence="4">basil</element>

<element occurence="2">bean</element>

<element occurence="1">tomato paste</element>

<element occurence="3">spaghetti</element>

<element occurence="2">chili</element>

<element occurence="3">oregano</element>

<element occurence="2">bread</element>

<element occurence="2">cornstarch</element>
```

```
            <element occurence="1">Parmesan cheese</element>
            <element occurence="1">almonds</element>
            <element occurence="1">bamboo</element>
            <element occurence="3">sesame</element>
            <element occurence="2">sherry</element>
            <element occurence="2">peanut</element>
            <element occurence="1">rice vinegar</element>
            <element occurence="1">cheese</element>
            <element occurence="3">mushrooms</element>
            <element occurence="3">parsley</element>
            <element occurence="1">pasta</element>
            <element occurence="10">oil</element>
            <element occurence="2">broccoli</element>
            <element occurence="1">green pepper</element>
            <element occurence="1">mushroom soup</element>
            <element occurence="5">salt</element>
            <element occurence="1">beef</element>
            <element occurence="1">rice wine</element>
            <element occurence="1">Salt</element>
      </concept>
</xml>
```

Preparation time gazetteer

```
<xml version="Melita">

       <concept name="preparation_time">

              <element occurence="1">30 seconds</element>

       </concept>

</xml>
```

Table 8

Measure gazetteer

```
<xml version="Melita">

       <concept name="measure">

              <element occurence="2">pound</element>

              <element occurence="1">ounce can</element>

              <element occurence="5">teaspoon</element>

              <element occurence="1">cloves</element>

              <element occurence="1">can</element>

              <element occurence="1">cloves</element>

              <element occurence="2">teaspoons</element>

              <element occurence="2">can</element>

              <element occurence="1">bell</element>

              <element occurence="2">tablespoon</element>

              <element occurence="3">tablespoons</element>

              <element occurence="3">glass</element>

              <element occurence="3">cup</element>

              <element occurence="2">ounces</element>

              <element occurence="1">tspns</element>

              <element occurence="1">tspn</element>

              <element occurence="0">kilo</element>

              <element occurence="0">kilogram</element>

              <element occurence="0">gram</element>

              <element occurence="0">liter</element>

              <element occurence="0">deciliter</element>
```

```
            <element occurence="0">centiliter</element>

            <element occurence="0">mililiter</element>

      </concept>

</xml>
```

Regular measure gazetteer

```
<xml version="Melita">

      <concept name="regular_measure">

            <element occurence="2">pound</element>

            <element occurence="1">pounds</element>

            <element occurence="2">kg</element>

            <element occurence="0">g</element>

            <element occurence="0">gram</element>

            <element occurence="0">grams</element>

            <element occurence="0">kilogram</element>

            <element occurence="0">oz</element>

      </concept>

</xml>
```

If the attribute occurrence=0 means that this is a new concept introduced by the user in the gazetteer, and has not been found jet in any text.

Table 9

Non regular measure gazetteer

```
<xml version="Melita">

      <concept name="non_regular_measure">

            <element occurence="5">tablespoons</element>
```

```
            <element occurence="2">pound</element>
            <element occurence="1">cups</element>
            <element occurence="2">clove</element>
            <element occurence="1">cans</element>
            <element occurence="3">teaspoons</element>
            <element occurence="2">teaspoon</element>
            <element occurence="3">tablespoon</element>
            <element occurence="1">cloves</element>
            <element occurence="1">bell</element>
            <element occurence="1">pints</element>
            <element occurence="3">can</element>
            <element occurence="1">bunch</element>
            <element occurence="15">cup</element>
      </concept>
</xml>
```

Utensil gazetteer

```
<xml version="Melita">
      <concept name="utensil">
            <element occurence="2">pot</element>
            <element occurence="1">lid</element>
            <element occurence="1">frying pan</element>
            <element occurence="1">bowl</element>
            <element occurence="2">oven</element>
      </concept>
</xml>
```

Verb gazetteer

```
<xml version="Melita">

       <concept name="verb">

              <element occurence="2">Serve</element>

              <element occurence="1">Top</element>

              <element occurence="2">saute</element>

              <element occurence="1">rest</element>

              <element occurence="1">pour</element>

              <element occurence="2">Remove</element>

              <element occurence="4">add</element>

              <element occurence="2">toss</element>

              <element occurence="3">Heat</element>

              <element occurence="2">Cover</element>

              <element occurence="1">torn</element>

              <element occurence="3">Stir-fry</element>

              <element occurence="1">Re<verb>heat</element>

              <element occurence="3">Combine</element>

              <element occurence="2">stand</element>

              <element occurence="1">Sprinkle</element>

              <element occurence="1">Preheat</element>

              <element occurence="1">Cook</element>

              <element occurence="1">Cut</element>

              <element occurence="2">Place</element>

              <element occurence="1">Stir</element>
```

```
            <element occurence="2">drain</element>

            <element occurence="1">Coat</element>

            <element occurence="1">mixed</element>

            <element occurence="2">Drain</element>

            <element occurence="1">Bake</element>

            <element occurence="3">mix</element>

            <element occurence="1">Dice</element>

            <element occurence="1">heat</element>

            <element occurence="1">refrigerate</element>

            <element occurence="1">Make</element>

            <element occurence="1">season</element>

            <element occurence="8">Add</element>

            <element occurence="1">cook</element>

            <element occurence="1">combine</element>

            <element occurence="1">fry</element>

            <element occurence="2">simmer</element>

            <element occurence="1">stir</element>

            <element occurence="1">Quarter</element>

      </concept>

</xml>
```

Verb gazetteer

```
<xml version="Melita">

      <concept name="verb">

            <element occurence="1">blend</element>

            <element occurence="1">fry</element>
```

```
<element occurence="0">boil</element>

<element occurence="0">heat</element>

<element occurence="0">dress</element>

<element occurence="0">season</element>

<element occurence="0">cut</element>

<element occurence="0">chop</element>

<element occurence="0">slice</element>

<element occurence="0">cook</element>

<element occurence="0">steam</element>

<element occurence="0">roast</element>

<element occurence="0">grill</element>

<element occurence="0">knead</element>

<element occurence="0">mix</element>

<element occurence="0">prepare</element>

<element occurence="0">flavour</element>

<element occurence="0">flavor</element>

<element occurence="0">season</element>

<element occurence="0">spice</element>

<element occurence="0">grind</element>

<element occurence="0">crush</element>

<element occurence="0">triturate</element>

<element occurence="0">beat</element>

<element occurence="0">whisk</element>

<element occurence="0">whip</element>

<element occurence="0">cream</element>
```

```
            <element occurence="0">peel</element>

      </concept>

      </xml>
```

Cooking time gazetteer

```
<xml version="Melita">

      <concept name="cooking_time">

            <element occurence="1">5 minutes</element>

            <element occurence="1">1 minute</element>

            <element occurence="1">1 to 2 hours</element>

            <element occurence="1">20 seconds</element>

      </concept>

</xml>
```

Ingredient gazetteer

```
<xml version="Melita">

      <concept name="ingredient">

            <element occurence="2">bay</element>

            <element occurence="3">ginger</element>

            <element occurence="2">vinegar</element>

            <element occurence="4">tomato</element>

            <element occurence="1">chives</element>

            <element occurence="1">lettuce</element>

            <element occurence="8">onion</element>

            <element occurence="3">sugar</element>
```

```
<element occurence="1">seeds</element>

<element occurence="1">basil</element>

<element occurence="1">mozzarella cheese</element>

<element occurence="1">sesame oil</element>

<element occurence="1">sauce</element>

<element occurence="3">water</element>

<element occurence="1">oregano</element>

<element occurence="9">tomatoes</element>

<element occurence="1">mint</element>

<element occurence="1">almond</element>

<element occurence="1">onions</element>

<element occurence="3">asparagus</element>

<element occurence="1">porkchops</element>

<element occurence="1">tomatoes</element>

<element occurence="9">garlic</element>

<element occurence="1">mushroom</element>

<element occurence="2">beef</element>

<element occurence="1">pasta</element>

<element occurence="1">egg</element>

<element occurence="3">noodles</element>

<element occurence="1">mayonnaise</element>

<element occurence="11">pepper</element>

<element occurence="2">Chicken</element>

<element occurence="2">scallions</element>

<element occurence="1">oil;</element>
```

```
<element occurence="7">chicken</element>

<element occurence="1">butter</element>

<element occurence="2">bean</element>

<element occurence="4">basil</element>

<element occurence="4">soy</element>

<element occurence="3">spaghetti</element>

<element occurence="1">tomato paste</element>

<element occurence="3">oregano</element>

<element occurence="2">chili</element>

<element occurence="2">cornstarch</element>

<element occurence="2">bread</element>

<element occurence="1">almonds</element>

<element occurence="1">Parmesan cheese</element>

<element occurence="0">cheese</element>

<element occurence="2">peanut</element>

<element occurence="2">sherry</element>

<element occurence="3">sesame</element>

<element occurence="1">bamboo</element>

<element occurence="1">rice vinegar</element>

<element occurence="1">cheese</element>

<element occurence="3">parsley</element>

<element occurence="3">mushrooms</element>

<element occurence="10">oil</element>

<element occurence="1">pasta</element>

<element occurence="1">green pepper</element>
```

```
            <element occurence="2">broccoli</element>

            <element occurence="1">mushroom soup</element>

            <element occurence="1">beef</element>

            <element occurence="5">salt</element>

            <element occurence="1">Salt</element>

            <element occurence="1">rice wine</element>

      </concept>

</xml>
```

More than one gazetteer of the same concept are allowed. They can be made in different sessions but used together.

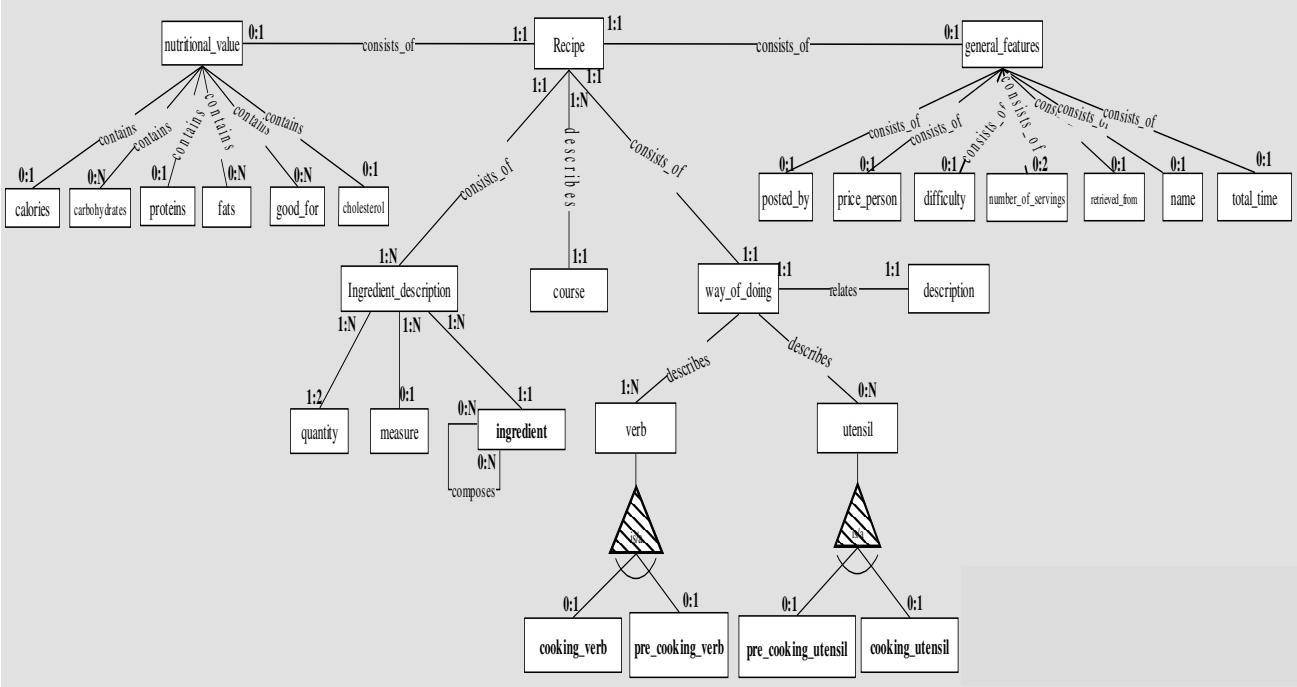
APPENDIX_13- Non-Attributed ER diagram

The models shown in the design chapter (the attributed ones) are simpler and clearer to model the recipes context. But unfortunately they are not compatible with the annotation tool, as it is explained in detail in.

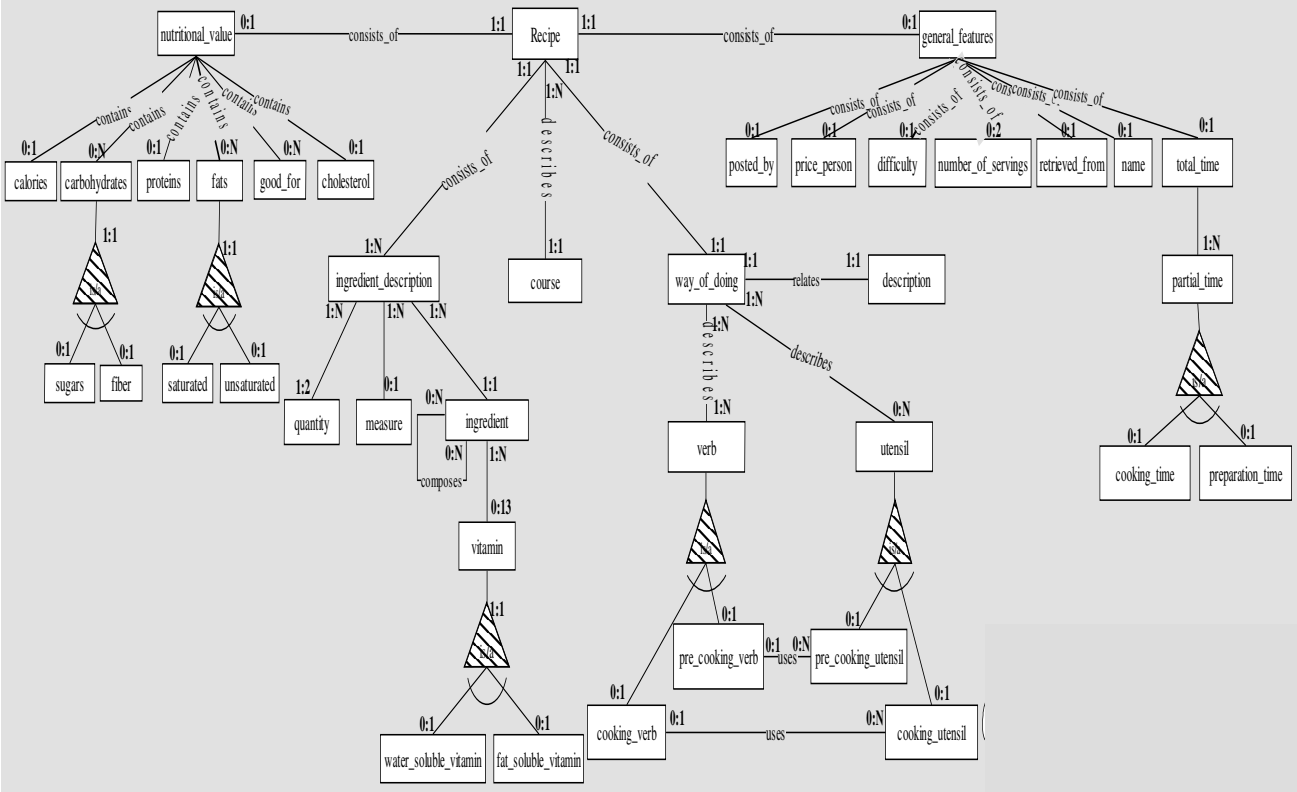
For compatibility and portability reasons the attributed ER diagram can not be used in the information extraction task. The annotation and IE tools do only pay attention to the entities in the Ontology; the knowledge stored in the attributes is lost.

After realizing of this inconvenience, many aspects of the original classification had to be remade, and all the important information had to be modeled as instances instead of attributes.

Next picture shows how the model looks like after this restructuring:



When modelling concepts as entities instead of attributes it is easy to improve the level of knowledge representation with IS-A relationships (some entities can be specialized by means of this relationship). Some new entities that inherit from the old ones have appeared. The final model looks like this:



I will briefly explain some of the performed changes:

Nutritionally value entity description

All its attributes has been transformed into subordinate entities, related to the *nutritional_value* entity trough the relationship *contains*. The cardinalities have been also set.

Ingredient_description entity description

The *measure* and the *quantity* attributes have been swapped to entities. Some knowledge has been added to the diagram, by means of cardinalities: An *ingredient_description* can contain 1 or 2 *quantities*, and 0 or 1 *measure*.

The recipe's ingredient description can be now modeled as following:

"250 grams of cheese":

ingredient_description= ingredient: cheese + quantity: 250 + measure: grams

"½ kg of tomatoes":

ingredient_description=ingredient: tomatoes + quantity: ½ + measure: kg

General features entity description

All its attributes have been transformed into entities as well. They are all related with cero or one cardinality, unless the *number_of_servings*, which can be null (not stated in the recipe),

one (as usual) or two values (many recipes are not accurate and give an approximation of the number of servings, for example: "yields 5 to 6 servings")

As all the features are now modeled as entities, I have taken the opportunity to improve the model. Some additional knowledge can be added. For example: the entity *time* can be detailed more. Can be modeled as an entity *total_time*; each recipe has cero or one *total_time*. A *total_time* is composed from one or many *partial_ times*. Finally each partial time IS-A *cooking_time* or a *preparation_time*. (these entities are a specialization of the entity *partial_time*, they inherit from it and specialize it)

Ingredient entity description

A new entity has been added to provide more knowledge to the model. This is the *vitamin* entity. It has been related to the *ingredient* entity, stating with the cardinalities that an *ingredient* can have from 0 up to 13 (the number of know vitamins) vitamins. One *vitamin* can be present in more than one *ingredient*.

The *vitamin* entity is specialized into the disjoint subclasses *water_soluble_vitamin* and *fat_soluble_vitamin*

Way of doing entity description

Verbs and *utensils* have been transformed to entities and related to *the way_of_doing* entity with the appropriate cardinalities. Here the model has been also enhanced by means of specializing and relating these entities as shown in the diagram.

APPENDIX_14– Initial Page Prototype

Recipes Intelligent Agent

General preferences

What are you interested in?

☐ Beverage ☑ Food

What kind of course are you interested in?

☐ Starter ☐ Appetizer ☐ Main course ☐ Dessert

Ingredient preferences

Which kind of ingredients are you looking for?

- Vegetables
- Meat
- Fish
- Fruit
- Eggs
- Dairy products
- (Other)

Which ingredients do you want to avoid?

- Vegetables
- Meat
- Fish
- Fruit
- Eggs
- Dairy products
- (Other)

I want to be more explicit:

I want to be more explicit:

COOKING preferences

Cooking utensils

I would like to use

- Frying pan
- Microwave
- Mixer
- Oven
- Pot
- All

But not use

- Frying pan
- Microwave
- Mixer
- Oven
- Pot
- None

Cooking method

I would like to

- Peel
- Cut
- Mix
- Chop
- Fry
- Boil
- Heat
- Cook
- Steam
- Other
- None

But not

- Peel
- Cut
- Mix
- Chop
- Fry
- Boil
- Heat
- Cook
- Steam
- Other
- None

nutritional preferences

I would like the following nutritional values ☐ Per serving ☐ Per 100 grams

kcalories	Carbohydrates	Fats	Cholesterol	Proteins
Less than 100	0-10	0	30-40	0-10

I need something good for:

You did not find wat you are looking for? Please, enter your wishes:

I want to look for:

I do not want any:

Search! Begin again

DTU(Denmark) **EPSIIG(Spain)**

Leticia Gutiérrez Villarías ***Master Thesis Project.***

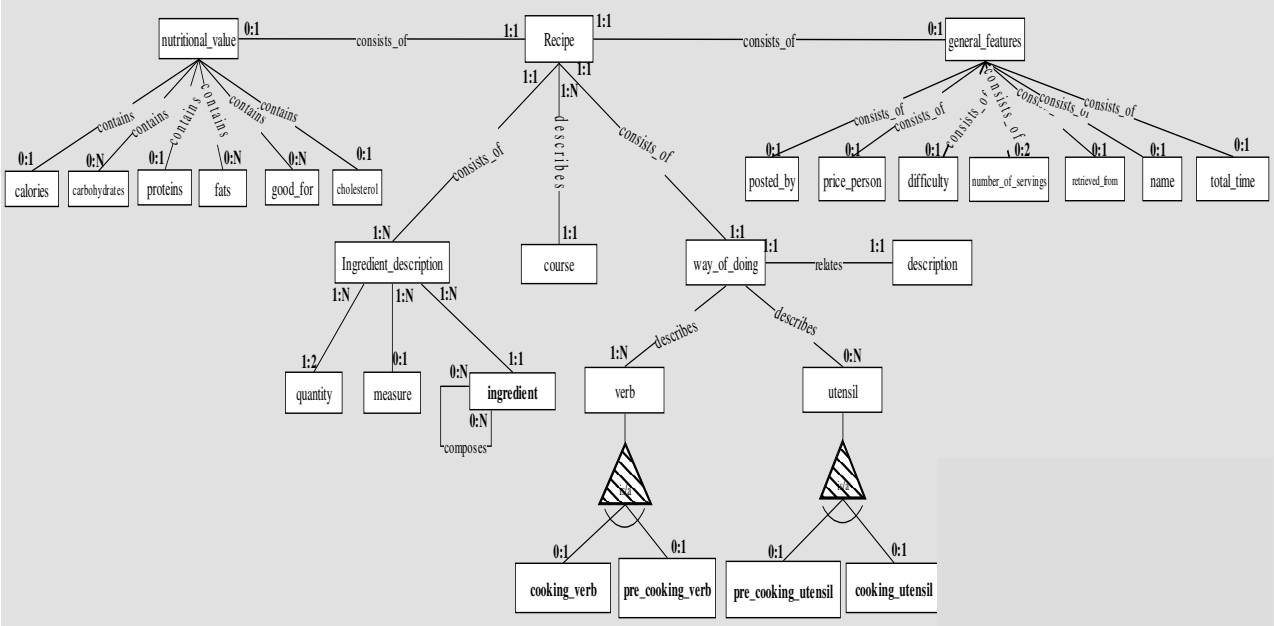
APPENDIX_15 – ER NON-ATTRIBUTED MODEL

The first attributed models designed, are the simpler and clearer ones to model the recipes context. But they are not compatible with the annotation tool.

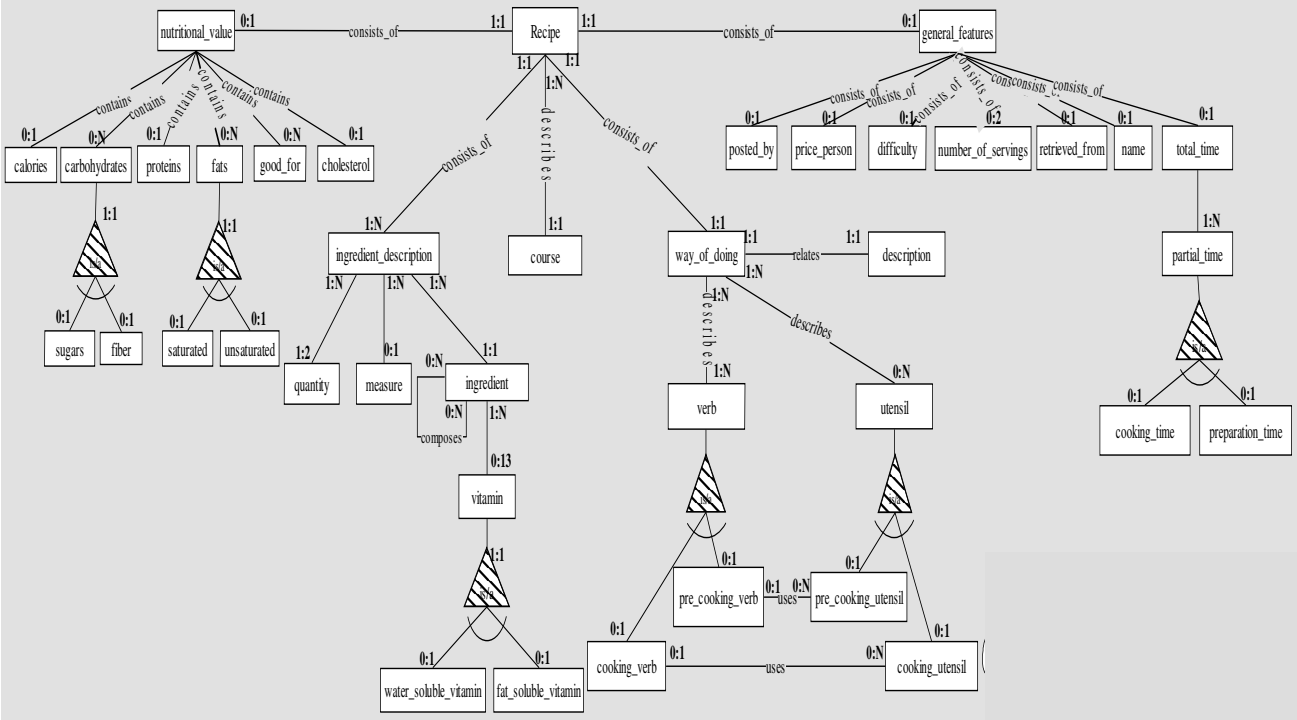
For compatibility and portability reasons the attributed ER diagram can not be used in the information extraction task. The annotation and IE tools do only pay attention to the entities in the Ontology; the knowledge stored in the attributes is lost.

After realizing of this inconvenience, many aspects of the original classification had to be remade, and all the important information had to be modeled as instances instead of attributes.

Next picture shows how the model looks like after this restructuring:



When modelling concepts as entities instead of attributes it is easy to improve the level of knowledge representation with IS-A relationships (some entities can be specialized by means of this relationship). Some new entities that inherit from the old ones have appeared. The final model looks like this:



I will briefly explain some of the performed changes:

Nutritionally value entity description

All its attributes has been transformed into subordinate entities, related to the *nutritional_value* entity trough the relationship *contains*. The cardinalities have been also set.

Ingredient_description entity description

The *measure* and the *quantity* attributes have been swapped to entities. Some knowledge has been added to the diagram, by means of cardinalities: An *ingredient_description* can contain 1 or 2 *quantities*, and 0 or 1 *measure*.

The recipe's ingredient description can be now modeled as following:

"250 grams of cheese":

ingredient_description= ingredient: cheese + quantity: 250 + measure: grams

"½ kg of tomatoes":

ingredient_description=ingredient: tomatoes + quantity: ½ + measure: kg

General features entity description

All its attributes have been transformed into entities as well. They are all related with cero or one cardinality, unless the *number_of_servings*, which can be null (not stated in the recipe), one (as usual) or two values (many recipes are not accurate and give an approximation of the

number of servings, for example: “yields 5 to 6 servings”)

As all the features are now modeled as entities, I have taken the opportunity to improve the model. Some additional knowledge can be added. For example: the entity *time* can be detailed more. Can be modeled as an entity *total_time*; each recipe has cero or one *total_time*. A *total_time* is composed from one or many *partial_ times*. Finally each partial time IS-A *cooking_time* or a *preparation_time*. (these entities are a specialization of the entity *partial_time*, they inherit from it and specialize it)

Ingredient entity description

A new entity has been added to provide more knowledge to the model. This is the *vitamin* entity. It has been related to the *ingredient* entity, stating with the cardinalities that an *ingredient* can have from 0 up to 13 (the number of know vitamins) vitamins. One *vitamin* can be present in more than one *ingredient*.

The *vitamin* entity is specialized into the disjoint subclasses *water_soluble_vitamin* and *fat_soluble_vitamin*

Way of doing entity description

Verbs and *utensils* have been transformed to entities and related to *the way_of_doing* entity with the appropriate cardinalities. Here the model has been also enhanced by means of specializing and relating these entities as shown in the diagram.

APPENDIX_16- Ontology skeleton in DAML

Ontology skeleton in DAML

```
<?xml version="1.0" encoding="ISO-8859-1"?>



<!DOCTYPE rdf:RDF

[<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'><!ENTITY xsd
'http://www.w3.org/2000/10/XMLSchema#'><!ENTITY dc
'http://purl.org/dc/elements/1.1/'><!ENTITY daml
'http://www.daml.org/2001/03/daml+oil#'><!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-
schema#'>]>



<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
xmlns="http://webode.dia.fi.upm.es/DAMLOIL/2004-03-10-recipes#">

  <!--Ontology description-->

  <daml:Ontology rdf:about="">

    <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil" />

    <daml:imports rdf:resource="http://www.w3.org/2000/10/XMLSchema" />

  </daml:Ontology>

  <daml:Class rdf:ID="animal_origin">

    <daml:subClassOf>
```

```
    <daml:Class rdf:about="#Ingredient" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Beer">
  <daml:subClassOf>
    <daml:Class rdf:about="#Drink" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Bread">
  <daml:subClassOf>
    <daml:Class rdf:about="#flour_ingredient" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Cereal_grain_potatoe">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Cheese">
  <daml:subClassOf>
    <daml:Class rdf:about="#Dairy_products" />
  </daml:subClassOf>
 </daml:Class>
```

```
<daml:Class rdf:ID="Coffee_tea_cocoa">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="cup">
  <daml:subClassOf>
    <daml:Class rdf:about="#non_regular_measure" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Dairy_products">
  <daml:subClassOf>
    <daml:Class rdf:about="#animal_origin" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Drink">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Egg">
  <daml:subClassOf>
    <daml:Class rdf:about="#Dairy_products" />
```

```
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Fat">
  <daml:subClassOf>
   <daml:Class rdf:about="#animal_origin" />
  </daml:subClassOf>
  <daml:subClassOf>
   <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Fish">
  <daml:subClassOf>
   <daml:Class rdf:about="#animal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="flour_ingredient">
  <daml:subClassOf>
   <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Fruit">
  <daml:subClassOf>
   <daml:Class rdf:about="#vegetal_origin" />
```

```
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Hard_drink">
  <daml:subClassOf>
   <daml:Class rdf:about="#Drink" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Ingredient">
  <daml:subClassOf>
   <daml:Class rdf:about="#Ingredient_description" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Ingredient_description" />
 <daml:Class rdf:ID="Ingredient_description_part">
  <daml:subClassOf>
   <daml:Class rdf:about="#Recipe" />
  </daml:subClassOf>
  <daml:subClassOf>
   <daml:Restriction>
        <daml:onProperty rdf:resource="#is_composed_of" />
        <daml:toClass rdf:resource="#Ingredient_description" />
   </daml:Restriction>
  </daml:subClassOf>
```

```
 </daml:Class>
 <daml:Class rdf:ID="kilo_based_measure">
  <daml:subClassOf>
   <daml:Class rdf:about="#regular_measure" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="liter_based_measure">
  <daml:subClassOf>
   <daml:Class rdf:about="#regular_measure" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Measure">
  <daml:subClassOf>
   <daml:Class rdf:about="#Ingredient_description" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Meat">
  <daml:subClassOf>
   <daml:Class rdf:about="#animal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Milk">
  <daml:subClassOf>
```

```
    <daml:Class rdf:about="#Dairy_products" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="non_regular_measure">
  <daml:subClassOf>
    <daml:Class rdf:about="#Measure" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Pasta">
  <daml:subClassOf>
    <daml:Class rdf:about="#flour_ingredient" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Poultry">
  <daml:subClassOf>
    <daml:Class rdf:about="#Meat" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Quantity">
  <daml:subClassOf>
    <daml:Class rdf:about="#Ingredient_description" />
  </daml:subClassOf>
 </daml:Class>
```

```
<daml:Class rdf:ID="Recipe" />
<daml:Class rdf:ID="Red_meat">
  <daml:subClassOf>
    <daml:Class rdf:about="#Meat" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="regular_measure">
  <daml:subClassOf>
    <daml:Class rdf:about="#Measure" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Rice">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Spice">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="spoon">
  <daml:subClassOf>
```

```
    <daml:Class rdf:about="#non_regular_measure" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="teaspoon">
  <daml:subClassOf>
    <daml:Class rdf:about="#non_regular_measure" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="Utensil" />
 <daml:Class rdf:ID="Vegetable">
  <daml:subClassOf>
    <daml:Class rdf:about="#vegetal_origin" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="vegetal_origin">
  <daml:subClassOf>
    <daml:Class rdf:about="#Ingredient" />
  </daml:subClassOf>
 </daml:Class>
 <daml:Class rdf:ID="way_of_doing_part">
  <daml:subClassOf>
    <daml:Class rdf:about="#Recipe" />
  </daml:subClassOf>
```

```
  <daml:subClassOf>
    <daml:Restriction>
          <daml:onProperty rdf:resource="#are_used_to_cook" />
          <daml:toClass rdf:resource="#Utensil" />
    </daml:Restriction>
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Wine">
  <daml:subClassOf>
    <daml:Class rdf:about="#Drink" />
  </daml:subClassOf>
</daml:Class>
<daml:Class rdf:ID="Yogurt">
  <daml:subClassOf>
    <daml:Class rdf:about="#Dairy_products" />
  </daml:subClassOf>
</daml:Class>
<daml:ObjectProperty rdf:ID="is_composed_of">
  <rdfs:domain rdf:resource="#Ingredient_description_part" />
  <rdfs:range rdf:resource="#Ingredient_description" />
</daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="are_used_to_cook">
  <rdfs:domain rdf:resource="#way_of_doing_part" />
```

```
  <rdfs:range rdf:resource="#Utensil" />
 </daml:ObjectProperty>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Ingredient_description_part" />
  <daml:Class rdf:resource="#way_of_doing_part" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Ingredient" />
  <daml:Class rdf:resource="#Quantity" />
  <daml:Class rdf:resource="#Measure" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#non_regular_measure" />
  <daml:Class rdf:resource="#regular_measure" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#kilo_based_measure" />
  <daml:Class rdf:resource="#liter_based_measure" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#cup" />
  <daml:Class rdf:resource="#spoon" />
  <daml:Class rdf:resource="#teaspoon" />
```

```
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Red_meat" />
  <daml:Class rdf:resource="#Poultry" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Cheese" />
  <daml:Class rdf:resource="#Milk" />
  <daml:Class rdf:resource="#Yogurt" />
  <daml:Class rdf:resource="#Egg" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#vegetal_origin" />
  <daml:Class rdf:resource="#animal_origin" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Fish" />
  <daml:Class rdf:resource="#Meat" />
  <daml:Class rdf:resource="#Dairy_products" />
  <daml:Class rdf:resource="#Fat" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Pasta" />
```

```
  <daml:Class rdf:resource="#Bread" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#flour_ingredient" />
  <daml:Class rdf:resource="#Fruit" />
  <daml:Class rdf:resource="#Rice" />
  <daml:Class rdf:resource="#Spice" />
  <daml:Class rdf:resource="#Vegetable" />
  <daml:Class rdf:resource="#Fat" />
  <daml:Class rdf:resource="#Drink" />
  <daml:Class rdf:resource="#Cereal_grain_potatoe" />
  <daml:Class rdf:resource="#Coffee_tea_cocoa" />
 </daml:Disjoint>
 <daml:Disjoint rdf:parseType="daml:collection">
  <daml:Class rdf:resource="#Hard_drink" />
  <daml:Class rdf:resource="#Wine" />
  <daml:Class rdf:resource="#Beer" />
 </daml:Disjoint>
 <daml:Class rdf:about="#Recipe" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
   <daml:Class rdf:resource="#Ingredient_description_part" />
   <daml:Class rdf:resource="#way_of_doing_part" />
  </daml:disjointUnionOf>
```

```
 </daml:Class>
 <daml:Class rdf:about="#Measure" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
   <daml:Class rdf:resource="#non_regular_measure" />
   <daml:Class rdf:resource="#regular_measure" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#Meat" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
   <daml:Class rdf:resource="#Red_meat" />
   <daml:Class rdf:resource="#Poultry" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#Dairy_products" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
   <daml:Class rdf:resource="#Cheese" />
   <daml:Class rdf:resource="#Milk" />
   <daml:Class rdf:resource="#Yogurt" />
   <daml:Class rdf:resource="#Egg" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#Ingredient" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
```

```
    <daml:Class rdf:resource="#vegetal_origin" />
    <daml:Class rdf:resource="#animal_origin" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#animal_origin" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
    <daml:Class rdf:resource="#Fish" />
    <daml:Class rdf:resource="#Meat" />
    <daml:Class rdf:resource="#Dairy_products" />
    <daml:Class rdf:resource="#Fat" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#flour_ingredient" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
    <daml:Class rdf:resource="#Pasta" />
    <daml:Class rdf:resource="#Bread" />
  </daml:disjointUnionOf>
 </daml:Class>
 <daml:Class rdf:about="#vegetal_origin" rdf:parseType="daml:collection">
  <daml:disjointUnionOf>
    <daml:Class rdf:resource="#flour_ingredient" />
    <daml:Class rdf:resource="#Fruit" />
    <daml:Class rdf:resource="#Rice" />
```

```
    <daml:Class rdf:resource="#Spice" />
    <daml:Class rdf:resource="#Vegetable" />
    <daml:Class rdf:resource="#Fat" />
    <daml:Class rdf:resource="#Drink" />
    <daml:Class rdf:resource="#Cereal_grain_potatoe" />
    <daml:Class rdf:resource="#Coffee_tea_cocoa" />
   </daml:disjointUnionOf>
  </daml:Class>
  <daml:Class rdf:about="#Drink" rdf:parseType="daml:collection">
   <daml:disjointUnionOf>
    <daml:Class rdf:resource="#Hard_drink" />
    <daml:Class rdf:resource="#Wine" />
    <daml:Class rdf:resource="#Beer" />
   </daml:disjointUnionOf>
  </daml:Class>
</rdf:RDF>
```

APPENDIX_17 Complete Ontology Description

17.1 Term Glossary

Name	Type
dairy_product : animal_origin	Class Attribute
fat_soluble : advices	Class Attribute
fat_soluble : description	Class Attribute
fish : animal_origin	Class Attribute
flavoring : animal_origin	Class Attribute
milk : fat_content	Class Attribute
vegetable : anima_origin	Class Attribute
water_soluble : advices	Class Attribute
water_soluble : description	Class Attribute
A	Concept
B1	Concept
B12	Concept
B2	Concept
B3	Concept
B5	Concept
B6	Concept
C	Concept
D	Concept
E	Concept
Folic_acid	Concept
H	Concept
K	Concept
baking_supply	Concept
beer	Concept
bread	Concept
butter	Concept
butter_substitute	Concept
cacao	Concept
carbohydrate	Concept
carbohydrate_description	Concept
caviar_roe	Concept
cereal	Concept
cereal_grain	Concept
cheese	Concept
cheese_substitute	Concept
chocolate	Concept
cholesterol	Concept

cocktail	Concept
coffee	Concept
common_vegetable	Concept
condiment	Concept
cooking_instructions	Concept
cooking_time	Concept
cooking_utensil	Concept
cooking_verb	Concept
course	Concept
crab	Concept
cream	Concept
cream_substitute	Concept
cured_precooked_meat	Concept
dairy_product	Concept
dairy_product_substitute	Concept
dark_chocolate	Concept
dessert	Concept
difficulty	Concept
drink	Concept
egg	Concept
energy	Concept
fat_oil	Concept
fat_soluble	Concept
fats	Concept
fats_description	Concept
fatty fish	Concept
fatty_fish_course	Concept
fiber	Concept
fish	Concept
fish_course	Concept
flavoring	Concept
food	Concept
food_quantity	Concept
fresh_fruit	Concept
fruit	Concept
fruit_juice	Concept
general_features	Concept
general_features_part	Concept
good_for	Concept
grain	Concept
hard_drink	Concept
herb	Concept
honey	Concept

infusion	Concept
ingredient	Concept
ingredient_description	Concept
ingredient_description_part	Concept
jam	Concept
juice	Concept
lean fish	Concept
lean_fish_course	Concept
leaven	Concept
legume	Concept
mammal_meat	Concept
margarine	Concept
measure	Concept
meat	Concept
meat_course	Concept
milk	Concept
milk_chocolate	Concept
milk_shake	Concept
milk_substitute	Concept
monounsaturated	Concept
non_regular_measure	Concept
non_sweet_dessert	Concept
number_of_portions	Concept
number_of_servings	Concept
nut	Concept
nutritional_value_100_grams	Concept
nutritional_value_100_grams_part	Concept
oil	Concept
pasta	Concept
pasta_bread	Concept
pasta_course	Concept
pasta_with_cream	Concept
pasta_with_non_cream_white_sauce	Concept
pasta_with_red_sauce	Concept
pasta_with_white_sauce	Concept
percentage_DV	Concept
polyunsaturated	Concept
posted_by	Concept
poultry_meat	Concept
pre_cooking_utensil	Concept
preparation_time	Concept
preparation_verb	Concept
price_person	Concept

proteins	Concept
quantity	Concept
recipe	Concept
regular_measure	Concept
regular_volume_measure	Concept
regular_weight_measure	Concept
reptile_meat	Concept
retrieved_from	Concept
rice_course	Concept
root_tuber	Concept
salt	Concept
saturated	Concept
sauce	Concept
sea_vegetable	Concept
seafood_course	Concept
shellfish	Concept
smoked_dry_fish	Concept
soda	Concept
sodium	Concept
soup	Concept
soy	Concept
spice	Concept
stimulant	Concept
sugar	Concept
sweet_dessert	Concept
sweetener	Concept
syrup	Concept
tea	Concept
time	Concept
tomato	Concept
tomato sauce	Concept
total_carbohydrate	Concept
unsaturated	Concept
utensil	Concept
vegetable	Concept
vegetable_juice	Concept
vegetal_origin_ingredient	Concept
vegetarian_course	Concept
verb	Concept
vinegar	Concept
vitamin	Concept
vitamins	Concept
water	Concept

water_soluble	Concept
way_of_doing_part	Concept
wheat	Concept
wine	Concept
yeast	Concept
yogurt	Concept
yogurt_substitute	Concept
kind_of_carbohydrate	Group
kind_of_fat	Group
kind_of_fat_soluble_vitamin	Group
kind_of_food_quantity	Group
kind_of_measure	Group
kind_of_non_regular_measure	Group
kind_of_regular_measure	Group
kind_of_time	Group
kind_of_unsaturated_fat	Group
kind_of_utensil	Group
kind_of_verb	Group
kind_of_vitamin	Group
kind_of_water_soluble_vitamin	Group
recipes_parts	Group
animal_origin	Instance Attribute
measure_attribute	Instance Attribute
name	Instance Attribute
quantity_attribute	Instance Attribute
recommendedDV	Instance Attribute
baked_spinach_with_cheese	Instance Set
consists_of*(ingredient_description, ingredient)*	Relation
consists_of*(ingredient_description, measure)*	Relation
consists_of*(ingredient_description, quantity)*	Relation
consists_of*(ingredient_description_part, ingredient_description)*	Relation
consists_of*(way_of_doing_part, difficulty)*	Relation
consists_of*(way_of_doing_part, cooking_instructions)*	Relation
consists_of*(nutritional_value_100_grams_part, cholesterol)*	Relation
consists_of*(nutritional_value_100_grams_part, sodium)*	Relation
consists_of*(nutritional_value_100_grams_part, proteins)*	Relation
consists_of*(nutritional_value_100_grams_part, energy)*	Relation
consists_of*(nutritional_value_100_grams_part, vitamins)*	Relation
consists_of*(nutritional_value_100_grams_part, fats)*	Relation
consists_of*(nutritional_value_100_grams_part, carbohydrate_description)*	Relation
consists_of*(nutritional_value_100_grams_part, fats_description)*	Relation
consists_of*(general_features_part, time)*	Relation
consists_of*(general_features_part, difficulty)*	Relation

consists_of*(general_features_part, name)*	Relation
consists_of*(general_features_part, food_quantity)*	Relation
consists_of*(general_features_part, good_for)*	Relation
consists_of*(general_features_part, posted_by)*	Relation
consists_of*(general_features_part, retrieved_from)*	Relation
consists_of*(general_features_part, price_person)*	Relation
consists_of*(vitamins, vitamin)*	Relation
consists_of*(vitamins, percentage_DV)*	Relation
consists_of*(fats, saturated)*	Relation
consists_of*(fats, polyunsaturated)*	Relation
consists_of*(fats, monounsaturated)*	Relation
consists_of*(carbohydrate_description, fiber)*	Relation
consists_of*(carbohydrate_description, sugar)*	Relation
consists_of*(fats_description, fats)*	Relation
consists_of*(total_carbohydrate, carbohydrate)*	Relation
consists_of*(recipe, ingredient_description_part)*	Relation
consists_of*(recipe, way_of_doing_part)*	Relation
consists_of*(recipe, general_features)*	Relation
consists_of*(recipe, nutritional_value_100_grams)*	Relation
consists_of*(ingredient_description, ingredient)*	Relation
consists_of*(ingredient_description, measure)*	Relation
consists_of*(ingredient_description, quantity)*	Relation
consists_of*(ingredient_description_part, ingredient_description)*	Relation
consists_of*(way_of_doing_part, difficulty)*	Relation
And much more relations removed due to space limitations	

17.2 Concept Dictionary of a specific recipe

Concept name	Synonyms	Instances	Class attributes	Instance attributes	Relations
calories	--	45	--	--	
carbohydrates		7.4g			
cholesterol		0mg			
difficulty	complexity hardness	Easy	--		--
fats		0.4g			
good_for	health properties	General well-being Hear diseases	--		--
non_regular_measure	--	dash egg inch	--	--	--
number_of_servings	--	4	--	--	--
oil	--	Olive oil	--	Animal_origin	--

posted_by	creator published by	Marie da Silva	--	--	--
Price_person	cost per person price per person course's price average per person	2.5 €	--	--	--
proteins	--	15.5 g	--	--	--
quantity	amount	1 8	--	--	--
recipe	--	Baked Spinach With Cheese	--	--	consists_of describes
Regular_measure	--	Liter gram	--	--	--
salt	--	salt	--	--	--
shellfish	--	Shrimp	--	Animal_origin	--
spice	--	nutmeg paprika pepper	--	Animal_origin	--
time	--	1 hour	--	--	--
utensil	--	Wooden spoon Fork Pot Fry pan	--	--	--
vegetable	veggie	Garlic onion spinach		Animal_origin	--
vegetarian_course	--	Baked spinach with cheese	--	--	--
Verb		Chop Boil Cut Dry Melt Serve	--	--	--
way_of_doing_part	--	Boil the spinach and put aside. Combine them with the garlic, and cheese. Fry the shrimps [...] serve hot	--	--	are_used_to_cook consists_of

This is an example of an instance of a recipe in the database. (Be aware that animal_origin is a Boolean attribute that shows if the ingredient has animal or vegetal origin)

17.3 Binary Relation Table of the ontology recipes

Relation name	Source concept	Source cardinality (Max)	Target concept
consists_of	ingredient_description	n	ingredient
consists_of	ingredient_description	1	measure
consists_of	ingredient_description	2	quantity
consists_of	ingredient_description_part	n	ingredient_description
consits_of	ingredient_description_part	n	ingredient_description
are_used_to_cook	way_of_doing_part	n	utensil
consists_of	way_of_doing_part	1	difficulty
consists_of	way_of_doing_part	n	cooking_instructions
consists_of	nutritional_value_100_grams_part	1	cholesterol
consists_of	nutritional_value_100_grams_part	1	sodium
consists_of	nutritional_value_100_grams_part	1	proteins
consists_of	nutritional_value_100_grams_part	1	energy
consists_of	nutritional_value_100_grams_part	1	vitamins
consists_of	nutritional_value_100_grams_part	1	fats
consists_of	nutritional_value_100_grams_part	1	carbohydrate_description
consists_of	nutritional_value_100_grams_part	1	fats_description
consists_of	general_features_part	n	time
consists_of	general_features_part	1	difficulty
consists_of	general_features_part	1	name
consists_of	general_features_part	2	food_quantity
consists_of	general_features_part	n	good_for
consists_of	general_features_part	n	posted_by
consists_of	general_features_part	1	retrieved_from
consists_of	general_features_part	1	price_person
consists_of	vitamins	n	vitamin
consists_of	vitamins	1	percentage_DV
consists_of	fats	1	saturated
consists_of	fats	1	polyunsaturated
consists_of	fats	1	monounsaturated
consists_of	carbohydrate_description	1	fiber
consists_of	carbohydrate_description	1	sugar
found_in	vitamin	n	ingredient
contains	cooking_instructions	n	ingredient
contains	cooking_instructions	1	utensil
contains	cooking_instructions	1	verb
consists_of	fats_description	n	fats
consists_of	total_carbohydrate	n	carbohydrate
consists_of	recipe	1	ingredient_description_part
consists_of	recipe	1	way_of_doing_part

describes	recipe	1	course
consists_of	recipe	1	general_features
consists_of	recipe	1	nutritional_value_100_grams
is_found_in	A	n	egg

WebODE 2.0

Top

APPENDIX_18 Complete Ontology Description

```
<rdf:RDF xmlns="http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns:daml="http://www.daml.org/2001/03/daml+oil#">
   <rdf:Property rdf:ID="COURSE">
      <rdfs:domain rdf:resource="#MEAL"/>
      <rdfs:range rdf:resource="#MEAL-COURSE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="GRAPE-SLOT">
      <rdfs:range rdf:resource="#WINE-GRAPE"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="FOOD">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#EDIBLE-THING"/>
      <rdfs:domain rdf:resource="#MEAL-COURSE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="SUGAR">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#WINE-SUGAR"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="BODY">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:domain rdf:resource="#WINE"/>
      <rdfs:range rdf:resource="#WINE-BODY"/>
   </rdf:Property>
   <rdf:Property rdf:ID="COLOR">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#WINE-COLOR"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="DRINK">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:domain rdf:resource="#MEAL-COURSE"/>
      <rdfs:range rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="REGION">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#WINE-REGION"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="FLAVOR">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#WINE-FLAVOR"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>
   <rdf:Property rdf:ID="MAKER">
      <rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
      <rdfs:range rdf:resource="#WINERY"/>
      <rdfs:domain rdf:resource="#WINE"/>
   </rdf:Property>

   <daml:Restriction rdf:ID="WHITE-COLOR-RESTRICTION">
      <daml:onProperty rdf:resource="#COLOR"/>
      <daml:hasValue rdf:resource="#WHITE"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="RED-COLOR-RESTRICTION">
      <daml:onProperty rdf:resource="#COLOR"/>
      <daml:hasValue rdf:resource="#RED"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="ROSE-COLOR-RESTRICTION">
      <daml:onProperty rdf:resource="#COLOR"/>
      <daml:hasValue rdf:resource="#ROSE"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SWEET-SUGAR-RESTRICTION">
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:hasValue rdf:resource="#SWEET"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="OFF-DRY-SUGAR-RESTRICTION">
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:hasValue rdf:resource="#OFF-DRY"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRY-SUGAR-RESTRICTION">
```

```
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:hasValue rdf:resource="#DRY"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SWEET-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:toClass>
      <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#OFF-DRY"/>
               <rdf:Description rdf:about="#SWEET"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRY-OR-OFF-DRY-SUGAR-HAS-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:hasClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#DRY"/>
               <rdf:Description rdf:about="#OFF-DRY"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:hasClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRY-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#SUGAR"/>
      <daml:toClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#DRY"/>
               <rdf:Description rdf:about="#OFF-DRY"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="LIGHT-BODY-RESTRICTION">
      <daml:onProperty rdf:resource="#BODY"/>
      <daml:hasValue rdf:resource="#LIGHT"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="MEDIUM-BODY-RESTRICTION">
      <daml:onProperty rdf:resource="#BODY"/>
      <daml:hasValue rdf:resource="#MEDIUM"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="FULL-BODY-RESTRICTION">
      <daml:onProperty rdf:resource="#BODY"/>
      <daml:hasValue rdf:resource="#FULL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="LIGHT-OR-MEDIUM-BODY-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#BODY"/>
      <daml:toClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#LIGHT"/>
               <rdf:Description rdf:about="#MEDIUM"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#BODY"/>
      <daml:toClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#FULL"/>
               <rdf:Description rdf:about="#MEDIUM"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DELICATE-OR-MODERATE-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#FLAVOR"/>
      <daml:toClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#MODERATE"/>
               <rdf:Description rdf:about="#DELICATE"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
```

```
   <daml:Restriction rdf:ID="DELICATE-FLAVOR-RESTRICTION">
      <daml:onProperty rdf:resource="#FLAVOR"/>
      <daml:hasValue rdf:resource="#DELICATE"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="STRONG-FLAVOR-RESTRICTION">
      <daml:onProperty rdf:resource="#FLAVOR"/>
      <daml:hasValue rdf:resource="#STRONG"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="MODERATE-FLAVOR-RESTRICTION">
      <daml:onProperty rdf:resource="#FLAVOR"/>
      <daml:hasValue rdf:resource="#MODERATE"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#FLAVOR"/>
      <daml:toClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
                  <rdf:Description rdf:about="#MODERATE"/>
                  <rdf:Description rdf:about="#STRONG"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:maxCardinality>
            1
      </daml:maxCardinality>
   </daml:Restriction>
   <daml:Restriction rdf:ID="GAMAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#GAMAY-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="MERLOT-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#MERLOT-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="PINOT-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#PINOT-BLANC-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="PINOT-NOIR-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#PINOT-NOIR-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SEMILLON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#SEMILLON-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#SAUVIGNON-BLANC-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="CHARDONNAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#CHARDONNAY-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="CHENIN-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#CHENIN-BLANC-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="CABERNET-SAUVIGNON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasValue rdf:resource="#CABERNET-SAUVIGNON-INDIVIDUAL"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SEMILLON-INDIVIDUAL-OR-SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-HAS-CLASS-
RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:hasClass>
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#SEMILLON-INDIVIDUAL"/>
               <rdf:Description rdf:about="#SAUVIGNON-BLANC-INDIVIDUAL"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:hasClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="SEMILLON-INDIVIDUAL-OR-SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-TO-CLASS-
RESTRICTION">
      <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
      <daml:toClass>
```

```
         <rdfs:Class>
            <daml:oneOf rdf:parseType="daml:collection">
               <rdf:Description rdf:about="#SEMILLON-INDIVIDUAL"/>
               <rdf:Description rdf:about="#SAUVIGNON-BLANC-INDIVIDUAL"/>
            </daml:oneOf>
         </rdfs:Class>
      </daml:toClass>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-STRONG-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#STRONG-FLAVOR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#MODERATE-FLAVOR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-DELICATE-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#DRY-SUGAR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#OFF-DRY-SUGAR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-SWEET-SUGAR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#SWEET-SUGAR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#WHITE-COLOR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#RED-COLOR-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-LIGHT-BODY-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#LIGHT-BODY-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#MEDIUM-BODY-RESTRICTION"/>
   </daml:Restriction>
   <daml:Restriction rdf:ID="DRINK-HAS-FULL-BODY-TO-CLASS-RESTRICTION">
      <daml:onProperty rdf:resource="#DRINK"/>
      <daml:toClass rdf:resource="#FULL-BODY-RESTRICTION"/>
   </daml:Restriction>

   <rdfs:Class rdf:ID="ZINFANDEL">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
            <daml:hasValue rdf:resource="#ZINFANDEL-INDIVIDUAL"/>
         </daml:Restriction>
         <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="WINERY">
   </rdfs:Class>
   <rdfs:Class rdf:ID="WINE-SUGAR">
      <rdfs:subClassOf rdf:resource="#WINE-PROPERTY"/>
      <daml:oneOf rdf:parseType="daml:collection">
         <rdf:Description rdf:about="#SWEET"/>
         <rdf:Description rdf:about="#OFF-DRY"/>
         <rdf:Description rdf:about="#DRY"/>
      </daml:oneOf>
   </rdfs:Class>
```

```
<rdfs:Class rdf:ID="WINE-REGION">
</rdfs:Class>
<rdfs:Class rdf:ID="WINE-PROPERTY">
</rdfs:Class>
<rdfs:Class rdf:ID="WINE-GRAPE">
   <rdfs:subClassOf rdf:resource="#GRAPE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="WINE-FLAVOR">
   <rdfs:subClassOf rdf:resource="#WINE-PROPERTY"/>
   <daml:oneOf rdf:parseType="daml:collection">
      <rdf:Description rdf:about="#DELICATE"/>
      <rdf:Description rdf:about="#MODERATE"/>
      <rdf:Description rdf:about="#STRONG"/>
   </daml:oneOf>
</rdfs:Class>
<rdfs:Class rdf:ID="WINE-COLOR">
   <rdfs:subClassOf rdf:resource="#WINE-PROPERTY"/>
   <daml:oneOf rdf:parseType="daml:collection">
      <rdf:Description rdf:about="#WHITE"/>
      <rdf:Description rdf:about="#ROSE"/>
      <rdf:Description rdf:about="#RED"/>
   </daml:oneOf>
</rdfs:Class>
<rdfs:Class rdf:ID="WINE-BODY">
   <rdfs:subClassOf rdf:resource="#WINE-PROPERTY"/>
   <daml:oneOf rdf:parseType="daml:collection">
      <rdf:Description rdf:about="#LIGHT"/>
      <rdf:Description rdf:about="#MEDIUM"/>
      <rdf:Description rdf:about="#FULL"/>
   </daml:oneOf>
</rdfs:Class>
<rdfs:Class rdf:ID="WINE">
   <rdfs:subClassOf rdf:resource="#POTABLE-LIQUID"/>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#MAKER"/>
         <daml:cardinality>
            1
         </daml:cardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#MAKER"/>
         <daml:toClass rdf:resource="#WINERY"/>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
         <daml:minCardinality>
            1
         </daml:minCardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
         <daml:toClass rdf:resource="#WINE-GRAPE"/>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#REGION"/>
         <daml:cardinality>
            1
         </daml:cardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#REGION"/>
         <daml:toClass rdf:resource="#WINE-REGION"/>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#SUGAR"/>
         <daml:cardinality>
            1
         </daml:cardinality>
```

```
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#SUGAR"/>
               <daml:toClass rdf:resource="#WINE-SUGAR"/>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#FLAVOR"/>
               <daml:cardinality>
                  1
               </daml:cardinality>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#FLAVOR"/>
               <daml:toClass rdf:resource="#WINE-FLAVOR"/>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#BODY"/>
               <daml:cardinality>
                  1
               </daml:cardinality>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#BODY"/>
               <daml:toClass rdf:resource="#WINE-BODY"/>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#COLOR"/>
               <daml:cardinality>
                  1
               </daml:cardinality>
            </daml:Restriction>
         </rdfs:subClassOf>
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#COLOR"/>
               <daml:toClass rdf:resource="#WINE-COLOR"/>
            </daml:Restriction>
         </rdfs:subClassOf>
      </rdfs:Class>
      <rdfs:Class rdf:ID="WHITE-WINE">
         <daml:intersectionOf rdf:parseType="daml:collection">
            <daml:Restriction rdf:about="#WHITE-COLOR-RESTRICTION"/>
            <rdfs:Class rdf:about="#WINE"/>
         </daml:intersectionOf>
      </rdfs:Class>
      <rdfs:Class rdf:ID="WHITE-TABLE-WINE">
         <daml:intersectionOf rdf:parseType="daml:collection">
            <daml:Restriction rdf:about="#WHITE-COLOR-RESTRICTION"/>
            <rdfs:Class rdf:about="#TABLE-WINE"/>
         </daml:intersectionOf>
      </rdfs:Class>
      <rdfs:Class rdf:ID="WHITE-NON-SWEET-WINE">
         <daml:intersectionOf rdf:parseType="daml:collection">
            <daml:Restriction rdf:about="#DRY-OR-OFF-DRY-SUGAR-HAS-CLASS-RESTRICTION"/>
            <rdfs:Class rdf:about="#WHITE-WINE"/>
         </daml:intersectionOf>
         <rdfs:subClassOf rdf:resource="#DRY-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
      </rdfs:Class>
      <rdfs:Class rdf:ID="WHITE-LOIRE">
         <rdfs:subClassOf>
            <daml:Restriction>
               <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
               <daml:toClass>
                  <rdfs:Class>
                     <daml:oneOf rdf:parseType="daml:collection">
                        <rdf:Description rdf:about="#CHENIN-BLANC-INDIVIDUAL"/>
                        <rdf:Description rdf:about="#PINOT-BLANC-INDIVIDUAL"/>
                        <rdf:Description rdf:about="#SAUVIGNON-BLANC-INDIVIDUAL"/>
                     </daml:oneOf>
```

```
            </rdfs:Class>
         </daml:toClass>
      </daml:Restriction>
   </rdfs:subClassOf>
   <daml:intersectionOf rdf:parseType="daml:collection">
      <rdfs:Class rdf:about="#LOIRE"/>
      <rdfs:Class rdf:about="#WHITE-WINE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="WHITE-BURGUNDY">
   <rdfs:subClassOf rdf:resource="#CHARDONNAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
   <daml:intersectionOf rdf:parseType="daml:collection">
      <rdfs:Class rdf:about="#BURGUNDY"/>
      <rdfs:Class rdf:about="#WHITE-WINE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="WHITE-BORDEAUX">
   <rdfs:subClassOf rdf:resource="#SEMILLON-INDIVIDUAL-OR-SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-TO-
CLASS-RESTRICTION"/>
   <daml:intersectionOf rdf:parseType="daml:collection">
      <rdfs:Class rdf:about="#BORDEAUX"/>
      <rdfs:Class rdf:about="#WHITE-WINE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="US-REGION">
   <rdfs:subClassOf rdf:resource="#WINE-REGION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="TOURS">
   <rdfs:subClassOf rdf:resource="#CHENIN-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#REGION"/>
         <daml:hasValue rdf:resource="#TOURS-INDIVIDUAL"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#LOIRE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="TABLE-WINE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction rdf:about="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:Class rdf:about="#WINE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="SWEET-WINE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction rdf:about="#SWEET-SUGAR-RESTRICTION"/>
      <rdfs:Class rdf:about="#WINE"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="SWEET-RIESLING">
   <rdfs:subClassOf rdf:resource="#DESSERT-WINE"/>
   <rdfs:subClassOf rdf:resource="#FULL-BODY-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction rdf:about="#SWEET-SUGAR-RESTRICTION"/>
      <rdfs:Class rdf:about="#RIESLING"/>
   </daml:intersectionOf>
</rdfs:Class>
<rdfs:Class rdf:ID="SWEET-FRUIT-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#SWEET-FRUIT"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-SWEET-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="SWEET-FRUIT">
   <rdfs:subClassOf rdf:resource="#FRUIT"/>
</rdfs:Class>
<rdfs:Class rdf:ID="SWEET-DESSERT-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#SWEET-DESSERT"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
```

```
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SWEET-DESSERT">
      <rdfs:subClassOf rdf:resource="#DESSERT"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="ST-EMILION">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#STRONG-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#CABERNET-SAUVIGNON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#ST-EMILION-INDIVIDUAL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#BORDEAUX"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SPICY-RED-MEAT-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#SPICY-RED-MEAT"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SPICY-RED-MEAT">
      <rdfs:subClassOf rdf:resource="#RED-MEAT"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SHELLFISH-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#SHELLFISH"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-FULL-BODY-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SHELLFISH">
      <rdfs:subClassOf rdf:resource="#SEAFOOD"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SEMILLON-OR-SAUVIGNON-BLANC">
      <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction rdf:about="#SEMILLON-INDIVIDUAL-OR-SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-
HAS-CLASS-RESTRICTION"/>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#SEMILLON-INDIVIDUAL-OR-SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-TO-
CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SEMILLON">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction rdf:about="#SEMILLON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
         <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
         <rdfs:Class rdf:about="#SEMILLON-OR-SAUVIGNON-BLANC"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SEAFOOD-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#SEAFOOD"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="SEAFOOD">
      <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
      <daml:disjointWith rdf:resource="#DESSERT"/>
      <daml:disjointWith rdf:resource="#FRUIT"/>
```

```
    </rdfs:Class>
    <rdfs:Class rdf:ID="SAUVIGNON-BLANC">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
          <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
          <rdfs:Class rdf:about="#SEMILLON-OR-SAUVIGNON-BLANC"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="SAUTERNE">
       <rdfs:subClassOf rdf:resource="#LATE-HARVEST"/>
       <rdfs:subClassOf rdf:resource="#BORDEAUX"/>
       <rdfs:subClassOf>
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasValue rdf:resource="#SAUTERNE-INDIVIDUAL"/>
          </daml:Restriction>
       </rdfs:subClassOf>
       <rdfs:subClassOf rdf:resource="#MEDIUM-BODY-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="SANCERRE">
       <rdfs:subClassOf rdf:resource="#MEDIUM-BODY-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#OFF-DRY-SUGAR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#SAUVIGNON-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasValue rdf:resource="#SANCERRE-INDIVIDUAL"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#LOIRE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="ROSE-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#ROSE-COLOR-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RIESLING">
       <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
             <daml:hasValue rdf:resource="#RIESLING-INDIVIDUAL"/>
          </daml:Restriction>
          <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#RED-COLOR-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-TABLE-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#RED-COLOR-RESTRICTION"/>
          <rdfs:Class rdf:about="#TABLE-WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-MEAT-COURSE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#FOOD"/>
             <daml:hasClass rdf:resource="#RED-MEAT"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#MEAL-COURSE"/>
       </daml:intersectionOf>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-MEAT">
       <rdfs:subClassOf rdf:resource="#MEAT"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-BURGUNDY">
       <rdfs:subClassOf rdf:resource="#PINOT-NOIR-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <rdfs:Class rdf:about="#BURGUNDY"/>
```

```
        <rdfs:Class rdf:about="#RED-WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="RED-BORDEAUX">
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
          <daml:toClass>
            <rdfs:Class>
              <daml:oneOf rdf:parseType="daml:collection">
                <rdf:Description rdf:about="#CABERNET-SAUVIGNON-INDIVIDUAL"/>
                <rdf:Description rdf:about="#MERLOT-INDIVIDUAL"/>
              </daml:oneOf>
            </rdfs:Class>
          </daml:toClass>
        </daml:Restriction>
      </rdfs:subClassOf>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <rdfs:Class rdf:about="#BORDEAUX"/>
        <rdfs:Class rdf:about="#RED-WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="POTABLE-LIQUID">
      <rdfs:subClassOf rdf:resource="#CONSUMABLE-THING"/>
      <daml:disjointWith rdf:resource="#EDIBLE-THING"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="PORT">
      <rdfs:subClassOf rdf:resource="#RED-WINE"/>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#REGION"/>
          <daml:hasValue rdf:resource="#PORTUGAL"/>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#FULL-BODY-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#STRONG-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#SWEET-SUGAR-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="PINOT-NOIR">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction rdf:about="#PINOT-NOIR-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="PINOT-BLANC">
      <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction rdf:about="#PINOT-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="PETITE-SYRAH">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
          <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
          <daml:hasValue rdf:resource="#PETITE-SYRAH-INDIVIDUAL"/>
        </daml:Restriction>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="PAUILLAC">
      <rdfs:subClassOf rdf:resource="#FULL-BODY-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#STRONG-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#CABERNET-SAUVIGNON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
          <daml:onProperty rdf:resource="#REGION"/>
          <daml:hasValue rdf:resource="#PAUILLAC-INDIVIDUAL"/>
        </daml:Restriction>
        <rdfs:Class rdf:about="#MEDOC"/>
      </daml:intersectionOf>
    </rdfs:Class>
```

```
<rdfs:Class rdf:ID="PASTA-WITH-WHITE-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA"/>
   <daml:disjointWith rdf:resource="#PASTA-WITH-RED-SAUCE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-SPICY-RED-SAUCE-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#PASTA-WITH-SPICY-RED-SAUCE"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-FULL-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-SPICY-RED-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA-WITH-RED-SAUCE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-RED-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-NON-SPICY-RED-SAUCE-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#PASTA-WITH-NON-SPICY-RED-SAUCE"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-NON-SPICY-RED-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA-WITH-RED-SAUCE"/>
   <daml:disjointWith rdf:resource="#PASTA-WITH-SPICY-RED-SAUCE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-LIGHT-CREAM-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA-WITH-WHITE-SAUCE"/>
   <daml:disjointWith rdf:resource="#PASTA-WITH-HEAVY-CREAM-SAUCE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-LIGHT-CREAM-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#PASTA-WITH-LIGHT-CREAM-SAUCE"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-LIGHT-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DELICATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-HEAVY-CREAM-SAUCE">
   <rdfs:subClassOf rdf:resource="#PASTA-WITH-WHITE-SAUCE"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA-WITH-HEAVY-CREAM-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#PASTA-WITH-HEAVY-CREAM-SAUCE"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="PASTA">
   <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
   <daml:disjointWith rdf:resource="#MEAT"/>
   <daml:disjointWith rdf:resource="#FOWL"/>
   <daml:disjointWith rdf:resource="#SEAFOOD"/>
   <daml:disjointWith rdf:resource="#DESSERT"/>
   <daml:disjointWith rdf:resource="#FRUIT"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="OYSTER-SHELLFISH-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#OYSTER-SHELLFISH"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-SWEET-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="OYSTER-SHELLFISH">
   <rdfs:subClassOf rdf:resource="#SHELLFISH"/>
</rdfs:Class>
<rdfs:Class rdf:ID="OTHER-TOMATO-BASED-FOOD-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#OTHER-TOMATO-BASED-FOOD"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="OTHER-TOMATO-BASED-FOOD">
   <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
   <daml:disjointWith rdf:resource="#PASTA"/>
   <daml:disjointWith rdf:resource="#MEAT"/>
   <daml:disjointWith rdf:resource="#FOWL"/>
   <daml:disjointWith rdf:resource="#SEAFOOD"/>
   <daml:disjointWith rdf:resource="#DESSERT"/>
   <daml:disjointWith rdf:resource="#FRUIT"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-SWEET-FRUIT-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#NON-SWEET-FRUIT"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DELICATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-SWEET-FRUIT">
   <rdfs:subClassOf rdf:resource="#FRUIT"/>
   <daml:disjointWith rdf:resource="#SWEET-FRUIT"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-SPICY-RED-MEAT-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#NON-SPICY-RED-MEAT"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-SPICY-RED-MEAT">
   <rdfs:subClassOf rdf:resource="#RED-MEAT"/>
   <daml:disjointWith rdf:resource="#SPICY-RED-MEAT"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-RED-MEAT-COURSE">
   <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Restriction>
         <daml:onProperty rdf:resource="#FOOD"/>
         <daml:hasClass rdf:resource="#NON-RED-MEAT"/>
      </daml:Restriction>
      <rdfs:Class rdf:about="#MEAL-COURSE"/>
   </daml:intersectionOf>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
   <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
</rdfs:Class>
<rdfs:Class rdf:ID="NON-RED-MEAT">
```

```
      <rdfs:subClassOf rdf:resource="#MEAT"/>
      <daml:disjointWith rdf:resource="#RED-MEAT"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="NON-OYSTER-SHELLFISH-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#NON-OYSTER-SHELLFISH"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="NON-OYSTER-SHELLFISH">
      <rdfs:subClassOf rdf:resource="#SHELLFISH"/>
      <daml:disjointWith rdf:resource="#OYSTER-SHELLFISH"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="NON-BLAND-FISH-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#NON-BLAND-FISH"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="NON-BLAND-FISH">
      <rdfs:subClassOf rdf:resource="#FISH"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="MUSCADET">
      <rdfs:subClassOf rdf:resource="#LIGHT-BODY-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#PINOT-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#MUSCADET-INDIVIDUAL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#LOIRE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="MEURSAULT">
      <rdfs:subClassOf rdf:resource="#FULL-BODY-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#MEURSAULT-INDIVIDUAL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#WHITE-BURGUNDY"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="MERLOT">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DELICATE-OR-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#LIGHT-OR-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction rdf:about="#MERLOT-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
         <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="MERITAGE">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
            <daml:toClass>
               <rdfs:Class>
                  <daml:oneOf rdf:parseType="daml:collection">
                     <rdf:Description rdf:about="#CABERNET-SAUVIGNON-INDIVIDUAL"/>
                     <rdf:Description rdf:about="#CABERNET-FRANC-INDIVIDUAL"/>
                     <rdf:Description rdf:about="#MALBEC"/>
                     <rdf:Description rdf:about="#PETITE-VERDOT"/>
                     <rdf:Description rdf:about="#MERLOT-INDIVIDUAL"/>
                  </daml:oneOf>
               </rdfs:Class>
            </daml:toClass>
```

```
        </daml:Restriction>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
          <daml:minCardinality>
            2
          </daml:minCardinality>
        </daml:Restriction>
        <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="MEDOC-REGION">
      <rdfs:subClassOf rdf:resource="#BORDEAUX-REGION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="MEDOC">
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
          <daml:onProperty rdf:resource="#REGION"/>
          <daml:hasClass rdf:resource="#MEDOC-REGION"/>
        </daml:Restriction>
        <rdfs:Class rdf:about="#BORDEAUX"/>
      </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="MEAT">
      <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
      <daml:disjointWith rdf:resource="#FOWL"/>
      <daml:disjointWith rdf:resource="#SEAFOOD"/>
      <daml:disjointWith rdf:resource="#DESSERT"/>
      <daml:disjointWith rdf:resource="#FRUIT"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="MEAL-COURSE">
      <rdfs:subClassOf rdf:resource="#CONSUMABLE-THING"/>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#FOOD"/>
          <daml:cardinality>
            1
          </daml:cardinality>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#FOOD"/>
          <daml:toClass rdf:resource="#EDIBLE-THING"/>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#DRINK"/>
          <daml:cardinality>
            1
          </daml:cardinality>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#DRINK"/>
          <daml:toClass rdf:resource="#WINE"/>
        </daml:Restriction>
      </rdfs:subClassOf>
      <daml:disjointWith rdf:resource="#POTABLE-LIQUID"/>
      <daml:disjointWith rdf:resource="#EDIBLE-THING"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="MEAL">
      <rdfs:subClassOf rdf:resource="#CONSUMABLE-THING"/>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#COURSE"/>
          <daml:minCardinality>
            1
          </daml:minCardinality>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction>
          <daml:onProperty rdf:resource="#COURSE"/>
          <daml:toClass rdf:resource="#MEAL-COURSE"/>
        </daml:Restriction>
      </rdfs:subClassOf>
      <daml:disjointWith rdf:resource="#MEAL-COURSE"/>
```

```
      <daml:disjointWith rdf:resource="#POTABLE-LIQUID"/>
      <daml:disjointWith rdf:resource="#EDIBLE-THING"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="MARGAUX">
      <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MERLOT-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#MARGAUX-INDIVIDUAL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEDOC"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="LOIRE-REGION">
      <rdfs:subClassOf rdf:resource="#FRENCH-REGION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="LOIRE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasClass rdf:resource="#LOIRE-REGION"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="LIGHT-MEAT-FOWL-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#LIGHT-MEAT-FOWL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-MODERATE-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="LIGHT-MEAT-FOWL">
      <rdfs:subClassOf rdf:resource="#FOWL"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="LATE-HARVEST">
      <rdfs:subClassOf rdf:resource="#WINE"/>
      <daml:disjointWith rdf:resource="#EARLY-HARVEST"/>
      <rdfs:subClassOf rdf:resource="#SWEET-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="ITALIAN-WINE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasClass rdf:resource="#ITALIAN-REGION"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="ITALIAN-REGION">
      <rdfs:subClassOf rdf:resource="#WINE-REGION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="ICE-WINE">
      <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <rdfs:Class rdf:about="#LATE-HARVEST"/>
         <rdfs:Class rdf:about="#DESSERT-WINE"/>
         <daml:Restriction rdf:about="#WHITE-COLOR-RESTRICTION"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="GRAPE">
      <rdfs:subClassOf rdf:resource="#SWEET-FRUIT"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="GERMAN-WINE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#GERMANY"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#WINE"/>
      </daml:intersectionOf>
```

```
    </rdfs:Class>
    <rdfs:Class rdf:ID="GAMAY">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#GAMAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
          <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FULL-BODIED-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#FULL-BODY-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FRUIT-COURSE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#FOOD"/>
             <daml:hasClass rdf:resource="#FRUIT"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#MEAL-COURSE"/>
       </daml:intersectionOf>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-WHITE-COLOR-TO-CLASS-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FRUIT">
       <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FRENCH-REGION">
       <rdfs:subClassOf rdf:resource="#WINE-REGION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FOWL">
       <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
       <daml:disjointWith rdf:resource="#SEAFOOD"/>
       <daml:disjointWith rdf:resource="#DESSERT"/>
       <daml:disjointWith rdf:resource="#FRUIT"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FISH-COURSE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#FOOD"/>
             <daml:hasClass rdf:resource="#FISH"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#MEAL-COURSE"/>
       </daml:intersectionOf>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="FISH">
       <rdfs:subClassOf rdf:resource="#SEAFOOD"/>
       <daml:disjointWith rdf:resource="#SHELLFISH"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="EDIBLE-THING">
       <rdfs:subClassOf rdf:resource="#CONSUMABLE-THING"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="EATING-GRAPE">
       <rdfs:subClassOf rdf:resource="#GRAPE"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="EARLY-HARVEST">
       <rdfs:subClassOf rdf:resource="#WINE"/>
       <rdfs:subClassOf rdf:resource="#DRY-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="DRY-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#DRY-SUGAR-RESTRICTION"/>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="DRY-WHITE-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <rdfs:Class rdf:about="#DRY-WINE"/>
          <rdfs:Class rdf:about="#WHITE-WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="DRY-RIESLING">
       <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#LIGHT-OR-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction rdf:about="#DRY-SUGAR-RESTRICTION"/>
          <rdfs:Class rdf:about="#RIESLING"/>
```

```
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DRY-RED-WINE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <rdfs:Class rdf:about="#DRY-WINE"/>
         <rdfs:Class rdf:about="#RED-WINE"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DESSERT-WINE">
      <rdfs:subClassOf rdf:resource="#WINE"/>
      <rdfs:subClassOf rdf:resource="#SWEET-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DESSERT-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#DESSERT"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-SWEET-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DESSERT">
      <rdfs:subClassOf rdf:resource="#EDIBLE-THING"/>
      <daml:disjointWith rdf:resource="#FRUIT"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DARK-MEAT-FOWL-COURSE">
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#FOOD"/>
            <daml:hasClass rdf:resource="#DARK-MEAT-FOWL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#MEAL-COURSE"/>
      </daml:intersectionOf>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-LIGHT-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-DELICATE-FLAVOR-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRINK-HAS-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="DARK-MEAT-FOWL">
      <rdfs:subClassOf rdf:resource="#FOWL"/>
      <daml:disjointWith rdf:resource="#LIGHT-MEAT-FOWL"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="COTES-D-OR">
      <rdfs:subClassOf rdf:resource="#MODERATE-FLAVOR-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#COTES-D-OR-INDIVIDUAL"/>
         </daml:Restriction>
         <rdfs:Class rdf:about="#RED-BURGUNDY"/>
      </daml:intersectionOf>
   </rdfs:Class>
   <rdfs:Class rdf:ID="CONSUMABLE-THING"/>
   <rdfs:Class rdf:ID="CHIANTI">
      <rdfs:subClassOf rdf:resource="#ITALIAN-WINE"/>
      <rdfs:subClassOf>
         <daml:Restriction>
            <daml:onProperty rdf:resource="#REGION"/>
            <daml:hasValue rdf:resource="#CHIANTI-INDIVIDUAL"/>
         </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
      <rdfs:subClassOf>
         <daml:Restriction>
            <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
            <daml:hasValue rdf:resource="#SANGIOVESE"/>
         </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="#MODERATE-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#LIGHT-OR-MEDIUM-BODY-TO-CLASS-RESTRICTION"/>
   </rdfs:Class>
   <rdfs:Class rdf:ID="CHENIN-BLANC">
      <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MODERATE-FLAVOR-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
      <rdfs:subClassOf rdf:resource="#DRY-OR-OFF-DRY-SUGAR-TO-CLASS-RESTRICTION"/>
      <daml:intersectionOf rdf:parseType="daml:collection">
```

```
        <daml:Restriction rdf:about="#CHENIN-BLANC-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CHEESE-NUTS-DESSERT-COURSE">
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
           <daml:onProperty rdf:resource="#FOOD"/>
           <daml:hasClass rdf:resource="#CHEESE-NUTS-DESSERT"/>
        </daml:Restriction>
        <rdfs:Class rdf:about="#MEAL-COURSE"/>
     </daml:intersectionOf>
     <rdfs:subClassOf rdf:resource="#DRINK-HAS-RED-COLOR-TO-CLASS-RESTRICTION"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CHEESE-NUTS-DESSERT">
     <rdfs:subClassOf rdf:resource="#DESSERT"/>
     <daml:disjointWith rdf:resource="#SWEET-DESSERT"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CHARDONNAY">
     <rdfs:subClassOf rdf:resource="#WHITE-COLOR-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction rdf:about="#CHARDONNAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CALIFORNIAN-REGION">
     <rdfs:subClassOf rdf:resource="#US-REGION"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CALIFORNIA-WINE">
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
           <daml:onProperty rdf:resource="#REGION"/>
           <daml:hasClass rdf:resource="#CALIFORNIAN-REGION"/>
        </daml:Restriction>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CABERNET-SAUVIGNON">
     <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MODERATE-OR-STRONG-FLAVOR-TO-CLASS-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MEDIUM-OR-FULL-BODY-TO-CLASS-RESTRICTION"/>
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction rdf:about="#CABERNET-SAUVIGNON-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="CABERNET-FRANC">
     <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MODERATE-FLAVOR-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#MEDIUM-BODY-RESTRICTION"/>
     <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
           <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
           <daml:hasValue rdf:resource="#CABERNET-FRANC-INDIVIDUAL"/>
        </daml:Restriction>
        <daml:Restriction rdf:about="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="BURGUNDY">
     <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
     <daml:intersectionOf rdf:parseType="daml:collection">
        <daml:Restriction>
           <daml:onProperty rdf:resource="#REGION"/>
           <daml:hasClass rdf:resource="#BOURGOGNE-REGION"/>
        </daml:Restriction>
        <rdfs:Class rdf:about="#WINE"/>
     </daml:intersectionOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="BOURGOGNE-REGION">
     <rdfs:subClassOf rdf:resource="#FRENCH-REGION"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="BORDEAUX-REGION">
     <rdfs:subClassOf rdf:resource="#FRENCH-REGION"/>
```

```
    </rdfs:Class>
    <rdfs:Class rdf:ID="BORDEAUX">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasClass rdf:resource="#BORDEAUX-REGION"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="BLAND-FISH-COURSE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#FOOD"/>
             <daml:hasClass rdf:resource="#BLAND-FISH"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#MEAL-COURSE"/>
       </daml:intersectionOf>
       <rdfs:subClassOf rdf:resource="#DRINK-HAS-DELICATE-FLAVOR-TO-CLASS-RESTRICTION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="BLAND-FISH">
       <rdfs:subClassOf rdf:resource="#FISH"/>
       <daml:disjointWith rdf:resource="#NON-BLAND-FISH"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="BEAUJOLAIS">
       <rdfs:subClassOf rdf:resource="#RED-COLOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#LIGHT-BODY-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DRY-SUGAR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#GAMAY-INDIVIDUAL-GRAPE-SLOT-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#GRAPE-SLOT-MAX-CARDINALITY-1-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasValue rdf:resource="#BEAUJOLAIS-INDIVIDUAL"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="AUSTRALIAN-REGION">
       <rdfs:subClassOf rdf:resource="#WINE-REGION"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="ANJOU">
       <rdfs:subClassOf rdf:resource="#ROSE-COLOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#LIGHT-BODY-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#DELICATE-FLAVOR-RESTRICTION"/>
       <rdfs:subClassOf rdf:resource="#OFF-DRY-SUGAR-RESTRICTION"/>
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasValue rdf:resource="#ANJOU-INDIVIDUAL"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#LOIRE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="AMERICAN-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasClass rdf:resource="#US-REGION"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="ALSATIAN-WINE">
       <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Restriction>
             <daml:onProperty rdf:resource="#REGION"/>
             <daml:hasValue rdf:resource="#ALSACE"/>
          </daml:Restriction>
          <rdfs:Class rdf:about="#WINE"/>
       </daml:intersectionOf>
    </rdfs:Class>
    <rdf:Description rdf:ID="FULL">
       <rdf:type rdf:resource="#WINE-BODY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="MEDIUM">
       <rdf:type rdf:resource="#WINE-BODY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="LIGHT">
       <rdf:type rdf:resource="#WINE-BODY"/>
```

```
    </rdf:Description>
    <rdf:Description rdf:ID="RED">
       <rdf:type rdf:resource="#WINE-COLOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="ROSE">
       <rdf:type rdf:resource="#WINE-COLOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="WHITE">
       <rdf:type rdf:resource="#WINE-COLOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="STRONG">
       <rdf:type rdf:resource="#WINE-FLAVOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="MODERATE">
       <rdf:type rdf:resource="#WINE-FLAVOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="DELICATE">
       <rdf:type rdf:resource="#WINE-FLAVOR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="DRY">
       <rdf:type rdf:resource="#WINE-SUGAR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="OFF-DRY">
       <rdf:type rdf:resource="#WINE-SUGAR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="SWEET">
       <rdf:type rdf:resource="#WINE-SUGAR"/>
    </rdf:Description>
    <rdf:Description rdf:ID="ALSACE">
       <rdf:type rdf:resource="#FRENCH-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="ANJOU-INDIVIDUAL">
       <rdf:type rdf:resource="#LOIRE-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="APPLES">
       <rdf:type rdf:resource="#NON-SWEET-FRUIT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="ARROYO-GRANDE">
       <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="BANANAS">
       <rdf:type rdf:resource="#SWEET-FRUIT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="BANCROFT">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="BANCROFT-CHARDONNAY">
       <rdf:type rdf:resource="#CHARDONNAY"/>
       <REGION rdf:resource="#NAPA"/>
       <MAKER rdf:resource="#BANCROFT"/>
       <SUGAR rdf:resource="#DRY"/>
       <FLAVOR rdf:resource="#MODERATE"/>
       <BODY rdf:resource="#MEDIUM"/>
    </rdf:Description>
    <rdf:Description rdf:ID="BEAUJOLAIS-INDIVIDUAL">
       <rdf:type rdf:resource="#FRENCH-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="BEEF-CURRY">
       <rdf:type rdf:resource="#SPICY-RED-MEAT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CABERNET-FRANC-INDIVIDUAL">
       <rdf:type rdf:resource="#WINE-GRAPE"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CABERNET-SAUVIGNON-INDIVIDUAL">
       <rdf:type rdf:resource="#WINE-GRAPE"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CAKE">
       <rdf:type rdf:resource="#SWEET-DESSERT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CENTRAL-COAST">
       <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHARDONNAY-INDIVIDUAL">
       <rdf:type rdf:resource="#WINE-GRAPE"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-CHEVAL-BLANC">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-CHEVAL-BLANC-ST-EMILION">
       <rdf:type rdf:resource="#ST-EMILION"/>
       <MAKER rdf:resource="#CHATEAU-CHEVAL-BLANC"/>
```

```
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-D-YCHEM">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-D-YCHEM-SAUTERNE">
       <rdf:type rdf:resource="#SAUTERNE"/>
       <GRAPE-SLOT rdf:resource="#SAUVIGNON-BLANC-INDIVIDUAL"/>
       <GRAPE-SLOT rdf:resource="#SEMILLON-INDIVIDUAL"/>
       <FLAVOR rdf:resource="#STRONG"/>
       <MAKER rdf:resource="#CHATEAU-D-YCHEM"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-DE-MEURSAULT">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-DE-MEURSAULT-MEURSAULT">
       <rdf:type rdf:resource="#MEURSAULT"/>
       <FLAVOR rdf:resource="#MODERATE"/>
       <MAKER rdf:resource="#CHATEAU-DE-MEURSAULT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-LAFITE-ROTHSCHILD">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-LAFITE-ROTHSCHILD-PAUILLAC">
       <rdf:type rdf:resource="#PAUILLAC"/>
       <MAKER rdf:resource="#CHATEAU-LAFITE-ROTHSCHILD"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-MARGAUX">
       <rdf:type rdf:resource="#MARGAUX"/>
       <MAKER rdf:resource="#CHATEAU-MARGAUX-WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-MARGAUX-WINERY">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-MORGON">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHATEAU-MORGON-BEAUJOLAIS">
       <rdf:type rdf:resource="#BEAUJOLAIS"/>
       <MAKER rdf:resource="#CHATEAU-MORGON"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHEESE">
       <rdf:type rdf:resource="#CHEESE-NUTS-DESSERT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHENIN-BLANC-INDIVIDUAL">
       <rdf:type rdf:resource="#WINE-GRAPE"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHIANTI-CLASSICO">
       <rdf:type rdf:resource="#CHIANTI"/>
       <BODY rdf:resource="#MEDIUM"/>
       <MAKER rdf:resource="#MCGUINNESSO"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHIANTI-INDIVIDUAL">
       <rdf:type rdf:resource="#ITALIAN-REGION"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CHICKEN">
       <rdf:type rdf:resource="#LIGHT-MEAT-FOWL"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CLAMS">
       <rdf:type rdf:resource="#NON-OYSTER-SHELLFISH"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CLOS-DE-LA-POUSSIE">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CLOS-DE-LA-POUSSIE-SANCERRE">
       <rdf:type rdf:resource="#SANCERRE"/>
       <MAKER rdf:resource="#CLOS-DE-LA-POUSSIE"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CLOS-DE-VOUGEOT">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CLOS-DE-VOUGEOT-COTES-D-OR">
       <rdf:type rdf:resource="#COTES-D-OR"/>
       <MAKER rdf:resource="#CLOS-DE-VOUGEOT"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CONGRESS-SPRINGS">
       <rdf:type rdf:resource="#WINERY"/>
    </rdf:Description>
    <rdf:Description rdf:ID="CONGRESS-SPRINGS-SEMILLON">
       <rdf:type rdf:resource="#SEMILLON"/>
       <MAKER rdf:resource="#CONGRESS-SPRINGS"/>
       <SUGAR rdf:resource="#DRY"/>
```

```
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORBANS">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORBANS-DRY-WHITE-RIESLING">
      <rdf:type rdf:resource="#RIESLING"/>
      <REGION rdf:resource="#NEW-ZEALAND"/>
      <MAKER rdf:resource="#CORBANS"/>
      <SUGAR rdf:resource="#OFF-DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORBANS-PRIVATE-BIN-SAUVIGNON-BLANC">
      <rdf:type rdf:resource="#SAUVIGNON-BLANC"/>
      <REGION rdf:resource="#NEW-ZEALAND"/>
      <MAKER rdf:resource="#CORBANS"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORBANS-SAUVIGNON-BLANC">
      <rdf:type rdf:resource="#SAUVIGNON-BLANC"/>
      <REGION rdf:resource="#NEW-ZEALAND"/>
      <MAKER rdf:resource="#CORBANS"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORTON-MONTRACHET">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CORTON-MONTRACHET-WHITE-BURGUNDY">
      <rdf:type rdf:resource="#WHITE-BURGUNDY"/>
      <MAKER rdf:resource="#CORTON-MONTRACHET"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="COTES-D-OR-INDIVIDUAL">
      <rdf:type rdf:resource="#BOURGOGNE-REGION"/>
   </rdf:Description>
   <rdf:Description rdf:ID="COTTURI">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="COTTURI-ZINFANDEL">
      <rdf:type rdf:resource="#ZINFANDEL"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#COTTURI"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="CRAB">
      <rdf:type rdf:resource="#NON-OYSTER-SHELLFISH"/>
   </rdf:Description>
   <rdf:Description rdf:ID="D-ANJOU">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="DUCK">
      <rdf:type rdf:resource="#DARK-MEAT-FOWL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="EDNA-VALLEY">
      <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
   </rdf:Description>
   <rdf:Description rdf:ID="ELYSE">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="ELYSE-ZINFANDEL">
      <rdf:type rdf:resource="#ZINFANDEL"/>
      <REGION rdf:resource="#NAPA"/>
      <MAKER rdf:resource="#ELYSE"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FETTUCINE-ALFREDO">
      <rdf:type rdf:resource="#PASTA-WITH-HEAVY-CREAM-SAUCE"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FLOUNDER">
```

```
      <rdf:type rdf:resource="#BLAND-FISH"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FORMAN">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FORMAN-CABERNET-SAUVIGNON">
      <rdf:type rdf:resource="#CABERNET-SAUVIGNON"/>
      <REGION rdf:resource="#NAPA"/>
      <MAKER rdf:resource="#FORMAN"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FORMAN-CHARDONNAY">
      <rdf:type rdf:resource="#CHARDONNAY"/>
      <REGION rdf:resource="#NAPA"/>
      <MAKER rdf:resource="#FORMAN"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FOXEN">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FOXEN-CHENIN-BLANC">
      <rdf:type rdf:resource="#CHENIN-BLANC"/>
      <REGION rdf:resource="#SANTA-BARBARA"/>
      <MAKER rdf:resource="#FOXEN"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="FRA-DIAVOLO">
      <rdf:type rdf:resource="#PASTA-WITH-SPICY-RED-SAUCE"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GAMAY-INDIVIDUAL">
      <rdf:type rdf:resource="#WINE-GRAPE"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GARLICKY-ROAST">
      <rdf:type rdf:resource="#SPICY-RED-MEAT"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GARY-FARRELL">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GARY-FARRELL-MERLOT">
      <rdf:type rdf:resource="#MERLOT"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#GARY-FARRELL"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GERMANY">
      <rdf:type rdf:resource="#WINE-REGION"/>
   </rdf:Description>
   <rdf:Description rdf:ID="GOOSE">
      <rdf:type rdf:resource="#DARK-MEAT-FOWL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="HALIBUT">
      <rdf:type rdf:resource="#BLAND-FISH"/>
   </rdf:Description>
   <rdf:Description rdf:ID="HANDLEY">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="KALIN-CELLARS">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="KALIN-CELLARS-SEMILLON">
      <rdf:type rdf:resource="#SEMILLON"/>
      <MAKER rdf:resource="#KALIN-CELLARS"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#STRONG"/>
      <BODY rdf:resource="#FULL"/>
   </rdf:Description>
   <rdf:Description rdf:ID="KATHRYN-KENNEDY">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="KATHRYN-KENNEDY-LATERAL">
      <rdf:type rdf:resource="#MERITAGE"/>
      <MAKER rdf:resource="#KATHRYN-KENNEDY"/>
      <SUGAR rdf:resource="#DRY"/>
```

```
      <FLAVOR rdf:resource="#DELICATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="LANE-TANNER">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="LANE-TANNER-PINOT-NOIR">
      <rdf:type rdf:resource="#PINOT-NOIR"/>
      <REGION rdf:resource="#SANTA-BARBARA"/>
      <MAKER rdf:resource="#LANE-TANNER"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#DELICATE"/>
      <BODY rdf:resource="#LIGHT"/>
   </rdf:Description>
   <rdf:Description rdf:ID="LOBSTER">
      <rdf:type rdf:resource="#NON-OYSTER-SHELLFISH"/>
   </rdf:Description>
   <rdf:Description rdf:ID="LONGRIDGE">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="LONGRIDGE-MERLOT">
      <rdf:type rdf:resource="#MERLOT"/>
      <REGION rdf:resource="#NEW-ZEALAND"/>
      <MAKER rdf:resource="#LONGRIDGE"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#LIGHT"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MALBEC">
      <rdf:type rdf:resource="#WINE-GRAPE"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARGAUX-INDIVIDUAL">
      <rdf:type rdf:resource="#MEDOC-REGION"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARIETTA">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARIETTA-CABERNET-SAUVIGNON">
      <rdf:type rdf:resource="#CABERNET-SAUVIGNON"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#MARIETTA"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARIETTA-OLD-VINES-RED">
      <rdf:type rdf:resource="#RED-TABLE-WINE"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#MARIETTA"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARIETTA-PETITE-SYRAH">
      <rdf:type rdf:resource="#PETITE-SYRAH"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#MARIETTA"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MARIETTA-ZINFANDEL">
      <rdf:type rdf:resource="#ZINFANDEL"/>
      <REGION rdf:resource="#SONOMA"/>
      <MAKER rdf:resource="#MARIETTA"/>
      <SUGAR rdf:resource="#DRY"/>
      <FLAVOR rdf:resource="#MODERATE"/>
      <BODY rdf:resource="#MEDIUM"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MCGUINNESSO">
      <rdf:type rdf:resource="#WINERY"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MENDOCINO">
      <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MERLOT-INDIVIDUAL">
      <rdf:type rdf:resource="#WINE-GRAPE"/>
   </rdf:Description>
   <rdf:Description rdf:ID="MEURSAULT-INDIVIDUAL">
      <rdf:type rdf:resource="#BOURGOGNE-REGION"/>
   </rdf:Description>
```

```
<rdf:Description rdf:ID="MIXED-FRUIT">
   <rdf:type rdf:resource="#SWEET-FRUIT"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNT-EDEN-VINEYARD">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNT-EDEN-VINEYARD-EDNA-VALLEY-CHARDONNAY">
   <rdf:type rdf:resource="#CHARDONNAY"/>
   <REGION rdf:resource="#EDNA-VALLEY"/>
   <MAKER rdf:resource="#MOUNT-EDEN-VINEYARD"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNT-EDEN-VINEYARD-ESTATE-PINOT-NOIR">
   <rdf:type rdf:resource="#PINOT-NOIR"/>
   <REGION rdf:resource="#SANTA-CRUZ-MOUNTAINS"/>
   <MAKER rdf:resource="#MOUNT-EDEN-VINEYARD"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#STRONG"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNTADAM">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNTADAM-CHARDONNAY">
   <rdf:type rdf:resource="#CHARDONNAY"/>
   <REGION rdf:resource="#SOUTH-AUSTRALIA"/>
   <MAKER rdf:resource="#MOUNTADAM"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#STRONG"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNTADAM-PINOT-NOIR">
   <rdf:type rdf:resource="#PINOT-NOIR"/>
   <REGION rdf:resource="#SOUTH-AUSTRALIA"/>
   <MAKER rdf:resource="#MOUNTADAM"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="MOUNTADAM-RIESLING">
   <rdf:type rdf:resource="#DRY-RIESLING"/>
   <REGION rdf:resource="#SOUTH-AUSTRALIA"/>
   <MAKER rdf:resource="#MOUNTADAM"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#DELICATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="MUSCADET-INDIVIDUAL">
   <rdf:type rdf:resource="#LOIRE-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="MUSSELS">
   <rdf:type rdf:resource="#NON-OYSTER-SHELLFISH"/>
</rdf:Description>
<rdf:Description rdf:ID="NAPA">
   <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="NEW-ZEALAND">
   <rdf:type rdf:resource="#WINE-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="NUTS">
   <rdf:type rdf:resource="#CHEESE-NUTS-DESSERT"/>
</rdf:Description>
<rdf:Description rdf:ID="OYSTERS">
   <rdf:type rdf:resource="#OYSTER-SHELLFISH"/>
</rdf:Description>
<rdf:Description rdf:ID="PAGE-MILL-WINERY">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="PAGE-MILL-WINERY-CABERNET-SAUVIGNON">
   <rdf:type rdf:resource="#CABERNET-SAUVIGNON"/>
   <REGION rdf:resource="#NAPA"/>
   <MAKER rdf:resource="#PAGE-MILL-WINERY"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="PASTA-WITH-WHITE-CLAM-SAUCE">
   <rdf:type rdf:resource="#PASTA-WITH-LIGHT-CREAM-SAUCE"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="PAUILLAC-INDIVIDUAL">
   <rdf:type rdf:resource="#MEDOC-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="PEACHES">
   <rdf:type rdf:resource="#SWEET-FRUIT"/>
</rdf:Description>
<rdf:Description rdf:ID="PETER-MCCOY">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="PETER-MCCOY-CHARDONNAY">
   <rdf:type rdf:resource="#CHARDONNAY"/>
   <REGION rdf:resource="#SONOMA"/>
   <MAKER rdf:resource="#PETER-MCCOY"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="PETITE-SYRAH-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="PETITE-VERDOT">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="PIE">
   <rdf:type rdf:resource="#SWEET-DESSERT"/>
</rdf:Description>
<rdf:Description rdf:ID="PINOT-BLANC-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="PINOT-NOIR-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="PIZZA">
   <rdf:type rdf:resource="#OTHER-TOMATO-BASED-FOOD"/>
</rdf:Description>
<rdf:Description rdf:ID="PORK">
   <rdf:type rdf:resource="#NON-RED-MEAT"/>
</rdf:Description>
<rdf:Description rdf:ID="PORTUGAL">
   <rdf:type rdf:resource="#WINE-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="PULIGNY-MONTRACHET">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="PULIGNY-MONTRACHET-WHITE-BURGUNDY">
   <rdf:type rdf:resource="#WHITE-BURGUNDY"/>
   <MAKER rdf:resource="#PULIGNY-MONTRACHET"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="RIESLING-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="ROAST-BEEF">
   <rdf:type rdf:resource="#NON-SPICY-RED-MEAT"/>
</rdf:Description>
<rdf:Description rdf:ID="ROSE-D-ANJOU">
   <rdf:type rdf:resource="#ANJOU"/>
   <MAKER rdf:resource="#D-ANJOU"/>
</rdf:Description>
<rdf:Description rdf:ID="SANCERRE-INDIVIDUAL">
   <rdf:type rdf:resource="#LOIRE-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SANGIOVESE">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="SANTA-BARBARA">
   <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SANTA-CRUZ-MOUNTAIN-VINEYARD">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SANTA-CRUZ-MOUNTAIN-VINEYARD-CABERNET-SAUVIGNON">
   <rdf:type rdf:resource="#CABERNET-SAUVIGNON"/>
   <REGION rdf:resource="#SANTA-CRUZ-MOUNTAINS"/>
   <MAKER rdf:resource="#SANTA-CRUZ-MOUNTAIN-VINEYARD"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#STRONG"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="SANTA-CRUZ-MOUNTAINS">
   <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SAUCELITO-CANYON">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SAUCELITO-CANYON-ZINFANDEL">
   <rdf:type rdf:resource="#ZINFANDEL"/>
   <REGION rdf:resource="#ARROYO-GRANDE"/>
   <MAKER rdf:resource="#SAUCELITO-CANYON"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="SAUTERNE-INDIVIDUAL">
   <rdf:type rdf:resource="#BORDEAUX-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SAUVIGNON-BLANC-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="SCHLOSS-ROTHERMEL">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SCHLOSS-ROTHERMEL-TROCHENBIERENAUSLESE-RIESLING">
   <rdf:type rdf:resource="#SWEET-RIESLING"/>
   <REGION rdf:resource="#GERMANY"/>
   <MAKER rdf:resource="#SCHLOSS-ROTHERMEL"/>
   <SUGAR rdf:resource="#SWEET"/>
   <FLAVOR rdf:resource="#STRONG"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
<rdf:Description rdf:ID="SCHLOSS-VOLRAD">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SCHLOSS-VOLRAD-TROCHENBIERENAUSLESE-RIESLING">
   <rdf:type rdf:resource="#SWEET-RIESLING"/>
   <REGION rdf:resource="#GERMANY"/>
   <MAKER rdf:resource="#SCHLOSS-VOLRAD"/>
   <SUGAR rdf:resource="#SWEET"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
<rdf:Description rdf:ID="SCROD">
   <rdf:type rdf:resource="#BLAND-FISH"/>
</rdf:Description>
<rdf:Description rdf:ID="SEAN-THACKREY">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SEAN-THACKREY-SIRIUS-PETITE-SYRAH">
   <rdf:type rdf:resource="#PETITE-SYRAH"/>
   <REGION rdf:resource="#NAPA"/>
   <MAKER rdf:resource="#SEAN-THACKREY"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#STRONG"/>
   <BODY rdf:resource="#FULL"/>
</rdf:Description>
<rdf:Description rdf:ID="SELAKS">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="SELAKS-ICE-WINE">
   <rdf:type rdf:resource="#ICE-WINE"/>
   <REGION rdf:resource="#NEW-ZEALAND"/>
   <MAKER rdf:resource="#SELAKS"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
   <COLOR rdf:resource="#WHITE"/>
</rdf:Description>
<rdf:Description rdf:ID="SELAKS-SAUVIGNON-BLANC">
   <rdf:type rdf:resource="#SAUVIGNON-BLANC"/>
   <REGION rdf:resource="#NEW-ZEALAND"/>
   <MAKER rdf:resource="#SELAKS"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="SEMILLON-INDIVIDUAL">
   <rdf:type rdf:resource="#WINE-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="SEVRE-ET-MAINE">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="SEVRE-ET-MAINE-MUSCADET">
   <rdf:type rdf:resource="#MUSCADET"/>
   <MAKER rdf:resource="#SEVRE-ET-MAINE"/>
</rdf:Description>
<rdf:Description rdf:ID="SONOMA">
   <rdf:type rdf:resource="#CALIFORNIAN-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SOUTH-AUSTRALIA">
   <rdf:type rdf:resource="#AUSTRALIAN-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="SPAGHETTI-WITH-TOMATO-SAUCE">
   <rdf:type rdf:resource="#PASTA-WITH-NON-SPICY-RED-SAUCE"/>
</rdf:Description>
<rdf:Description rdf:ID="ST-EMILION-INDIVIDUAL">
   <rdf:type rdf:resource="#BORDEAUX-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="STEAK">
   <rdf:type rdf:resource="#NON-SPICY-RED-MEAT"/>
</rdf:Description>
<rdf:Description rdf:ID="STONLEIGH">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="STONLEIGH-SAUVIGNON-BLANC">
   <rdf:type rdf:resource="#SAUVIGNON-BLANC"/>
   <REGION rdf:resource="#NEW-ZEALAND"/>
   <MAKER rdf:resource="#STONLEIGH"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#DELICATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="SWORDFISH">
   <rdf:type rdf:resource="#NON-BLAND-FISH"/>
</rdf:Description>
<rdf:Description rdf:ID="TAYLOR">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="TAYLOR-PORT">
   <rdf:type rdf:resource="#PORT"/>
   <MAKER rdf:resource="#TAYLOR"/>
</rdf:Description>
<rdf:Description rdf:ID="THOMPSON-SEEDLESS">
   <rdf:type rdf:resource="#EATING-GRAPE"/>
</rdf:Description>
<rdf:Description rdf:ID="TOURS-INDIVIDUAL">
   <rdf:type rdf:resource="#LOIRE-REGION"/>
</rdf:Description>
<rdf:Description rdf:ID="TUNA">
   <rdf:type rdf:resource="#NON-BLAND-FISH"/>
</rdf:Description>
<rdf:Description rdf:ID="TURKEY">
   <rdf:type rdf:resource="#LIGHT-MEAT-FOWL"/>
</rdf:Description>
<rdf:Description rdf:ID="VEAL">
   <rdf:type rdf:resource="#NON-SPICY-RED-MEAT"/>
</rdf:Description>
<rdf:Description rdf:ID="VENTANA">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="VENTANA-CHENIN-BLANC">
   <rdf:type rdf:resource="#CHENIN-BLANC"/>
   <REGION rdf:resource="#CENTRAL-COAST"/>
   <MAKER rdf:resource="#VENTANA"/>
   <SUGAR rdf:resource="#OFF-DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="WHITEHALL-LANE">
   <rdf:type rdf:resource="#WINERY"/>
</rdf:Description>
<rdf:Description rdf:ID="WHITEHALL-LANE-CABERNET-FRANC">
   <rdf:type rdf:resource="#CABERNET-FRANC"/>
   <REGION rdf:resource="#NAPA"/>
   <MAKER rdf:resource="#WHITEHALL-LANE"/>
   <SUGAR rdf:resource="#DRY"/>
   <FLAVOR rdf:resource="#MODERATE"/>
   <BODY rdf:resource="#MEDIUM"/>
</rdf:Description>
<rdf:Description rdf:ID="WHITEHALL-LANE-PRIMAVERA">
   <rdf:type rdf:resource="#DESSERT-WINE"/>
   <REGION rdf:resource="#NAPA"/>
   <MAKER rdf:resource="#WHITEHALL-LANE"/>
```

```
      <SUGAR rdf:resource="#SWEET"/>
      <FLAVOR rdf:resource="#DELICATE"/>
      <BODY rdf:resource="#LIGHT"/>
   </rdf:Description>
   <rdf:Description rdf:ID="ZINFANDEL-INDIVIDUAL">
      <rdf:type rdf:resource="#WINE-GRAPE"/>
   </rdf:Description>
</rdf:RDF>
```

Produced from http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml using hyperdaml.java

APPENDIX_19 Possible extension: Make suggestions of typical menus of each country.

One possible design of this feature is to have some auxiliary database, with the name of the country in one column and the typical ingredients, way of cooking or the name of some typical dishes associated to it.
For example:

Country	Typical ingredients
Spain	Legumes
Spain	Vegetables
Denmark	Pork
Denmark	Potatoes
China	Rise
Pakistan	Rise
Pakistan	Paprika

Country	Typical way of cooking
Spain	Fry
Denmark	Bake
Denmark	Boil
Spain	Stew

Country	Typical dishes (name of the recipe)
Spain	Paella
Spain	Churros (flour fritter eaten with coffee or hot chocolate)
Denmark	Ris a l'amande
Denmark	rød grød med fløde
France	Ris a l'amande
Denmark	Stægt flæske med persille
Denmark	Koldskål
Italy	Pizza
Italy	Pasta
Italy	Tomato
Italy	Oregano

This feature can be modeled as part of the Ontology or as auxiliary databases (in XML for instance). Each time the system needs to suggest a typical menu, it can check both the recipes in the knowledge base and these auxiliary tables, and then suggest the menu.

APPENDIX_20 MEASUREMENTS CONVERSIONS

The measurement system differs from country to country. This appendix will relate the differences of each one, and how to relate them to allow comparisons among recipes.

There are three different measurement systems: Imperial, US Standard and Metric.
All the ingredients of the recipes have to be related to the same system in order to allow comparisons.

Besides from these different systems, the way of measuring ingredients also varies from different countries.

In the United States for example, cooks use to measure all the ingredients by volume (e.g.: cups, spoons, milliliters, liters, etc…) whether in Europe the trend is to measure dry ingredients by weight (kilograms, grams, etc..) and liquid ingredients by volume (liters, milliliters…)

The following tables show the conversions among the different measurement units.

20.1 Volume measures

TEASPOON	SPOON	CUP	LITER	MILILITER	FLUID OUNCE	PINT	QUART
0.2029	0.0676	0.00423	0.001	1.0	0.0338	0.00211	0.00106
1.0	0.33	0.02	0.00497	4.97	0.17	0.01	0.005
3.0	1.0	0.06	0.0149	14.9	0.5	0.03	0.015
6.0	2.0	0.125	0.029	29.6	1.0	0.0625	0.03
48	16	1.0	0.2366	236.6	8.0	0.5	0.25
96	32	2.0	0.4732	473.2	16.0	1.0	0.5
192.0	64	4.0	0.9464	946.4	32	2.0	1.0
202.9	67.6	4.23	1.0	1000	33.8	2.11	1.06

20.2 Weight measures

POUND	OUNCE	KILOGRAM	GRAM
0.0022	0.0353	0.001	1.0
1.0	16	0.454	453.59
0.0625	1.0	0.028	28.35
2.2	35.3	1.0	1000

The conversion between weight measures and volume measures can only be made straight in the case of the water, because it is the only element where its weight is equal to its volume (1

kg = 1 Liter) or (1 oz = 1 oz. fl). In the rest of the ingredients another study of their density is needed to transform the quantity.

The agreement reached in this project is to reference all the measurements of solid ingredients to **kilograms**, and the liquids to **liters**.

The most complex issue is that several countries use identical measuring names but with different meaning (use the same measuring device, but with different sizes). For example, one cup has different meaning depending on the country it is referred from: U.S. = 237 milliliters, U.K. = 284 milliliters and Australia = 250 milliliters

We need to know the place where the recipe is from and make the conversion to is respective measure (which is a very complex task) or state a standard and refer always to it. In this project the measurement system selected as reference is Metric, as it is the standard in Europe.

The next tables show the conversions between the U.S, U.K and Australian Measures:

US Standard to Australian Measurement Conversions

U.S. Standard	To	Australian
1 cup	.....	**3/4 cup and 2 tbsp and 1 dsp**
3/4 cup	.....	**2/3 cup plus 1 dsp**
2/3 cup	.....	**1/2 cup plus 3 dsp**
1/2 cup	.....	**1/3 cup plus 2 tbsp**
1/3 cup	.....	**1/4 cup plus 1 tbsp**
1/4 cup	.....	**3 tbsp**
1 tbsp	.....	**1 dsp plus 1 tsp**
1 tsp	.....	**1 tsp**
3/4 tsp	.....	**3/4 tsp**
1/2 tsp	.....	**1/2 tsp**
1/4 tsp	.....	**1/4 tsp**

US Standard to UK Measurement Conversions

U.S. Standard	To	U.K.
1 cup	.....	**3/4 cup and 2 dsp**

3/4 cup	.....	1/2 cup plus 2 tbsp
2/3 cup	.....	1/2 cup plus 1 tbsp
1/2 cup	.....	1/3 cup plus 2 dsp
1/3 cup	.....	1/4 cup plus 1 tsp
1/4 cup	.....	1/4 cup minus 1 dsp
1 tbsp	.....	2 1/2 tsp
1 tsp	.....	3/4 tsp (slightly rounded)
3/4 tsp	.....	1/2 tsp (rounded)
1/2 tsp	.....	1/2 tsp (scant)
1/4 tsp	.....	1/4 tsp (scant)

Also the temperature measure differs from some countries to others. The two temperature scales (Fahrenheit and Celsius grades) are compared in the next table

TEMPERATURE CONVERSIONS		
Fahrenheit	To	Celsius
32 F	.....	0 C
100 F	.....	40 C
125 F	.....	50 C
140 F	.....	60 C
150 F	.....	65 C
160 F	.....	70 C
175 F	.....	80 C
180 F	.....	82 C
200 F	.....	95 C

Some of this information has been retrieved from: http://allrecipes.com/advice/ref/conv_ and http://info.pue.udlap.mx/gente/ua012785/cursos/doc5.doc

APPENDIX_21 MEASUREMENTS CONVERSIONS

```
<!--
     ================================================================
     WebODE's DTD 1.0
     ================================================================
     This document type definition describes the format of WebODE's
     export/import files.

     These files may also be used to interoperate with any other
     application supporting XML and a validating XML parser.

     The latest version of this DTD can be found at
     http://babage.dia.fi.upm.es/webode/DTD/webode_1_0.dtd.

     (c) Artificial Intelligence Lab, 2000.
     School of Computer Science (FI).  Technical University of Madrid.
     ================================================================
-->

<!-- The root for the document -->
<!ELEMENT Ontology (Name, Namespace?, Description?, Author?, Creation-
Date,
                   Related-Reference*, Conceptualization?,
                Instances?, Views?)>

<!-- General elements used throughout the document -->
<!ELEMENT Name                 (#PCDATA)>
<!ELEMENT Namespace            (#PCDATA)>
<!ELEMENT namespace            (#PCDATA)>
<!ELEMENT namespace_identifier (#PCDATA)>
<!ELEMENT Original-Name        (#PCDATA)>
<!ELEMENT Description          (#PCDATA)>
<!ELEMENT Type                 (#PCDATA)>
<!ELEMENT Minimum-Cardinality  (#PCDATA)>
<!ELEMENT Maximum-Cardinality  (#PCDATA)>
<!ELEMENT Minimum-Value        (#PCDATA)>
<!ELEMENT Maximum-Value        (#PCDATA)>
<!ELEMENT Measurement-Unit     (#PCDATA)>
<!ELEMENT Related-Reference    (#PCDATA)>
<!ELEMENT Precision            (#PCDATA)>
<!ELEMENT Related-Formula      (#PCDATA)>
<!ELEMENT Related-Property     (#PCDATA)>
<!ELEMENT Related-Concept      (#PCDATA)>
<!ELEMENT Value                (#PCDATA)>
<!ELEMENT Author               (#PCDATA)>
<!ELEMENT Creation-Date        (#PCDATA)>
<!ELEMENT Origin               (#PCDATA)>
<!ELEMENT Destination          (#PCDATA)>
<!ELEMENT Expression           (#PCDATA)>
<!ELEMENT URL                  (#PCDATA)>
<!ELEMENT URI                  (#PCDATA)>
<!ELEMENT Inferred             (#PCDATA)>
```

```
<!ELEMENT Instance-of          (#PCDATA)>
<!ELEMENT Origin-Concept       (#PCDATA)>
<!ELEMENT Destination-Concept  (#PCDATA)>
<!ELEMENT X                    (#PCDATA)>
<!ELEMENT Y                    (#PCDATA)>


<!-- Conceptualization issues -->
<!ELEMENT Conceptualization (Imported-Term*, Reference*, Concept*,
                             Group*, Term-Relation*, Formula*,
                        Constant*, Property*)>

<!-- Term -->
<!ELEMENT Concept (Name, Description?, Class-Attribute*, Instance-
Attribute*,
                Synonym*, Abbreviation*, Related-Reference*, Related-
Formula*)>

<!-- Class attribute -->
<!ELEMENT Class-Attribute (Name, Description?, Type, Minimum-
Cardinality,
                           Maximum-Cardinality, Measurement-Unit?,
Precision?,
                      Value*, Related-Reference*, Synonym*,
Abbreviation*,
                      Related-Formula*, Inferred*)>

<!-- Instance attribute -->
<!ELEMENT Instance-Attribute (Name, Description?, Type, Minimum-
Cardinality,
                              Maximum-Cardinality, Measurement-Unit?,
                              Precision?, Minimum-Value?, Maximum-
Value?, Value*,
                         Related-Reference*, Synonym*, Abbreviation*,
                         Related-Formula*, Inferred*)>

<!-- Synonym -->
<!ELEMENT Synonym (Name, Description?)>

<!-- Abbreviation -->
<!ELEMENT Abbreviation (Name, Description?)>

<!-- Constant -->
<!ELEMENT Constant (Name, Description?, Type, Value, Measurement-Unit)>

<!-- Relations -->
<!ELEMENT Term-Relation (Name, Description?, Origin, Destination,
                         Maximum-Cardinality, Related-Reference*,
Related-Property*)>

<!-- Formulas -->
<!ELEMENT Formula (Name, Description?, Type, Expression, Related-
Reference*)>

<!-- Properties -->
<!ELEMENT Property (Name, Description?, Related-Reference*, Related-
Formula*)>
```

```
<!-- References -->
<!ELEMENT Reference (Name, Description?)>

<!-- Imported terms -->
<!ELEMENT Imported-Term (URI, namespace, namespace_identifier)>

<!-- Groups -->
<!ELEMENT Group (Name, Description?, Related-Concept+)>

<!-- Instances -->
<!ELEMENT Instances (Instance-Set*)>

<!ELEMENT Instance-Set (Name, Description?, Instance*, Relation-
Instance*)>
<!ELEMENT Instance (Name, Instance-of, Description?, Class*)>
<!ELEMENT Class (Name, Attribute*)>
<!ELEMENT Attribute (Name, Value)>
<!ELEMENT Relation-Instance (Name, Instance-of, Origin-Concept?,
Destination-Concept?, Description?, Origin, Destination)>
<!-- Angel will change the Origin-Concept? and Destination-Concept?
once all XML files have been generated correctly -->

<!-- Views -->
<!ELEMENT Views (View*)>

<!ELEMENT View (Name, View-Element*, View-Relation*)>
<!ELEMENT View-Element  (Name, X , Y)>
<!ELEMENT View-Relation (Name, Origin, Destination, X, Y)>
```

APPENDIX_22 Brief example of how to query the system

The following queries constitute an example of how to query the knowledge base to get the desired information. These queries have been made using the AG Quip free-ware source.

22.1 Query example 1 – Number of recipes that compose the D.B.

Query the number of recipes

```
<IngredientsRecipe>
{
for $r in document("XMLExport[1].xml")/Ontology/Instances
return
<number_of_recipes_in_database>
        {count($r/Instance-Set)}
</number_of_recipes_in_database>
}
</IngredientsRecipe>
```

Example of the XML result

```
<?xml version="1.0"  encoding="UTF-8" ?>

<quip:result xmlns:quip = "http://namespaces.softwareag.com/tamino/quip/">

  <IngredientsRecipe>
        <number_of_recipes_in_database>7</number_of_recipes_in_database>
</IngredientsRecipe>

</quip:result>
```

22.2 Query example 2 – Query all the recipes' names

Query the titles of all the recipes

```
for $a in document("XMLExport[1].xml")//Ontology
return
<Query>
        List all the recipes that make up the knowledge-base
        {
                <Recipes>
                        {$a/Instances/Instance-Set/Name}
                </Recipes>
        }
</Query>
```

Example of the XML result

```
<?xml version="1.0"  encoding="UTF-8" ?>

<quip:result xmlns:quip = "http://namespaces.softwareag.com/tamino/quip/">

 <Query>

        List all the recipes that make up the knowledge-base

<Recipes>

        <Name>almond chicken</Name>
        <Name>baked_spinach_with_cheese</Name>
        <Name>beef hamburger</Name>
        <Name>chocolate cake</Name>
        <Name>easy lasagna sauce</Name>
        <Name>paella</Name>
        <Name>spaghetti carbonara</Name>

</Recipes>

</Query>

</quip:result>
```

22.3 Query example 3 – Query the ingredients of a recipe

Query the ingredients of a recipe

```
<results>
        {
                for $instance in
document("XMLExport[1].xml")/Ontology/Instances/Instance-Set/Instance,
                    $recipe in document("XMLExport[1].xml")/Ontology/Instances/Instance-
Set
                where $instance/Instance-of = "ingredient" and $recipe/Name =
"baked_spinach_with_cheese" and $instance = $recipe/Instance
                return
                        <ingredient>
                        {$instance/Name/text()}
                        </ingredient>
        }
</results>
```

Example of the XML result

```
<?xml version="1.0"  encoding="UTF-8" ?>

<quip:result xmlns:quip = "http://namespaces.softwareag.com/tamino/quip/">

  <results>

        <ingredient>spinach</ingredient>
        <ingredient>garlic</ingredient>
        <ingredient>onion</ingredient>
        <ingredient>salt</ingredient>
        <ingredient>shrimp</ingredient>
        <ingredient>cream</ingredient>

</results>

</quip:result>
```

APPENDIX_23 Examples of a XML auxiliary files

The first file show how the normalization of the Ontology can be made by means of external files.It refers to the measurement units. The other files show how the XML can be used to give additional information to the Ontology, the example refers to the vitamins.

23.1 Measures conversion file

Measures conversion skeleton (XMLSchema)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
leticia (my office) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="measure_units_conversion_table">
            <xs:annotation>
                  <xs:documentation>Comment describing your root
element</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="weight_measure">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="kilo">
                                                <xs:complexType>
                                                      <xs:sequence>
                                                            <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                            <xs:element
ref="conversion_to_kilo"/>
                                                      </xs:sequence>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="gram">
                                                <xs:complexType>
                                                      <xs:sequence>
                                                            <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                            <xs:element
ref="conversion_to_kilo"/>
                                                      </xs:sequence>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="milligram">
                                                <xs:complexType>
                                                      <xs:sequence>
                                                            <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                            <xs:element
ref="conversion_to_kilo"/>
                                                      </xs:sequence>
```

```
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element name="pound">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_kilo"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element name="ounce">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_kilo"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                          </xs:sequence>
                                    </xs:complexType>
                              </xs:element>
                              <xs:element name="liquid_measure">
                                    <xs:complexType>
                                          <xs:sequence>
                                                <xs:element name="liter">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element name="deciliter">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element
name="centiliter">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
```

```
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element
name="milliliter">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element
name="fluid_ounce">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                                <xs:element name="gallon">
                                                      <xs:complexType>
                                                            <xs:sequence>
                                                                  <xs:element
ref="synonym" maxOccurs="unbounded"/>
                                                                  <xs:element
ref="conversion_to_liter"/>
                                                            </xs:sequence>
                                                      </xs:complexType>
                                                </xs:element>
                                          </xs:sequence>
                                    </xs:complexType>
                              </xs:element>
                        </xs:sequence>
                  </xs:complexType>
            </xs:element>
            <xs:element name="synonym" type="xs:string"/>
            <xs:element name="conversion_to_liter" type="xs:decimal"/>
            <xs:element name="conversion_to_kilo" type="xs:decimal"/>
</xs:schema>
```

Measures conversion file (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com)-->
<quantity_conversion_table xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="C:\Documents and
Settings\s021450\My Documents\Project\XML sources\XML
projects\Project3\03-04-2004MeasuresTaxonmy.xsd">
            <0.5>
                  <synonym>half</synonym>
```

```
                  <synonym>1/2</synonym>
                  <conversion_to_integer>0.5</conversion_to_integer>
            </0.5>
            <0.75>
                  <synonym>three quarters</synonym>
                  <synonym>3/4</synonym>
                  <conversion_to_integer>0.75</conversion_to_integer>
            </0.75>
            <1>
                  <synonym>one</synonym>
                  <synonym>a</synonym>
                  <conversion_to_integer>1</conversion_to_integer>
            </1>
            <1.5>
                  <synonym>one and a half</synonym>
                  <synonym>1 1/2</synonym>
                  <synonym>1 and a half</synonym>
                  <conversion_to_integer>1.5</conversion_to_integer>
            </1.5>
            <2>
                  <synonym>two</synonym>
                  <synonym>pair</synonym>
                  <synonym>couple</synonym>
                  <conversion_to_kilo>2</conversion_to_kilo>
            </2>
            <3>
                  <synonym>three</synonym>
                  <conversion_to_kilo>3</conversion_to_kilo>
            </3>
                        <2>
                  <synonym>two</synonym>
                  <synonym>pair</synonym>
                  <synonym>couple</synonym>
                  <conversion_to_kilo>2</conversion_to_kilo>
            </2>
</quantity_conversion_table>
```

23.2 Ingredients and their vitamins auxiliary file

Vitamins and ingredients skeleton (XMLSchema)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
leticia (none) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<!--  <xs:include schemaLocation="E:\My Pictures\Tivoli\Picture005.jpg"
xsi:type="gif"/> -->
            <!--  <xs:image format="image/jpeg" url="E:\My
Pictures\Tivoli\Picture005.jpg" /> -->

      <xs:element name="vitaminDescription">
            <xs:annotation>
                  <xs:documentation>vitamin auxiliary
classification</xs:documentation>
```

```
            </xs:annotation>
            <xs:complexType mixed="false">
                  <xs:sequence maxOccurs="unbounded">
                        <xs:element name="vitamin">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element
name="vitamin_name" type="xs:string"/>
                                          <xs:choice>
                                                <xs:element
name="fat_soluble">
                                                      <xs:complexType/>
                                                </xs:element>
                                                <xs:element
name="water_soluble">
                                                      <xs:complexType/>
                                                </xs:element>
                                          </xs:choice>
                                          <xs:element name="found_in"
minOccurs="0">
                                                <xs:complexType>
                                                      <xs:sequence
minOccurs="0" maxOccurs="unbounded">
                                                            <xs:element
name="ingredient" type="xs:string" minOccurs="0"/>
                                                      </xs:sequence>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                              </xs:complexType>
                        </xs:element>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Vitamins and ingredients file (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
leticia (none) -->
<!--Sample XML file generated by XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com)-->
<vitaminDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="C:\Documents and
Settings\s021450\My Documents\Project\XML sources\XML
projects\Project3\vitaminDescription.xsd">
      <vitamin>
            <vitamin_name>A</vitamin_name>
            <fat_soluble/>
            <found_in>
                  <ingredient>eggs</ingredient>
                  <ingredient>milk</ingredient>
                  <ingredient>cheese</ingredient>
```

```
                <ingredient>liver</ingredient>
                <ingredient>carrotr</ingredient>
                <ingredient>potatoe</ingredient>
                <ingredient>spinach</ingredient>
                <ingredient>mango</ingredient>
                <ingredient>apricot</ingredient>
                <ingredient>oatmeal</ingredient>
                <ingredient>pepper</ingredient>
                <ingredient>peach</ingredient>
                <ingredient>peas</ingredient>
                <ingredient>papaya</ingredient>
        </found_in>
    </vitamin>
    <vitamin>
        <vitamin_name>D</vitamin_name>
        <fat_soluble/>
        <found_in>
                <ingredient>cod liver</ingredient>
                <ingredient>beef liver</ingredient>
                <ingredient>salmon</ingredient>
                <ingredient>mackerel</ingredient>
                <ingredient>sardines</ingredient>
                <ingredient>eel</ingredient>
                <ingredient>milk</ingredient>
                <ingredient>margarine</ingredient>
                <ingredient>cereal</ingredient>
                <ingredient>String</ingredient>
        </found_in>
    </vitamin>
    <vitamin>
        <vitamin_name>E</vitamin_name>
        <fat_soluble/>
        <found_in>
                <ingredient>wheat</ingredient>
                <ingredient>almonds</ingredient>
                <ingredient>corn</ingredient>
                <ingredient>mango</ingredient>
                <ingredient>peanuts</ingredient>
                <ingredient>nuts</ingredient>
                <ingredient>mayonnaise</ingredient>
                <ingredient>broccoli</ingredient>
                <ingredient>pistachio</ingredient>
                <ingredient>spinach</ingredient>
                <ingredient>kiwi</ingredient>
        </found_in>
    </vitamin>
    <vitamin>
        <vitamin_name>K</vitamin_name>
        <fat_soluble/>
        <found_in>
                <ingredient>broccoli</ingredient>
                <ingredient>spinach</ingredient>
                <ingredient>vegetable</ingredient>
                <ingredient>pork</ingredient>
                <ingredient>cheese</ingredient>
        </found_in>
    </vitamin>
```

```
	<vitamin>
		<vitamin_name>B1</vitamin_name>
		<water_soluble/>
		<found_in>
			<ingredient>pork</ingredient>
			<ingredient>vegetable</ingredient>
			<ingredient>milk</ingredient>
			<ingredient>cheese</ingredient>
			<ingredient>peas</ingredient>
			<ingredient>fresh </ingredient>
			<ingredient>dried fruit</ingredient>
			<ingredient>egg</ingredient>
			<ingredient>bread</ingredient>
			<ingredient> cereal</ingredient>
		</found_in>
	</vitamin>
	<vitamin>
		<vitamin_name>B2</vitamin_name>
		<synonym>riboflavin</synonym>
		<water_soluble/>
		<found_in>
			<ingredient>milk</ingredient>
			<ingredient>egg</ingredient>
			<ingredient>cereal</ingredient>
			<ingredient>rice</ingredient>
			<ingredient>mushrooms</ingredient>
		</found_in>
	</vitamin>
	<vitamin>
		<vitamin_name>B3</vitamin_name>
		<water_soluble/>
		<synonym>niacin</synonym>
		<synonym>niacinamide</synonym>
		<synonym>nicotinic acid</synonym>
		<found_in>
			<ingredient>beef</ingredient>
			<ingredient>pork</ingredient>
			<ingredient>chicken</ingredient>
			<ingredient>wheat flour</ingredient>
			<ingredient>maize flour</ingredient>
			<ingredient>egg</ingredient>
			<ingredient>milk</ingredient>
		</found_in>
	</vitamin>
	<vitamin>
		<vitamin_name>B5</vitamin_name>
		<water_soluble/>
		<synonym>pantothenic acid</synonym>
		<found_in>
			<ingredient>beef</ingredient>
			<ingredient>potatoes</ingredient>
			<ingredient>porridge</ingredient>
			<ingredient>tomatoes</ingredient>
			<ingredient>liver</ingredient>
			<ingredient>kidney</ingredient>
			<ingredient>yeast</ingredient>
			<ingredient>eggs</ingredient>
```

```
                <ingredient>broccoli</ingredient>
                <ingredient>brown rice </ingredient>
                <ingredient>wholemeal bread</ingredient>
          </found_in>
    </vitamin>
    <vitamin>
          <vitamin_name>B6</vitamin_name>
          <water_soluble/>
          <found_in>
                <ingredient>cereal</ingredient>
                <ingredient>potatoe</ingredient>
                <ingredient>banana</ingredient>
                <ingredient>chickpea</ingredient>
                <ingredient>chicken</ingredient>
                <ingredient>oatmeal</ingredient>
                <ingredient>pork</ingredient>
                <ingredient>beef</ingredient>
                <ingredient>sunflower seed</ingredient>
                <ingredient>oatmeal</ingredient>
                <ingredient>spinach</ingredient>
                <ingredient>tomatoe</ingredient>
                <ingredient>avocado</ingredient>
                <ingredient>salmon</ingredient>
                <ingredient>tuna</ingredient>
                <ingredient>peanut</ingredient>
          </found_in>
    </vitamin>
    <vitamin>
          <vitamin_name>H</vitamin_name>
          <water_soluble/>
          <synonym>biotin</synonym>
          <found_in>
                <ingredient>kidney</ingredient>
                <ingredient>liver</ingredient>
                <ingredient>egg</ingredient>
                <ingredient>fruit</ingredient>
                <ingredient>vegetables</ingredient>
          </found_in>
    </vitamin>
    <vitamin>
          <vitamin_name>Folic acid</vitamin_name>
          <water_soluble/>
          <found_in>
                <ingredient>broccoli</ingredient>
                <ingredient>brussels sprouts</ingredient>
                <ingredient>peas</ingredient>
                <ingredient>chickpeas</ingredient>
                <ingredient>yeast</ingredient>
                <ingredient>brown rice </ingredient>
                <ingredient>orange</ingredient>
                <ingredient>banana</ingredient>
                <ingredient>cereal</ingredient>
                <ingredient>bread</ingredient>
          </found_in>
    </vitamin>
    <vitamin>
          <vitamin_name>B12</vitamin_name>
```

```
            <water_soluble/>
            <found_in>
                  <ingredient>liver</ingredient>
                  <ingredient>cereal</ingredient>
                  <ingredient>salmon</ingredient>
                  <ingredient>beef</ingredient>
                  <ingredient>haddock</ingredient>
                  <ingredient>clam</ingredient>
                  <ingredient>oyster</ingredient>
                  <ingredient>tuna</ingredient>
                  <ingredient>milk</ingredient>
                  <ingredient>yogurt</ingredient>
                  <ingredient>pork</ingredient>
                  <ingredient>egg</ingredient>
                  <ingredient>cheese</ingredient>
                  <ingredient>chicken</ingredient>
            </found_in>
      </vitamin>
      <vitamin>
            <vitamin_name>C</vitamin_name>
            <synomym>ascorbic acid</synonym>
            <water_soluble/>
            <found_in>
                  <ingredient>pepper</ingredient>
                  <ingredient>brocoli</ingredient>
                  <ingredient>brussels sprout</ingredient>
                  <ingredient>sweet potatoe</ingredient>
                  <ingredient>orange</ingredient>
                  <ingredient>kiwi</ingredient>
            </found_in>
      </vitamin>
</vitaminDescription>
```

23.3 Vitamin advices auxiliary file

Vitamin advices skeleton (XMLSchema)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
leticia (my office) -->
<!--W3C Schema generated by XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
      <xs:element name="advices">
            <xs:complexType mixed="true"/>
      </xs:element>
      <xs:element name="fat_soluble">
            <xs:complexType mixed="true">
                  <xs:sequence>
                        <xs:element ref="advices"/>
                  </xs:sequence>
            </xs:complexType>
```

```
      </xs:element>
      <xs:element name="vitamin_properties">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="fat_soluble"/>
                        <xs:element ref="water_soluble"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="water_soluble">
            <xs:complexType mixed="true">
                  <xs:sequence>
                        <xs:element ref="advices"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Vitamin advices file (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 3 U
(http://www.xmlspy.com)-->
<vitaminDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="C:\Documents and
Settings\s021450\My Documents\Project\XML sources\XML
projects\Project3\vitaminDescription2.xsd">

<vitamin_properties>
<fat_soluble>can not be dissolved in water
            <advices>
                  Have to be ingested with the fat of some food, in
order to assimilate its nutrients
                  If it is ingested as a pill they have to be eaten
with some fatty-food
                  Not to be taken on an empty stomach
                  If taken in big quantity is ingested can be toxic
            </advices>
      </fat_soluble>
      <water_soluble>can be dissolved in water
            <advices>
                  They are lost when the aliments are cooked.
                  They are not toxic in big amounts due to they are
disposed of the organism
                  They should be ingested almos daily
            </advices>
      </water_soluble>
</vitamin_properties>
```

APPENDIX_24 Comparative of different semistructured web languages files

Dimension	XML (Schema)	RDF (Schema)	DAML	Notes
Contexts	Default Namespace : xmlns Declaring others : xmlns:<label> *XMLSchema Namespace (typically labelled xsd)* xmlns:xsd="www.w3.org/2001/XMLSchema" [Final W3C Recommendation] *targetNamespace* refers to the namespace defined by the current file. Example Namespace Declarations **Note:** Namespaces need not point to anything in the XML Namespaces specification. <xsi:schemaLocation..> provides the location of the schema.	RDF uses XML Namespaces. *RDF Syntax Namespace* xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" *RDF Schema Namespace* xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#" Example Namespace Declarations in RDF **Note**: In RDF, the namespace URI reference also identifies the location of the RDF schema	DAML also uses XML Namespaces. It uses RDF & RDF Schema elements by referring to their respective Namespaces. The latest *DAML Namespace* xmlns:daml="http://www.daml.org/2001/03/daml+oil#" Example Namespace Declarations in DAML In DAML, we have to Import ontologies to be able to use the classes defined in the ontology.	
Object Classes & Properties	No concept of classes and properties, only Elements of certain Types. The Type can be *simpleType* or a *complexType*. (A class could be any element & its property could be its child element - NO DEFINED SEMANTICS)	Resource is the top level class. (http://www.w3.org/2001/01/rdf-schema#Resource) Example of Classes, & Properties Cycles in class hierarchy were not allowed till a little while ago. Latest revisions to the RDF spec allow this.	DAML also has Classes & Properties. Classes can also be a subClassOf an anonymous class created due to a '*Restriction*' on the set of all 'Things'. Two kinds of Properties are defined: *ObjectProperties* (Relate an object to another object - the value of the property is also an object) & *DatatypeProperties* (Relate an object to a primitive data type - the value of the property is a primitive data	

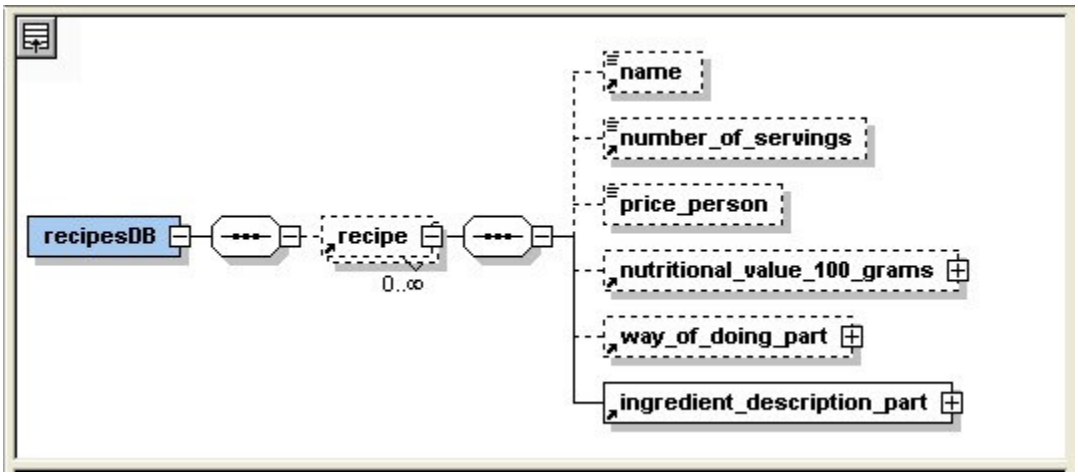
			type) Cycles in the class hierarchy are allowed.	
Inheritance	Since there aren't any Classes, there is no concept of inheritance. However, Types can be *extended* or *restricted*, thus defining subTypes.	A class can be a *subClassOf* other classes (multiple inheritance is allowed) Properties can also be *subPropertyOf* other properties.	Same as in RDF.	
Property / Element Range	Can be specified globally. For a locally specificied (element specific) range, the element has to be declared locally. Example of a global element. The global element can be referred to like this. Example of local element	Can only be specified globally <rdfs:range....> Multiple range statements imply conjunction (i.e. all of them should be satisfied) Example of a range specification	Can be specified globally (<rdfs:range...>) as well as locally <daml:Restriction> <daml:onProperty...><daml:toClass....> </daml:Restriction> Multiple range statements imply conjunction (i.e. all of them should be satisfied)	Range specifies the kinds of values (elements / classes / datatypes) the property can have.
Property / Element Domain	No explicit declaration of the domain of the element. The domain is implicitly the element in which the definition appears.	Can only be specified globally <rdfs:domain....> Multiple domain statements imply conjunction (i.e. all of them should be satisfied) Example of a domain in RDF here.	Can be specified globally (<rdfs:domain...>). Multiple domain statements imply conjuction (i.e. all of them should be satisfied) Example of a domain in DAML	Domain specifies which elements / classes can have the particular property
Property / Element Cardinality	Can be specified using *minOccurs, maxOccurs* Example	Not defined in the core RDF Schema. By default there are no cardinality restrictions on properties. However, new constraints like these can be	Can be specified locally *minCardinality*, *maxCardinality*, *cardinality* Can also be specified globally although only as a UniqueProperty (single valued i.e. having cardinality of	The cardinality of a property specifies the number of occurences of

		specified by making them a *subClassOf* the '*ConstraintProperty*' Class. Look at how cardinality is defined in OIL-Ontology	1).	the property/element for a certain class/element.
Basic Datatypes	Datatypes supported by XMLSchema are mainly variations of numerical, temporal and string datatypes. For a hierarchy of all built-in datatypes visit this link.	The core RDF Schema only includes 'Literals' which is the set of all strings. Example of the use of Literals. (The latest RDF spec is expected to include XMLSchema datatypes)	Allows the use of XMLSchema Datatypes by just referring to the XMLSchema URI. Example More information can be found here.	
Enumeration of Property Values	Possible with the <*enumeration*> tag. See an example here.	Not possible.	Enumeration of property types is possible with the <*oneOf rdf:ParseType="daml:collection"...*> tag See an example here. It is also possible to simply point to an enumerated data type declared using XMLSchema.	An Enumeration restricts the value space of a property to a certain set of values.
Ordered Data Set	Data Sets maintain order by default **Note:** Type declarations having the <*sequence*> tag, just mean that data should appear in the order specified Example	Data Set ordered with the <*rdf:Seq...*> tag Example can be found here.	Can use the <*daml:list*> tag	
Bounded Lists	Not possible to specify.	Not possible to specify.	Possible with the <*daml:collection*> tag	
Transitive Properties	Not possible to specify.	Cannot be stated.	Possible with the <*daml:TransitiveProperty*> tag See an example.	
Negation	Not possible.	Not present.	Possible with the <*daml:complementOf..*> tag	Negation implies the

			Example	absence of some element (ex. no Car is a person)
Disjunctive / Disjoint Classes	A union of possible types for an element is possible with the <*union*> tag Example	Can use a Bag to Indicate unordered collections (or unions of properties). However, we cannot have a class as a union of 2 classes.	A Class can be a union of 2 other Classes. Possible with the <*unionOf...*> tag. We can represent disjoint unions with the <*disjointUnionOf...*> tag. Example	
Necessary & Sufficient conditions for Class Membership	No (although <unique> could be interpreted as an UnambigousProperty)	No	Yes *sameClassAs*, *equivalent,* using boolean combinations of Class Expressions. *UnambigousProperty* specifies a property which identifies the resource (like a primary key).	Necessary and Sufficient conditions for Class Membership specify a class definition that can be used: 1) to determine (or recognize) whether an instance belongs to that class 2) to determine (or classify) whetheher the class is a subclass of another class.

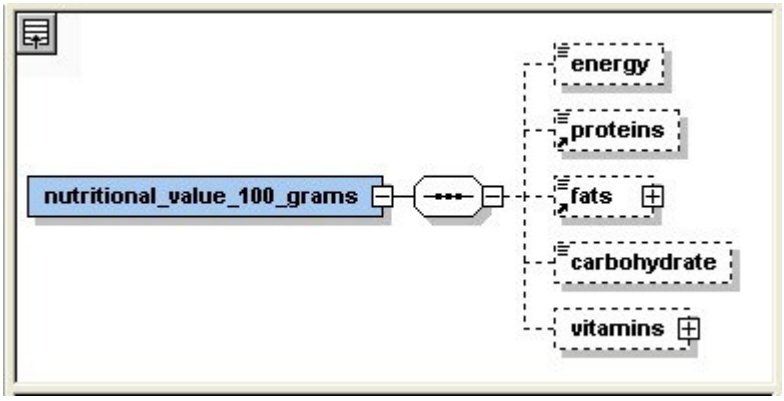
APPENDIX_25 Comparative of different semistructured web languages files

We have a principal element called recipesDB,which has 0..n recipe children



Each recipe has a number of nested elements:

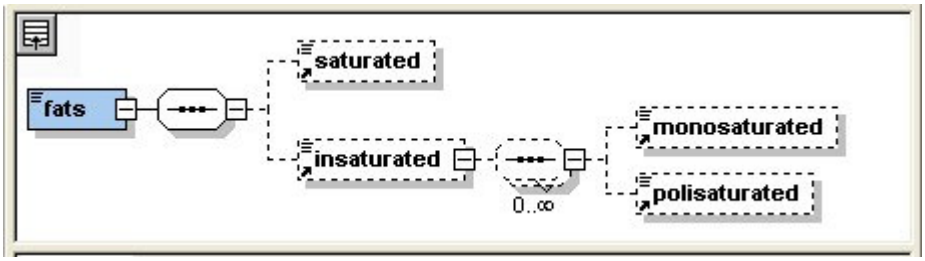
- **name**: name of the recipe: optional
- **number_of_servings**: the number of people that recipe is for: optional
- **price_person**: cost of the recipe per person: optional
- **nutritional_value_100_grams**: the nutritional value of the recipe, it has complex content, the next children elements:



  - **energy**: per 100 grams of dish. It is optional within the nutritional value, but if exists, it has the following fields.:
    <energy measure="String">0</energy>
    Its content is complex and has:

- Its value showing the **quantity of energy** and it is of an integer type.
- An attribute **"measure"** showing the kind of energy.
  For example: If it is measured in kj or kcal: its type is a string and it is required because we need to know the measure of the energy, in order to work with it.

- **proteins**: number of proteins per 100 grams. It has a decimal value and it is optional.
- **fats**: It is a complex entity, with a single value and also a sequence of children. It is optional as well as its siblings.

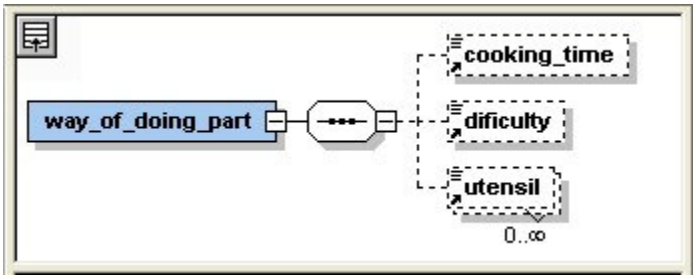


- **saturated**: Number of saturated fats among the total fats. It is also optional. It has a decimal value.
- **insaturated**: Number of saturated fats among the total fats. It is also optional. It has a decimal value, and also a sequence of children:
  - **monosaturated:** Number of monosaturated fats among the insaturated fats. It is also optional. It has a decimal value.
  - **polisaturated:** Number of polisaturated fats among the insaturated fats. It is also optional. It has a decimal value.

- **carbohydrate**: number of carbohydrate per 100 grams. It has a decimal value and it is optional.
- **vitamins**: It has a complex value. It is a sequence of the vitamins of the recipe. The list can have from 1 to n vitamin.
  - **vitamin**: Unique child entity of vitamins. At least one occurrence and at most 13 (nowadays the experts only know 13 vitamins). Each one has a complex value, with two childs:
    - **name**: of the vitamin (A, B, C, etc). String value. Non optional. 1 and only 1 ocurrence.
    - **percentage_DV**: Percentage of the daily value. This is the recommended quantity of each vitamin that we

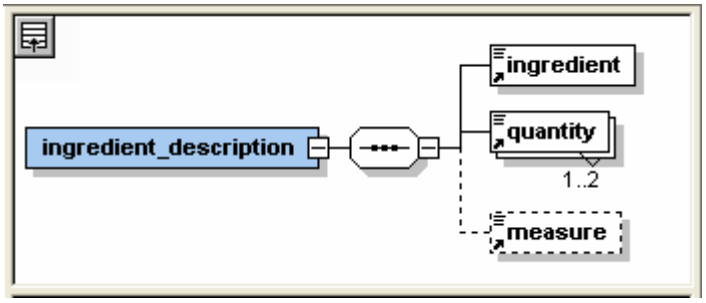
should consume daile. It is optional. And has a simple decimal value.

- **way_of_doing_part**: This is the part in the recipe where all the cooking process is explained. It is a complex element with a sequence of three children. It is optional. In fact everything in the recipe element is optional but the description of the ingredients, because in some recipes some of these parts can be missing, but not the ingredient one, which is mandatory (as long as I have seen in all the recipes I have studied)
  Its three children are:



  - **cooking_time**: Optional. It is complex. It has a string content that shows the quantity of time, and a **time_measure** attribute that describes the time it is measured in, seconds, hours, etc. This attribute is mandatory because it is not useful the time without the measure.
  - **difficulty**: Optional. It has a simple string value, which describes the difficulty of the recipe. E.g: simple, medium, high, low….etc.
  - **utensil:** List of the utensils used to make the dish. Can be 0 or more (unbounded). Each utensil has a complex value. With a string type that has the name of the utensil and a **cooking_purpose** attribute, which is an optional Boolean. If its value is true it means that the utensil is a cooking utensil (like oven, frying pan, etc) if it is false then it is a pre-cooking utensil (like plate, knife, fork, etc). This attribute is used afterwards to sort the recipes and make advanced queries.
- **ingredient_description_ part**: This is the only mandatory section of the recipe. It has a minimum and a maximum value of 1. It is a complex entity that has a list of ingredient_description elements (1..n)

  - **ingredient_description:** complex entity that describes the occurrence of each entity in the recipe, through its three nested elements:



- **ingredient:** Mandatory element. With a simple string value. It can only appear once in the ingredient description. This is the name of the ingredient in the recipe.
- **quantity:** Mandatory simple element with an integer value. It represents the amount of the above ingredient that is used in the recipe. Each ingredient description can contain one or two quantities. For example: 5 grams of yeast, or 5 or 6 tomatoes.
- **measure:** Optional (if exists can be only one per ingredient description) complex element with a string value, which represents the measure of the ingredient, like: grams, kilograms, liters, cups, etc.
  It also has a Boolean attribute regular, that indicates if the measure is a regular measure (like, gram, kilogram, liter, deciliter, etc) or if it is a non-regular measure (cup, spoon, teaspoon, etc)

XML RecipesDB Schema Example

```
?xml version="1.0" encoding="UTF-8" ?>
<!--
edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
 leticia (my office)
 -->
<!--
3C Schema generated by XMLSPY v2004 rel. 3 U
 (http://www.xmlspy.com)
 -->
xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
xs:element name="name" type="xs:string" />
xs:element name="price_person">
xs:simpleType>
xs:restriction base="xs:float">
xs:minExclusive value="0" />
 </xs:restriction>
 </xs:simpleType>
```

```
</xs:element>
xs:element name="nutritional_value_100_grams">
xs:complexType>
xs:sequence>
xs:element name="energy" minOccurs="0">
xs:complexType>
xs:simpleContent>
xs:extension base="xs:integer">
xs:attribute name="measure" type="xs:string" use="required" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
xs:element ref="proteins" minOccurs="0" />
xs:element ref="fats" minOccurs="0" />
xs:element name="carbohydrate" type="xs:decimal" minOccurs="0" />
xs:element name="vitamins" minOccurs="0">
xs:complexType mixed="false">
xs:sequence>
xs:element name="vitamin" maxOccurs="unbounded">
xs:complexType mixed="false">
xs:sequence>
xs:element ref="name" />
xs:element name="percentage_CDR" type="xs:decimal" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
xs:element name="kcal">
xs:complexType />
</xs:element>
xs:element name="proteins" type="xs:decimal" />
xs:element name="vitamins" type="xs:decimal" />
xs:element name="fats">
xs:complexType mixed="true">
xs:sequence>
xs:element ref="saturated" minOccurs="0" />
xs:element ref="insaturated" minOccurs="0" />
```

```
</xs:sequence>
</xs:complexType>
</xs:element>
xs:element name="ingredient_description_part">
xs:complexType>
xs:sequence>
xs:element ref="ingredient_description" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
xs:element name="way_of_doing_part">
xs:complexType>
xs:sequence>
xs:element ref="cooking_time" minOccurs="0" />
xs:element ref="dificulty" minOccurs="0" />
xs:element ref="utensil" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
xs:element name="cooking">
xs:complexType />
</xs:element>
xs:element name="cooking_time">
xs:complexType>
xs:simpleContent>
xs:extension base="xs:string">
xs:attribute name="time_measure" type="xs:time" use="required" />
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
xs:element name="dificulty" type="xs:string" />
xs:element name="ingredient" type="xs:string" />
xs:element name="ingredient_description">
xs:complexType>
xs:sequence>
xs:element ref="ingredient" />
xs:element ref="quantity" maxOccurs="2" />
xs:element ref="measure" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
xs:element name="measure">
```

```
xs:complexType>
xs:simpleContent>
xs:extension base="xs:string">
xs:attribute name="regular" type="xs:boolean" use="optional" />
 </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
 </xs:element>
xs:element name="number_of_servings" type="xs:integer" />
xs:element name="pre_cooking">
xs:complexType />
 </xs:element>
xs:element name="quantity" type="xs:integer" />
xs:element name="recipe">
xs:complexType>
xs:sequence>
xs:element ref="name" minOccurs="0" />
xs:element ref="number_of_servings" minOccurs="0" />
xs:element name="price_person" minOccurs="0">
xs:complexType>
xs:simpleContent>
xs:extension base="xs:decimal">
xs:attribute name="currency" type="xs:string" use="required" />
 </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
 </xs:element>
xs:element ref="nutritional_value_100_grams" minOccurs="0" />
xs:element ref="way_of_doing_part" minOccurs="0" />
xs:element ref="ingredient_description_part" />
 </xs:sequence>
 </xs:complexType>
 </xs:element>
xs:element name="recipesDB">
xs:complexType>
xs:sequence>
xs:element ref="recipe" minOccurs="0" maxOccurs="unbounded" />
 </xs:sequence>
 </xs:complexType>
 </xs:element>
xs:element name="insaturated">
xs:complexType mixed="true">
xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
xs:element ref="monosaturated" minOccurs="0" />
xs:element ref="polisaturated" minOccurs="0" />
 </xs:sequence>
 </xs:complexType>
 </xs:element>
xs:element name="monosaturated" type="xs:decimal" />
xs:element name="polisaturated" type="xs:decimal" />
xs:element name="saturated" type="xs:decimal" />
xs:element name="utensil">
xs:complexType>
xs:simpleContent>
xs:extension base="xs:string">
xs:attribute name="cooking_purpose" type="xs:boolean" use="optional" />
 </xs:extension>
 </xs:simpleContent>
 </xs:complexType>
 </xs:element>
 </xs:schema>
```

APPENDIX_26 Testing

26.1 Corpus1 – Ingredient taxonomy detailed

This corpus is formed by 25 files of different kinds of recipes. They are mainly Web pages in HTML and some little amount of plain texts. There is also included a dictionary of tagged ingredients to help the annotation tool to recognize these elements.

26.1.1 Annotation correctness (Corpus 1)

The first test performed, conformed to the following Ontology:

Ontology test-1
things(X) ==> concept(X) v relation(X). concept(X) ==> general_features(X) v ingredient_description_part(X) v way_of_doing(X) v nutritional_value(X) v course(X). general_features(X) ==>name(X) v retrieved_from(X) v posted_by(X) v price_person(X) v difficulty(X) v cooking_time(X) v number_of_servings(X). way_of_doing(X) ==> verb(X) v utensil(X). nutritional_value(X) ==> fats(X) v carbohydrates(X) v cholesterol(X) v proteins(X) v calories(X) v good_for(X). ingredient_description_part(X) ==> ingredient_description(X). ingredient_description(X) ==> ingredient(X) v quantity(X) v measure(X). ingredient(X) ==> food(X) v drink(X). food(X) ==> fish(X) v dairy_produt(X) v vegetable(X) v fat_oil(X) v cereal_grain(X) v egg(X) v meat(X) v fruit(X) v miscellaneous(X) cereal_grain_based(X). vegetable(X) ==> spice(X).

This table is about the correctness when annotating a corpus. These statistics correspond to the correctness to pair of tags; e.g.: <measure> </measure>
These statistics are obtained running the system over the same training corpus (it firstly removes the user annotations). Applies the rules it has inferred during the training phase and then compare these annotations with the user's ones, making these statistics:

The input corpus was formed by a set of 25 already annotated web pages. This corpus can be found in the CD. It is not included in the Appendix due to space limitations.

These are some of the results provided by the IE process:

Afterwards some statistics of this IE process are explained. These are the concepts they relate to:

Possible: Total number of elements originally annotated (by the user) with these tags.
Actual: Total number of elements the IE system has been able to annotate

Correct: Number of the annotated elements by the system that corresponds with the user annotations.
Wrong: Number of the annotated elements by the system that does not correspond with the user annotations.
Partial: Number of annotations partially correct (some parts are missing)
Missing: Number of the user annotations the system has not been able to reproduce.
Precision: Correctness of the inserted tags (%) [see XX for a more detailed explanation]
Recall: How many instances the system is able to recognize (%) [See XX for a more detailed explanation]

TAG	Possible	Actual	Correct	Wrong	Partial	Missing	Precision	Recall
<measure>	8	14	7	7	0	1	50	87
<calories>	5	8	5	3	0	0	62	100
<carbohydrates>	5	5	5	0	0	0	100	100
<cholesterol>	5	7	5	2	0	0	71	100
<egg>	4	1	1	0	0	3	100	25
<fats>	5	5	5	0	0	0	100	100
<number_of_servings>	9	12	8	4	0	1	66	88
<proteins>	5	5	5	0	0	0	100	100
<quantity>	78	79	42	13	24	12	53	77
<vegetable>	4	0	0	0	0	4	0	0
<verb>	7	4	3	0	1	3	75	50

26.2 Corpus2 – Remake the ingredient taxonomy

In the second test about the IE performance, the Ontology has changed a little.
As already explained the IE process can not perform well with such a detailed Ontology with so few training texts. The ideal solution would be to increase the corpus size, but due to technical limitations the other way around had to be chosen: reduce the complexity of the Ontology. As seen in the results of Corpus1, the conflictive point is the ingredient taxonomy. It is too detailed for this reduced training corpus.

In this new test, all the tags about ingredient kinds have disappeared and only one tag called <ingredient> (and the corresponding closing tag: </ingredient>) appear in the Ontology.

26.2.1 Annotation correctness (Corpus 2)

TAG	Possible	Actual	Correct	Wrong	Partial	Missing	Precision	Recall
<measure>	8	14	7	7	0	1	50	87

<calories>	5	8	5	3	0	0	62	100
<carbohydrates>	5	5	5	0	0	0	100	100
<cholesterol>	5	7	5	2	0	0	71	100
<ingredient>	57	33	21	1	11	25	63	45
<number_of_servings>	9	12	8	4	0	1	66	88
<proteins>	5	5	5	0	0	0	100	100
<quantity>	78	79	42	13	24	12	53	77
<verb>	7	4	3	0	1	3	75	50

The ingredient tag outperforms some of the ingredients described in the first test, like vegetable, although it is worst in other cases like the egg entity (as it had a really rigid structure it had the best performance)

26.3 Corpus 3 – Add gazetteers to the corpus

The corpus is the same than in the previous chapter, with the only difference that it includes several dictionaries (in XML files) that pretend to help obtaining a better performance.

This tests can not be shown because of some problems experienced with the IE tool. Adding the gazetteers to the initial corpus, the amount of data was much bigger of what the GUI could support. It could never show the results. Although this corpus could be connected trough the API to perform the IE process and database storage. I can state that it performs better but I can not give any statistics about it because of the tool limitations.