UNIVERSIDADE da MADEIRA

Developing an Ontology for e-Tourism using the Zachman Framework

Ana Lisete Nunes Escórcio

(Licenciada)

Thesis Submitted to the University of Madeira
for the degree of Master in Software Engineering

Funchal – Portugal

Dezembro 2007

Developing an Ontology for e-Tourism using the Zachman Framework

Ana Lisete Nunes Escórcio

(Licenciada)

Thesis Submitted to the University of Madeira
for the degree of Master in Software Engineering

Funchal – Portugal

Dezembro 2007

Orientador

Professor Doutor António Jorge Cardoso

Senior Researcher - SAP Research CEC Dresden

ABSTRACT

Tourism Information Services (TIS) are a new type of business systems that serve and support e-tourism and e-travel organizations, such as hoteliers, leisure suppliers, and travel agencies. One class of these systems relies on travel related information sources, such as Web sites, to create tourism products and services. The information extracted from these sources can serve as the springboard for a variety of tasks, including dynamic packaging, leisure activities and price comparison. Currently, with most tourism information systems, customers need to visit multiple independent Web sites to plan their leisure activities, register their personal information multiple times, spend hours or days waiting for response or confirmation, and make multiple payments by credit card. Consumers are discouraged with the lack of function falsities. To answer these limitations, an ontology for complementary e-tourism offer (Ce-TO) was built. By complementary e-tourism offer we understand everything that can be done by tourists in a city. Complementary offers exclude hotel and flight resources.

This ontology is the solution for data integration between different web applications since it offers a shared, organized, and common understanding of the data allowing better integration, communication, and interoperability of inter and intra organizational tourism information systems. Concepts like leisure, gastronomy will figure in our ontology.

A survey concerning the ontology tools available was done. This survey main goal was to analyse the tools that give support to methodologies that exist. At the end of our study, we noticed that every tool has some nice, but particular, features and we consider that choosing the right tool is related with the type of user (beginner/expert), with the type of ontology (small/large), with the features that a developer might require (for example, graph view, quick development, using certain data formats, etc.) and, not least, with the methodology that he/she pretends to use. Some of these tools were created during projects to build ontologies. The same happens to methodologies and ontologies. They emerged throughout the development of projects that made use of ontologies.

A methodology for building domain ontologies is proposed. This methodology is based on the Zachman Framework. This methodology was used to build Ce-TO.

Keywords

Software engineering, semantic web, ontologies, methodology, ontological engineering, e-tourism, tourism information systems.

ACKNOWLEDGMENT

I thank my advisor Dr. Jorge Cardoso for giving me an opportunity to work under him and in the Semantic E-Tourism Dynamic Packaging (SEED) project. I would also like to thank him for his encouragement, patience, guidance and timely support throughout the entire duration of the project and thesis.

I wish as well to thank my colleagues at SEED project for the great work environment they helped to create and for many fruitful idea discussions. Friday meetings at Universidade da Madeira were a very stimulating time.

While working towards my degree, I teached at Escola Básica e Secundária do Carmo (Câmara de Lobos) and I would like to thank the support and motivation gave by my colleagues and by my students from 12º2 (2006-2007).

Finally, I must thank my family, who has provided love and support over all of these years. My mother, father and sister have always believed in me and that has helped me to get to where I am now. They have also provided me with the best possible start to life that I could have ever wished for. I must also thank my sister for her availability to answer to all my print requests.

Most importantly, I want to thank my wonderful husband Nuno for his love and understanding. He never complained when I had to spend a weekend working on this dissertation, and always offered words of encouragement when I was feeling down.

This work was partially supported by POSI/EIA/56164/2004.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACM – Association for Computing Machinery

B2B – Business to Business

BNF – Backus Naur Form

CWM – Common Warehouse Metamodel

DAML - DARPA Agent Markup Language

DARPA - Defense Advanced Research Projects Agency

DL - Description Logic

ER – Entity Relationship

FIFA – Foundation for Intelligent Physical Agents

Flogic - Frame Logic

FTP - File Transfer Protocol

HTML - HyperText Markup Language

http - HyperText Transfer Protocol

IEEE- Institute of Electrical and Electronics Engineers

IT – Information Technology

IS – Information System

JOE - Java Ontology Editor

NS – NameSpace

OIL - Ontology Inference Layer

OTA – Open Travel Alliance

OWL - Web Ontology Language

TAGA – Travel Agent Game in Agenticities

TIS- Tourism Information System

RDF - Resource Description Framework

RDFS - Resource Description Framework Schema

SEED – Semantic E-Tourism Dynamic Packaging

SGML - Standard Generalized Markup Language

SHOE - Simple HTML Ontology Extensions

SOAP - Simple Object Access Protocol

SW – Semantic Web

UDDI -Universal Description, Discovery and Integration

UML – Unified Modeling Language

UNSPSC - United Nations Standard Products and Services Code

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

W3C - World Wide Web Consortium

Web - World Wide Web

WSDL – Web Service Definition Language

WTO – World Tourism Organization

WWW - world wide web

XML - eXtensible Markup Language

XMLS - eXtensible Markup Language Schema

1 INTRODUCTION

"It's time to get Semantic Web wise."
(Berners-Lee, 2007)

The World Wide Web (WWW) has changed the way people communicate with each other and the way business is conducted. It lies at the heart of a revolution that is currently transforming the developed world toward a knowledge economy and, more broadly speaking, to a knowledge society. Most of today´s Web content is suitable for human consumption. Typical uses of the Web today involve people's seeking and making use of information, searching for and getting in touch with other people, reviewing catalogs of online stores, ordering products, and buying things. These activities are not particularly well supported by software tools. The main obstacle to providing better support to Web users is that, at present, the meaning of Web content is not machine-accessible. Of course, there are tools that can retrieve texts, split them into parts, check spelling, and count their words. But when it comes to interpreting sentences and extracting useful information for users, the capabilities of current software are still very limited. The solution is to represent Web content in a form that is more easily machine-processable and to use intelligent techniques to take advantage of these representations. It is possible to claim that the challenge is engineering and technology adoption rather than a scientific one: partial solutions to all important parts of the problem exist. At the present, the greatest needs are in the areas of integration. We refer to this plan for revolutionizing the Web as the Semantic Web (SW). The SW will not be a new global information highway parallel to the existing World Wide Web; instead it will gradually involve out of existing Web (Antoniou and van Harmelen, 2004). The Internet is already the primary source of tourist destination information for travelers. About 95% of Web users use the Internet to gather travel related information and about 93% indicate that they visited tourism Web sites when planning for vacations (Lake, 2001). The number of people turning to the Internet for vacation and travel planning has increased more than 300% over the past five years. Tourism Information Systems (TIS) are a type of business systems that serve and support e-tourism and e-travel organizations, such as airlines, hoteliers, car rental

companies, leisure suppliers, and travel agencies. One class of these systems relies on travel related information sources, such as Web sites, to create tourism products and services. The information extracted from these sources can serve as the springboard for a variety of tasks, including dynamic packaging, travel planning, and price comparison. The World Tourism Organization (WTO, 2005) predicts that by 2020 tourist arrivals around the world would increase over 200%. Tourism has become a highly competitive business for tourism destination over the world. Competitive advantage is no longer natural, but increasingly driven by science, information technology and innovation (Cardoso, 2005).

Ontologies are engineering artifacts that are built according to a specific vocabulary used to describe a certain reality with a set of explicit assumptions regarding the intended meaning. An ontology describes a formal specification of a certain domain and can be used as: 1) a shared understanding of a domain of interest, in our case, e-tourism, 2) a formal and machine manipulability model of a domain of interest and 3) by being domain specific enables modularization. According to (Swartout et al., 1997), ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base.

As a conclusion, Semantic Web technology, namely ontologies, has a huge potential when used in e-Tourism since it promotes: 1) semantically enriched information searching between operators and users, 2) integration and interoperability between systems, 3) personalized and context aware recommendations and 4) internalization (standardization).

1.1 OVERVIEW

Tourism is one of the sectors of the World economy with the best outlook. The increase in time for leisure activities and its social importance promotes the growth of the Tourism Industry. The rapid growth of the Internet and the continual adoption of innovative technology have led to serious changes in the travel industry. Due to the constantly changing business environment, one of the latest concepts of the tourist industry is "Dynamic Packaging". We can define dynamic packaging as the ability of a system to combine different tourism products in order to create a package. This ability will offer advantages for the tourist, because by creating packages he can obtain products with lower prices. Another advantage is that the tourist only has to introduce his personal data, like name, address, credit card, etc, only one time to buy all the products in a package. The SEED project started with the aim of developing a new way to implement dynamic packaging systems. To create dynamic packages, systems must integrate different tourism data sources. These data sources can have a very different data formats and can be accessed by very different forms. To deal with the heterogeneity of the tourism data sources, Semantic Web technology is used. The use of Semantic Web technology helps the integration of the data sources. By creating a semantic model of the tourism domain and associating this model with each one of the data sources, we can more easily integrate them

(Cardoso and Fernandes, 2005). The semantic model can be an ontology. Ontologies provide a conceptualization of a domain that can be shared among people, computer systems and applications (Grubber, 1993).

1.2 PROBLEM STATEMENT

The dynamic packaging technology helps online travel customers to build and book vacations. It can be described as the ability for a customer to put together elements of a trip. In the offline world, such packages used to be put together by tour operators in brochures. This new technology includes the ability to combine multiple travel components on demand to create a reservation. The package that is created is handled seamlessly as one transaction and requires only one payment from the consumer, hiding the pricing of individual components (Cardoso, 2005).

Dynamic packaging has been introduced as an approach for achieveing competitive advantage, since it aims at providing consumers with individually customized and flexible travel packages.

Tourism is a data rich domain. This data is stored in many hundreds of data sources and many of these sources need to be used in concert during, for instance, the organization of a day in family or a wellness afternoon experience.

The success of a tourist area largely depends on the complementary products it offers. In times when the demands of tourism are changing and are much more demanding, the role played by the complementary offer is considered to be of extraordinary importance. We all have experienced difficulties when looking for information about complementary offer. Sometimes only brochures and tables are available at the hotel reception (offline world).

The main goal of the proposed research is dedicated to helping the travel industry take full advantage of the latest Internet technologies. One of the main points of our work is focus on the development of an ontology that can serve as basis for the semantic integration of several suppliers in dynamic packaging (Cardoso, 2005). An ontology based approach for supporting interoperability of different information systems in the tourism industry through semantic mappings has been described in (Fodor and Werthner 2004-5).

With the growth of demand for customized tourism activities, agencies and hotels seek technology that provides their personnel and clients the flexibility to put together unique dynamic packages from a range of alternatives. Ontologies can be applied in the area of dynamic packaging to explicitly connect data and information from tourism information systems to its definition and context in machine-processable form.

Initially we developed a small ontology without following any of the methodologies available. We were confronted with some difficulties concerning the identification of terms. Our initial set of

terms was huge and we did not know how to select them. A study of the methodologies available was made. This study showed that all methodologies presented drawbacks.

In the previous year we studied the Zachman Framework. Since this framework represents the processes around different points of view (known as perspectives) taken by various players and not as a series of steps like all the ontology development methodologies we concluded this would be a good approach for developing a new methodology.

The deployment of semantics (ontologies) help articulate a well-defined set of common data elements or vocabulary that can support communication across multiple channels, expedite the flow of information, and meet travel industry and customer needs. The ontology is going to be developed taking in consideration the traveler's point of view. Therefore, we consider two main aspects:

- Resource: "What can a tourist do while staying at Madeira Island?"
- Experience: "What can a tourist experience while staying at Madeira Island?"

The resource concept gathers all the products available. For instance, transportation resources, or entertainment resources. In this case the tourist knows what he wants to do. The tourist is only looking for information about a certain resource. Experience is related to different kinds of feeling a tourist can have. For example, he does not know what he wants to do but he knows he wants to feel something or experience something. Therefore, resources are organized and combined in ways that fulfill the requirements of a certain type of experience.

1.3 CONTRIBUTIONS

Information dissemination and exchanges are the key backbones of travel industry, which is currently mainly based on the printed brochures, posters, advertisements via television or limited web access. Semantic Web will bring the revolution to this area by not only exponentially extending the dissemination and exchange channels with unlimited access, unlimited time and unlimited locations, but also assisting users with smart information searching, integrating, recommending and various intelligent services (E-Tourism Group, 2004).

Ontologies are a way of specifying the structure of domain knowledge in a formal logic designed for machine processing. The effect on information technology (IT) is to shift the burden of capturing the meaning of data content from the procedural operations of algorithms and rules to the representation of the data itself.

Numerous commercial and open-source software tools are available for building and deploying ontologies, and for integrating inference systems with web and database infrastructures. Increasingly, these tools directly support the emerging web ontology standards.

The intent of this survey is to: 1) summarize the features of the manual editing tools so that we can select an editing tool that fulfills all the needs and 2) to determine the tools that support the methodologies to develop ontologies.

The study of the methodologies avalaible allowed to identify some of the drawbacks of the methodologies. We decided to build a new methodology that tries to overcome some of these drawbacks. This new methodology is based on the Zachman Framework since: 1) this framework can be divided into two views ("machine" view and "human" view) and 2) this framework is organized around several points of view.

The e-Tourism ontology will (Uschold and Gruninger, 1996) provide a way of viewing the world of tourism. It organizes tourism related information and concepts. The ontology will allow achieving interoperability through the use of a shared vocabulary and meanings for terms with respect to other terms. In an early stage of our project, a partial ontology for the e-Tourism was created using Protégé (Protégé, 2005) and the OWL (W3C, 2004) language. This was a very time-consuming task since it was necessary to find out information about real tourism activities and infrastructures on the Web and feed them into the knowledge base (Cardoso, 2005).

1.4 THESIS LAYOUT

This thesis is organized in 9 chapters and is structured as followed. Chapter 2 describes the methodologies that can be used to develop ontologies. It is also given an overview of the tourism ontologies that have been created.

In Chapter 3, important concepts to this thesis are given. This chapter is split into five main sections. The first section is the Semantic Web concept. The notion of an ontology and the advantages of using it is specified in the second section. Ontological engineering is defined in the third section. A synopsis of the ontology languages available is also done in the fourth section. The last section is concerned with the application of the Semantic Web.

A study was made concerning the editing tools for ontology construction and it was published as a chapter in a book. Chapter 4 sums up this chapter.

This thesis was integrated in a project, the SEED project. An overview of this project is given is Chapter 5.

Since this thesis is focused on the development of an ontology for e-Tourism using a new methodology, Zachman Ontology Framework (ZOF), Chapter 6 describes this new methodology and Chapter 7 overviews the ontology built with this methodology.

Finally, Chapter 8 contains a brief description of the ontology developed using the ZOF methodology, its strengths and weaknesses, along with conclusions and possible extensions of this work.

2 Related Work

An overview of the work related to the area of this thesis is done in this section. Section 2.1 gives an overview of the methodologies used to develop ontologies. Section 2.2 presents some of the tourism ontologies developed over the past few years.

2.1 Ontology Methodologies

There are several methodologies that support ontology development. Before building our ontology a study concerning the available methodologies was done. This study main goal was to help selecting a methodology that could guide us during the ontology development. However, all the methodologies had drawbacks and were built in projects. Therefore, many of them worked fine for the projects were they were built.

Figure 1 shows how this evolution occurred. In 1990, Lenat and Guha published information concerning the Cyc development (Lenat and Guha, 1990). In 1995, in the domain enterprise modeling, the experience from two projects dictated guidelines that were later refined according to the experience from developing the Enterprise Ontology (Uschold and King, 1995) and the TOVE Project Ontology (Grüninger and Fox, 1995). A new method to build an ontology in the domain of electrical networks was presented as part of the Espirit KACTUS Project (Bernaras et al., 1996). Also in 1996, the Methontology methodology (Gómez-Pérez et al., 1996) emerged for the first time. This methodology was extended in further papers. In 1997, a new method was proposed based on the SENSUS ontology (Swartout et al., 1997). The On-to-Knowledge methodology (Staab et al., 2001) only came out some years later. Figure 1, works as timeline for presenting the main methodologies evolution. As we can see in figure 1, the 1990s and the first years of this new century have witnessed the growing interest of many practitioners in approaches for building ontologies from scratch and for reusing other ontologies (Gómez- Pérez et al., 2004).

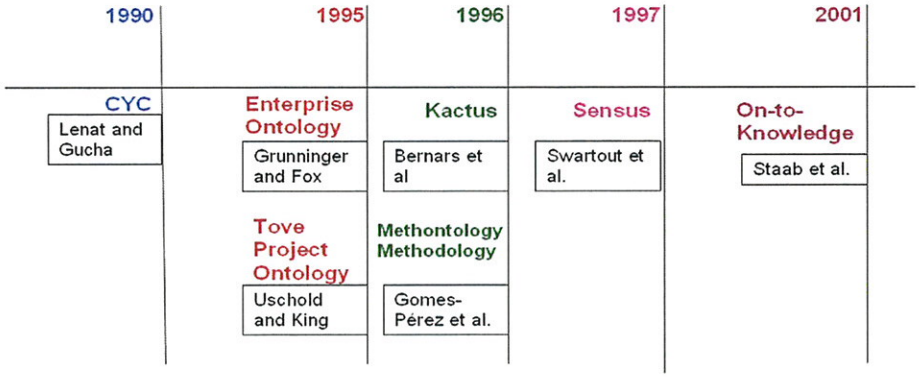

Figure 1: Methods and Methodologies for building ontologies

Each methodology will be described in a further section.

2.1.1 Ontology Life Cycle

The Ontology Life Cycle (Pinto and Martins, 2004) identifies when the activities should be carried out, that is, it identifies the set of stages through which the ontology moves during its life time. It also describes what activities are to be performed in each stage and how the stages are related.

According to Helena Sofia Pinto and João P. Martins (Pinto and Martins, 2004) the accepted stages through which an ontology is built are: 1) specification, 2) conceptualization, 3) formalization, 4) implementation and 5) maintenance. Each stage is characterized by the activities that are performed or the techniques that are used. It is possible to see that the stages identified follow some of the guidelines from the stages of Software Engineering. For instance, the first stage, specification, is the starting point for software development. The implementation stage is also a typical stage in the software engineering development process (Pinto and Martins, 2004). And so, it is possible to corroborate that there are some similarities between the ontology development and software development. This happens since the software engineering field is more mature than the ontological engineering.

2.1.1.1 Stages

In the first stage, the specification stage (Pinto and Martins, 2004), the purpose and scope of the ontology has to be identified. A list of purpose questions and scope questions should be made. Purpose questions are questions that justify the need for an ontology to be build. The question *"Why is the ontology being built?"* is a purpose question. In order to clarify and bound the domain a list of scope questions should be organized. A question like *"What are its intended uses and end users?"* is a scope question given that the answers to this questions means identifying the end public of the ontology.

In the paper *Ontology Development 101: AGuide to Creating Your First Ontology* (Noy and MacGuiness, 2001) it is suggested that the first step to start the development of an ontology is

to determine the domain and scope of the ontology. They believe that several questions should be answered, for instance:

- What domain will the ontology cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?
- Who will use and maintain the ontology?

Another group of ontologists, Grunninger and Fox (Gruninger and Fox, 1995), believe that one of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on an ontology should be able to answer (competency questions).

The second stage, the conceptualization stage (Pinto and Martins, 2004), the ontology should be described in a conceptual model so that it meets the specification found in the previous step. Different methodologies propose the use of different conceptual models, from informal sketchy models like mind maps to semi-formal models like the binary relations diagram or concept dictionary. But, what is a conceptual model of an ontology? The conceptual model of an ontology consists of concepts in the domain and relationships among those concepts. These groups of highly connected concepts usually correspond to different modules (subontologies) into which the domain can be decomposed.

In the formalization (Pinto and Martins, 2004), the third stage, the conceptual description should be transformed into a formal model.

In the Implementation stage (Pinto and Martins, 2004), the formalized ontology should be implemented in a knowledge representation language. For that, one commits to representation ontology, chooses a representation language and writes the formal model in the representation language using the representation ontology.

In the maintenance stage (Pinto and Martins, 2004) the ontology should be updated and corrected if needed.

There are also activities that should be performed during the whole life cycle, as stated by Pinto and Martins (Pinto and Martins, 2004):

- *Knowledge acquisition*: Acquire knowledge about the subject either by using elicitation techniques on domain experts or by referring to relevant bibliography. Several techniques can be used to acquire knowledge, such as brainstorming, interviews, questionnaires, text, analysis, and inductive techniques
- *Evaluation*: technically judge the quality of the ontology
- *Documentation*: report what was done, how it was done and why it was done. Documentation associated with the terms represented in the ontology is particularly important, not only to improve its clarity, but also to facilitate maintenance, use and reuse.

2.1.2 Main Ontology Development Methods and Methodologies

It is not possible to evaluate which method/methodology is the best since it depends of our intention. For instance, a certain method/methodology can be the key to support the building of a linguistic ontology and not so good for the construction of a domain ontology.

Let us distinguish method from methodology. As we saw before in the previous definition of methodology, both methods and techniques are parts of methodologies. Method is a set of "orderly process or procedure used in the engineering of a product or performing a service" (IEEE, 1990).

2.1.2.1 The Cyc Method

Microelectronics and Computer Technology Corporation started to create Cyc (Lenat and Guha, 1990), a huge knowledge base with common sense knowledge. To implement Cyc, the CycL Language was used.This method proposes the following processes:

- Manual Coding of articles and pieces of knowledge
- Knowledge coding aided by tools using the knowledge already stored in the Cyc KB.
- Knowledge codification mainly performed by tools using knowledge already stored in the Cyc method.

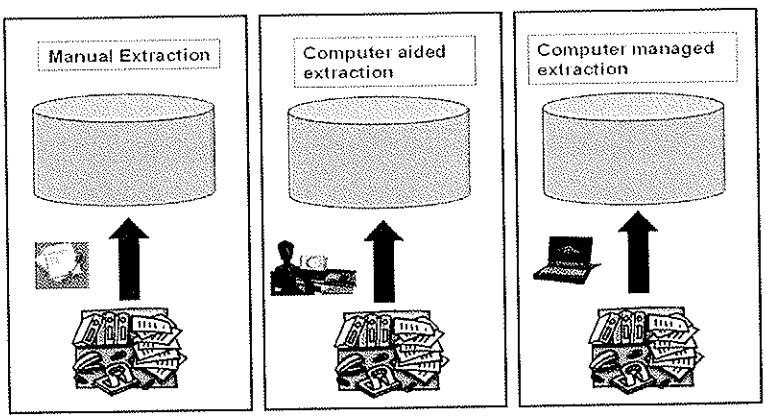

Figure 2: Processes proposed by the Cyc method (Lenat and Guha, 1990).

The reason why Cyc can be considered as an ontology is because it can be used as a substrate for building different intelligent systems that can communicate and interact.

This methodology can be supported by almost all of the ontology development tools available.

2.1.2.2 The Uschold and King's method

The ontology was developed as a part of the enterprise project by the Artificial Intelligence Applications Institute at the University of Edinburgh with its partners IBM, Lloyd's Register, Logical UK limited, and Unilever.

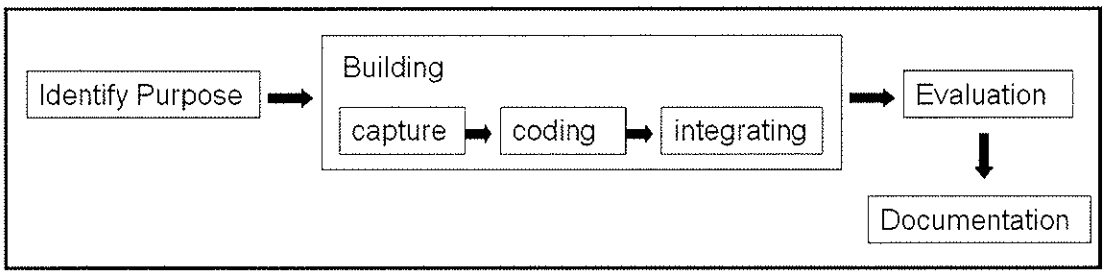

Figure 3: Main processes of the Uschold and King's method (Uschold and King, 1995).

As we can see in the figure 3, the processes that need to be performed in order to build an ontology are: 1) identify the purpose of the ontology; 2) build it; 3) evaluate it and 4) document it (Uschold and King, 1995).

In order to identify the purpose of the ontology the following questions should be answer: Why the ontology is going to be build?; who are its intended users?; what the relevant terms on domain will be?

During the stage of building the ontology it is important to capture the knowledge. The following tasks should be done for capturing knowledge: 1) identify key concepts and relationships in the domain of interest; 2) produce precise and unambiguous textual definitions for such concepts and relationships and 3) identify the terms that refer to such concepts and relationships.

Uschold and King pointed out 3 strategies to identify the concepts: bottom-up; top-down and middle-out.

To evaluate the ontology Uschold and King took the definition from Gomez-Perez and colleagues (Goméz-Pérez et al, 1996) and stated that: "to make a technical judgment of the ontologies, their associated software environment, and documentation with respect to a frame of reference. (...) The frame of a reference may be requirement specifications, competency questions, and/or the real world."

The process of documentation is a guideline example to locate similar definitions together or to create naming conventions such as: using upper or lowercase letters to name the terms, writing the terms of the representation ontology in uppercase, for instance.

The main drawback of this method is the lack of a conceptualization process before implementing the ontology. This methodology can be supported by almost all the ontology tools.

2.1.2.3 Grunninger and Fox's methodology

This methodology is inspired by the development of knowledge based systems using first order logic. Gruninger and Fox (Gruninger and Fox, 1995) propose identifying intuitively the main scenarios, that is, possible applications in which the ontology will be used. Then, a set of a natural language questions, called competency questions, are used to determine the scope of the ontology. These questions and their answers are both used to extract the main concepts and their properties, relations and formal axioms of the ontology.

On the other hand, knowledge is formally expressed in first-order logic. This is a very formal methodology that takes advantage of the robustness of classic logic and can be used as a guide to transform informal scenarios in computable models.

They identified the following processes: 1) identify motivating scenarios, 2) elaborate informal competency questions, 3) specify the terminology using first order logic, 4) write competency questions in a formal way using formal terminology, 5) specify axioms using first order logic and 6) specify the completeness theorems.

The main drawback of this methodology is the fact that some management and support activities are missing.

2.1.2.4 KACTUS Approach

The purpose of this project was to investigate the feasibility of knowledge reuse in complex technical systems and the role of ontologies (Bernaras et al, 1996). This approach for developing ontologies is conditional by application development. Thus, every time an application is built, the ontology that represents the knowledge required for the application is refined.

The following processes occur every time an application is developed: 1) specification of the application; 2) preliminary design based on relevant top-level ontological categories and 3) ontology refinement and structuring.

The main drawback of this methodology is the fact that the ontology of this application can be considered as the junction between the domain ontology for diagnosis and service recovery planning. The union of the ontologies yields a set of sub-ontologies that belong to the intersection and other sets used for one application or the other but not for the both at the same time. The sub-ontologies of the intersection are more likely to be reused, since the relevant adaptation in these ontologies should be carried out in, at least, two different applications.

2.1.2.5 Methontology Methodology

This methodology was developed within the Ontology group at Universidad Politécnica de Madrid. Methontology (Fernández-López et al, 1997) enables the construction of ontologies at the knowledge level. Methontology has its roots in the main activities identified by the software development process and in knowledge engineering methodologies. This methodology includes: the identification of the ontology development process, a life cycle based on evolving prototypes, and techniques to carry out each activity in the management, development-oriented, and support activities. ODE (Blázquez et al., 1998) and WEBODE (Aspiréz et al., 2003) were built to give technological support to Methontology.

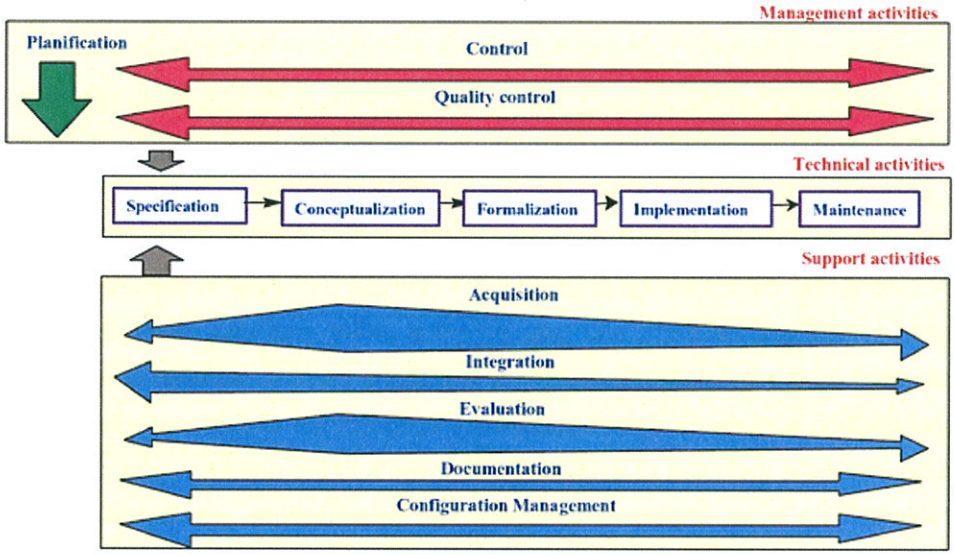

Figure 4: Methontology Methodology activities (Fernández-López et al., 1997).

Methontology also considers that the activities performed during the development of an ontology may involve performing other activities in other ontologies already built or under construction.

Therefore, Methontology considers not only intra-dependencies, but also inter-dependencies. Inter-dependencies are defined as the relationships between activities carried out when building different ontologies. Instead of talking about the life cycle of ontology, we should talk about crossed life cycles of ontologies. The reason is that, most of the times and before integrating ontology in a new one, the ontology to be reused is modified or merged with other ontologies of the same domain. This methodology is supported by WebODE ontology development tool.

2.1.2.6 SENSUS-based Method

The SENSUS-based Method (Swartout et al., 1997) is used for building the skeleton of the domain ontology starting from a huge ontology, the SENSUS ontology. The method proposes to link domain specific terms to the ontology and to prune, in the huge ontology, those terms that are irrelevant for the new ontology we wish to build. According to this method, to build an ontology in a specific domain the following processes should be done: 1) identify seed terms (key terms); 2) link manually the seed terms to SENSUS; 3) add paths to the root; 4) add new domain terms and 5) add complete subtrees.

2.1.2.7 On-To-Knowledge Methodology

The On-To-Knowledge methodology (Staab et al., 2001) proposes to build the ontology taking into account how the ontology will be used in further applications. Consequently, ontologies developed with this methodology are dependent of the application. Another important characteristic is that On-To-Knowledge proposes ontology learning for reducing the efforts made to develop the ontology. This methodology also includes the identification of goals to be

achieved by knowledge management tools, and is based on analysis of usage scenarios (Staab et al., 2001).

The processes proposed by this methodology are: 1) feasibility study, 2) kickoff, 3) refinement, 4) evaluation and 5) maintenance. The feasibility study is applied to the complete application and, therefore, should be carried out before developing the ontologies. In fact, the feasibility study serves as a basis for the kickoff process. The kickoff is an ontology requirements specification document that describes the following: the domain and goal of the ontology; the design guidelines; available knowledge sources; potential users and use cases as well as applications supported by the ontology. Competency questions can be useful to elaborate the requirements specification document. The requirements specification should lead the ontology engineer to decide about the inclusion or exclusion of concepts in the ontology, and about their hierarchical structure. The most important concepts and relationships are identified on an informal level. In the kickoff process, the developers should look for potentially reusable ontologies already developed. In the refinement process the goal is to produce a mature and application-oriented "target ontology" according to the specification given in the Kickoff process. This refinement process is divided into two activities: knowledge elicitation process with domain experts and formalization (the ontology is implemented using an ontology language). The evaluation process works as a proof of the usefulness of the developed ontologies and their associated software environment. The maintenance is important to clarify who is responsible for the maintenance and how this should be carried out. This methodology proposes to carry out ontology maintenance as part of the system software and is supported by OntoEdit development tool.

2.1.3 Life Cycle Comparison

The tables below show the classification of the activities/stages that the main methodologies (TOVE, Enterprise and Methontology) have when comparing their Life Cycle (Specification, Conceptualization, Formalization, Implementation, Maintenance, Knowledge Acquisition, Evaluation and Documentation).

TOVE (Gruninger and Fox's methodology)

Activity	Corresponds to
Capture motivating scenarios and formulate informal competency questions	*Specification*
Specify terminology, formulate formal competency questions and specify axioms and definitions in FOL	*Conceptualization* *Formalization* *Implementation*
Evaluate competency and completeness	*Evaluation*

Table 1: Similarities between Gruninger and Fox's methodology and the ontology life cycle.

Table 1 compares the activities proposed by Gruninger and Fox's methodology with the life cycle proposed by Pinto and Martins. Conceptualization, Formalization and Implementation are done in the second activity proposed by Gruninger and Fox's methodology. There is a clear separation between the specification stage and the others stages and also between the evaluation stage and the other stages.

Enterprise (Uschold and King's methodology)

Activity	Corresponds to
Identify purpose and scope	*Specification*
Capturing knowledge	*Knowledge Acquisition* *Conceptualization*
Coding	*Formalization* *Implementation*
Evaluate	*Evaluation*
Documentation	*Documentation*

Table 2: Similarities between Uschold and King's methodology and the ontology life cycle.

Table 2 compares the activities proposed by Uschold and King's methodology with the life cycle proposed by Pinto and Martins. The activity capturing knowledge corresponds to the second and third stage of the ontology life cycle. Formalization and Implementation corresponds to the coding activity of Uschold and King's methodology. This methodology introduces a new activity, documentation. This activity corresponds to the documentation stage in the ontology life cycle.

Methontology

Activity	Corresponds to
Requirement specification	*Specification*
Conceptualization and domain knowledge	*Conceptualization*
Formalization of conceptual model	*Formalization*
Implementation of formal model	*Implementation*
Maintenance	*Maintenance*
Knowledge acquisition	*Knowledge acquisition*
Documentation	*Documentation*
Evaluation	*Evaluation*

Table 3: Similarities between Methontology and the ontology life cycle.

Table 3 compares the Methontology with the ontology life cycle. This methodology is the only one whose activities have direct correspondence to a stage in the ontology life cycle proposed by Helena Pinto and João Martins (Pinto and Martins, 2004)

It is possible to summarize these tables as followed:

Stages	*Methontology*	*Enterprise*	*TOVE*
Specification	Requirement Specification	Identify purpose and scope	Capture motivating scenarios and formulate informal competency questions
Conceptualization	Conceptualization and domain knowledge	Capture knowledge	Specify terminology, formulate formal competency questions and specify axioms and definitions in FOL
Formalization	Formalization of conceptual model	Coding	
Implementation	Implementation of formal model		
Maintenance	Maintenance	×	×
Knowledge Acquisition	Knowledge acquisition	Capture knowledge	×
Evaluation	Evaluation	Evaluation	Evaluation
Documentation	Documentation	Documentation	×

Table 4: Summary of tables 1,2 and 3 presented above.

Table 4 sums up table 1, 2 and 3. Maintenance is one of the stages that is not considered in some of the methodologies. The other stages are considered in almost all of the methodologies. Specification is the initial stage for all the methodologies presented.

2.1.4 Methodologies Comparison

To compare the construction strategy the following set of criteria was proposed by Fernandez-Lopez and Gómez-Pérez (Gómez-Pérez et al., 2004):

- *Life cycle proposal*: identifies the set of stages through which the ontology moves during life time.
- *Strategy according to the application*: relates the degree of dependency of the ontology with the application using it.
- *Strategy to identify concepts*: there are three possible strategies to identify concepts (bottom-up; top-down or middle-out).
- *Use of a core ontology*: analyze whether it is possible or not to use a core ontology as a starting point in the development of the domain ontology.

Concerning the life cycle, the Cyc methodology, the KACTUS, and the Methontology methodology have proposed evolving prototypes. Grunninger and Fox have also proposed evolving prototypes or incremental. Uschold and King and SENSUS have not proposed a life cycle. The On-to-Knowledge has proposed an incremental and cyclic methodology with evolving prototypes.

The Cyc, Ushold and King and Methontology methodologies are independent of an application. The Kactus, and the On-to-Knowledge methodology are dependent of an application. The Grunniner and Fox methodology and SENSUS are semi-dependent of an application.

In order to identify concepts Uschold and King's methdology, Grunninger and Fox's methodology and Methontology methodology use the middle-out strategy. The KACTUS Methontology uses the top-down methodology. SENSUS and Cyc do not specify a strategy. The On-to-knowledge offers the possibility to use top-down, bottom-up or middle-out to identify concepts.

The use of a core ontology only happens with the Cyc and SENSUS methodology. Methontology and On-to-Knowledge methodologies use a core ontology according to the available resources.

There are tools that give full or partial support to the methodologies. For instance, the Cyc tools give support to the Cyc methodology and ODE, WebOde, OntoEdit and Protégé-2000 gives support to the Methontology methodology. The OntoEdit supports the On-to-knowledge methodology.

To sum up, the prevailing life cycle model is evolving prototypes, although there are methodologies that do not propose any life cycle model like Ushold and King or SENSUS. There is a wide variety of ontology development strategies. Some approaches consider the application dependent, and others independent. In order to identify concepts in taxonomy the most commonly used strategy is middle-out. Most of the methodologies cover the development activities, especially the ontology implementation. So far none of the methodologies cover all the stages identified by Helena Sofia and João Pinto.

The most complete methodology is Methontology since it provides the most accurate descriptions of each activity. The On-to-knowledge methodology describes more activities than the other approaches. Grunninger and Fox's methodology is the methodology that presents the highest degree of formality. This methodology has only been tested on the business domain. The SENSUS-based methodology does not mention the ontology life cycle.

The majority of the methodologies present some drawbacks. Some of the methodologies have not been used by external groups or have only been used in a single domain. Moreover, none of the available tools gives support to all the activities necessary in ontology building.

2.1.5 Main Drawbacks of the Methodologies

According to the paper Methodologies for Ontology Development (Jones et al., 2000), there are considerable differences between some of the methodologies described above. They believe that some issues need to be addressed if a proper methodology, with a clearly defined range of application, is to emerge. The issues are:

Many of the methodologies take the task of describing the capabilities of the ontology as a starting point;

The methodologies described seem to be split between mainly stage-based models (TOVE and ENTERPRISE) and evolving prototype models (Methontology);

There are typically separated stages to produce first an informal description of the ontology, and then its formal embodiment in an ontology language.

There is an expectation that a library of ontologies will be accumulated and form the basis for further ontology development;

A complete methodology must provide guidelines to assist the ontological engineering in making choices at a variety of levels, from the high level structure of the ontology, to the fine detail of whether or not to include some particular distinction.

So, these five issues ought to be addressed when building a methodology for building ontologies.

To build an ontology we need, at least, a source of knowledge and some means to extract the knowledge and encode it in the desired formal language. Therefore, the typical "core" steps in ontology building are: 1) knowledge acquisition, 2) conceptualization and 3) formalization/implementation (Jones et al., 2000). Typically, during knowledge acquisition, a set of concepts is produced, along with their natural language definitions. These natural language definitions will serve as a basis to elicit the formal definitions that will form the ontology. Additionally, they will serve to document the intended meaning of the concept names. As the ontology builders are typically not experts in the target domain, this and the next steps should be followed by domain experts. If this is not possible, at least some discussion should take place with them to elicit knowledge and verify that the interpretation is correct. After completing part of the knowledge acquisition phase, it is time to start building the conceptual model – conceptualization. This model defines the meaning of concepts through the relations in which they participate and, maybe, some formal axioms or axiom sketches. The effort here is to encode in relations what is described in the natural language definitions, at least as much as is appropriate to the intended use of the ontology. Finally, formalization/implementation consists in expressing the conceptual model in a formal language with a particular expressive power. This may require a revision of the conceptual model to conform to the expression constraints imposed by the formal language if these were neglected during the creation of the conceptual model. This step is usually supported by ontology editing tools that provide a way to enter the conceptual model and then write it to a number of formal languages in a user friendly way.

For ontologies that are to be shared and used by groups other than their creators, there must be some assurances about their development and quality. Ontology building is usually done for a specific use and that influences the design decisions made during its development. Besides proper documentation, some form of evaluation of the quality of the ontology is also necessary. These requirements do not correspond to steps, by itself, but to activities that must occur throughout the development process. Ontology evaluation in the knowledge acquisition phase usually overlaps with the creation of the conceptual model (Jones et al., 2000).

2.2 DOMAIN ONTOLOGIES FOR E-TOURISM

In this section an overview of the domain ontologies for e-tourism is done. There are five subsections. In each subsection a description of the ontology is given.

This study helped to: 1) better understand the tourism domain, 2) know how the ontologies were organized and 3) to employ the knowledge obtained in the construction of our ontology.

Domain ontologies (Mizoguchi et al., 1995) are reusable in a given specific domain (medical, engineering, law, enterprise, tourism, etc). These ontologies provide vocabularies about concepts within a domain and their relationships, about the activities taking place in that domain, and about the theories and elementary principles governing that domain (Gómez-Pérez et al., 2004).

According to Guarino and colleagues (Guarino et al. 1995), an ontology can be understood as an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality.

There are several ontologies for the Tourism domain. They are: TAGA ontology, OTA ontology, Mondeca ontology and the Harmonize ontology (Gordon et al., 2005).

2.2.1 TAGA

The Travel Agent Game in Agentcities (TAGA) is an agent framework for simulating the global travel market on the Web. Its purpose is to demonstrate Agentcities (Agentcities, 2007) and Semantic Web technologies. TAGA works on FIPA-compliant platforms within the Agentcities Environment (FIPA, 2007). In addition to the FIPA content language ontology, TAGA defines two domain ontologies to be used in simulations. The first TAGA ontology covers basic travel concepts such as itineraries, customers, travel services, and servicereservations. The second ontology is devoted to auctions and defines different types of auctions, roles the participants play in them, and the protocols used etc. Unfortunately, TAGA ontologies are limited by their usability (only defining very broad concepts, in not much detail) and fairly unrealistic due to the nature of TAGA simulations (Gordon et al., 2005).

2.2.2 OTA

The Open Travel Alliance (OTA) specifications have been designed to serve two purposes: (a) as a common language for travel-related terminology and (b) a mechanism for exchange of information between travel industry members. The OTA specification is an attempt to create, from the business perspective, a possibly complete ontology for the "world of travel". While the word ontology itself is not used in the description of the project (OpenTravel, 2007), it is possible to view the OTA specifications as a comprehensive ontology, defining concepts such as AirSchedule, GolfCourseReservation, HotelContentDescription, HotelPreferences, etc. The

OTA specification has already been utilized in some travel-related AgentCities projects (Gordon et al.).

2.2.3 Mondeca

Mondeca's (Mondeca, 2007) tourism ontology defines tourism concepts based on the WTO thesaurus (World Tourism, 2005). These include, among others, terms for tourism object profiling, tourism and cultural objects (place, museum, restaurant, housing, transportation, events...), tourism packages and tourism multimedia content. Mondeca created a proprietary system ITM that is used to manage its travel ontology (Gordon et al., 2005).

2.2.4 Harmonize

Harmonize is an attempt at ontology-mediated integration of tourism systems following different standards (Deri, 2004). Its goal is to allow organizations to exchange information without changing data structures. The Harmonize project also involves sub-domains that are only partially related to the world of travel: geographical and geo-spatial concepts, means of transportation, political, temporal, activity/interest, gastronomy etc. These sub-domain concepts can be used within the travel system (as needed) or incorporated into the ontology constructed for the system. A comprehensive guide was prepared within the E-Tourism project. Here, it is claimed that next generation of "e-tourism" will be powered by the Semantic Web technology (resulting in an e-tourism Semantic Web portal which will connect the customers and virtual travel agents from anywhere at anytime) (Gordon et al., 2005).

2.2.5 DAML-Based Ontologies

A number of "minimalist" travel ontologies can be found within the DAML language portal (DAML, 2006). For instance, the Itinerary-ont is a simple ontology for representing travel itineraries (DAML, 2006). It was actually defined as an itinerary-ont.n3 and translated to DAML (and later OWL) using Common Warehouse Metamodel (CWM). It reuses the airport codes ontology and involves definitions of only the most basic terms like Aircraft, Class, and Flight etc. Another example is the Trip Report Ontology (DAML webpage) that defines terms like Airfare, Amount, Date, etc (Gordon et al., 2005).

Tourism is a data rich domain. Data is stored in many hundred of data sources and many of these sources need to be used in concert during the development of tourism information systems (Cardoso and Sheth, 2006). These ontologies present a way of viewing the world of tourism. It organizes tourism related information and concepts. The use of ontologies for tourism will provide ways to achieve integration and interoperability between systems and persons.

3 CONCEPTS

Before presenting the work developed it is important to make an introduction to the area in which this work fits and the concepts involved. Section 3.1 describes the Semantic Web area, presenting some defintions, goals and principles. Section 3.2 presents the ontology concept.

Ontological engineering has been the subject of research for a long time and multiple methodologies for the creation of ontologies have been proposed. Section 3.3 gives an overview of the ontological engineering field.

Web Ontology Language (OWL) is the language used to build the ontology. This language is described in Section 3.4.

Examples of the application of SW are provided in the last Section, Section 3.5.

3.1 SEMANTIC WEB

> "...The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation" (Berners-Lee et al., 2001)

The Semantic Web (SW) is a project and a vision of the World Wide Web Consortium. It is an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The Web displays a lot of data that is mostly understood by humans (Berners Lee et al., 2001). While the current web is only human-understandable, the SW vision intends to represent web content in such form that it enables machine processing. Therefore, the challenge of the SW is to provide a language that

expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported onto the Web. For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning.

Knowledge representation, as this technology is often called, is currently in a state comparable to that of hypertext before the advent of the Web. Traditional knowledge-representation systems typically have been centralized, requiring everyone to share exactly the same definition of common concepts such as "parent" or "vehicle." But central control is stifling, and increasing the size and scope of such a system rapidly becomes unmanageable. The Semantic Web will enable machines to comprehend semantic documents and data, not human speech and writings (Berners-Lee et al., 2001).

SW technologies find their first commercial interest in organizations consistently facing data integration needs and always seeking for better data integration solutions. In fact, according to the InfoWorld's 2002 Application Integration Survey of Information Technology (IT) Leaders, integration costs consumed at that time an average of 24 percent of the yearly IT budget (Yager, 2002). For midsize to large companies this represents millions of dollars. Company merges, integration of new software together with legacy systems which need to share data, the need to be compliant with emerging standards to enable and maintain Business to Business (B2B) cooperation, are all forces driving the need of integration. Therefore, companies seek the ability to easily integrate data sources not only available inside but also outside the organization into their information systems.

There are different approaches for data integration. The traditional solution consists in developing specific transformation scripts that transform data in one format to another enabling one application to import data from another one. These transformations are point-to-point or application-to-application, also called ad-hoc, and are not scalable because the number of point-to-point mapping that need to be maintained. Maintenance is a huge problem because if a schema used by one application changes, a lot of mapping must be updated. Clearly, this approach is not a suitable solution. The other approach consists in building a central model representing all the enterprise domain concepts and their relations. The data sources to be integrated are mapped to this central model. Such paradigm is much more scalable than the one introduced previously.

Ontologies are particularly suitable to play the role of the central model. In fact, the conceptualization would be an abstract model of all the enterprise domain concepts. These domain concepts explicitly defined and related to other concepts independently of the underlying application. As stated by Grubber (Grubber, 1993): "An ontology is a formal, explicit specification of a shared conceptualization". This formal specification also enables inference, that is, logical reasoning about concepts. Consequently, it is possible to derive (potentially new) knowledge from given facts. Ontologies, and therefore information, can be shared between applications or business partners because they are not organization specific but domain specific.

3.1.1 Technologies for developing Semantic Web

Two important technologies for developing the Semantic Web are already in place: eXtensible Markup Language (XML) and the Resource Description Framework (RDF). XML lets everyone create their own tags hidden labels such as or that annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the script writer has to know what the page writer uses each tag for. In short, XML allows users to add arbitrary structure to their documents but says nothing about what the structures mean. (Berners-Lee et al., 2001)

The RDF family of standards leverages Uniform Resource Identifiers (URIs) and XML to provide a stepwise set of functionality to represent these relationships and meaning. URIs are a fundamental component of the current Web and are the foundation of the Semantic Web. The Extensible Markup Language (XML) is also a fundamental component for supporting the Semantic Web. XML provides an interoperable syntactical foundation upon which the more important issue of representing relationships and meaning can be built. URIs provide the ability for uniquely identifying resources as well as relationships among resources (Berners-Lee and Miller, 2002).

The Semantic Web, in naming every concept simply by a URI, lets anyone express new concepts that they invent with minimal effort. Its unifying logical language enables these concepts to be progressively linked into a universal Web. This structure opens up the knowledge and workings of human kind to meaningful analysis by software agents, providing a new class of tools by which we can live, work and learn together (Gruber, 1993).

The Semantic Web provides the following advantages: 1) enables communities to expose their data so that a program doesn't have to strip the formatting, pictures and ads from a Web page to guess at the relevant bits of information and 2) allows people to write (or generate) files which explain - to a machine - the relationship between different sets of data (Berners-Lee and Miller, 2002) .

The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing (SW, 2001).

The Web has developed most rapidly as a medium of documents for people rather than for data and information that can be processed automatically. The SW aims to make up for this (Berners-Lee et al., 2001). Some people have said, "Why do I need the Semantic Web? I have Google!" Google is great for helping people finding things, yes! But finding things more easily is not the same thing as using the Semantic Web. It's about creating things from data you've compiled yourself, or combining it with volumes (think databases, not so much individual

documents) of data from other sources to make new discoveries. It's about the ability to use and reuse vast volumes of data (Updegrove, 2005).

The specifications that the SW proposes will enable users to search not only for documents that contain data, but also for the desired data itself, through "semantic" identification and location techniques. The result of implementing these standards will be the creation of a next-generation "Semantic Web." This new Web will be capable of supporting software agents that are able not only to locate data, but also to "understand" it in ways that will allow computers to perform meaningful tasks with data automatically and on the fly that today must be done manually and episodically by computer users (Updegrove, 2005).

The real power of the SW will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine-readable Web content and automated services (including other agents) become available. The SW promotes this synergy: even agents that were not expressly designed to work together can transfer data among themselves when the data comes with semantics (Gruber, 1993). The information on the Web can be defined in a way that can be used by computers not only for display purposes, but also for interoperability and integration between systems and applications. One way to enable machine-to-machine exchange and automated processing is to provide the information in such a way that computers can understand it. This is precisely the objective of the SW – to make possible the processing of Web information by computers. The next generation of the Web will combine existing Web technologies with knowledge representation formalisms (Grau 2004).

SW goal is the development of standards and technologies that allow machines and automatic processes to understand the information expressed on the Web, supporting the discovery of knowledge, data integration and task automation.

By adopting these representations for documents it is possible to develop new ways of: 1) publishing information, 2) searching, 3) data integration and 4) data exchange.

When doing a search, the results will be more precise. On the other hand, it will be simple and possible to integrate data from different sources since it is possible to compare and establish a relation between data with accuracy.

Nowadays, there are a few SW format documents, but they represent a small amount of documents produced. There are older documents that could be traduced to the Semantic Web. The Web is always in permanent updating and it is important to the world to be aware of and clarify the advantages of the SW. The idea is that this will be more than a successful experience for certain domains. It is desirable that the SW reaches a dimension similar to the actual web.

3.1.2 The Impact of Semantic Web

The impact of the SW global dissemination is huge since it allows taking part of the capabilities of the machine processing in a more effective way and with an efficiency never reached by humans.

There are six principles that work as basis for the concept of Semantic Web. They are: 1) everything can be identified by URI, 2) resources and references can have types, 3) it is acceptable to have partial information, 4) there is no need for absolute true; 5) there is place for evolution and 6) minimalist design.

People, animals, places, object and everyting that exists in the physical and imaginable world needs an identifier so that it is possible to refer to it unequivocally. The web already has the concept of identifier for a document or resource, it is known as the URI. As an example for the identification of a place, the city of Helsínquia can be refered in a Semantic Web by the URI of the official Web page.

The documents that we usually find in the Web are thought to be human analysed, without meta-information explaining for what they work for or are related to. When we find a link in the middle of the text, referring to another document, the user uses his intuition to deduct the relationship between the documents, but machines can not.

Resources are connected to other resources in Semantic Web by links for certain URI identifiers. What's new is the fact that resources and links can have types. Types give more expressiveness by associating concepts that can be captured by machines.

Sometimes documents change place and links do not point out to the right resource. The same can happen in the SW too. It is necessary that the applications are conscious against the possibility and can work with the data available. Generally, the information available is partial and is with this that the application has to work with.

As the Web, the Semantic Web is equally affected by cases of incorrect information. The fiability of a document must be analysed by the application that processes the information. This process can be based in the attribution of a context to the assertions found. Each context with assertions is labeled with the author of the assertion.

3.2 ONTOLOGIES

Ontologies provide a conceptualization of a domain that can be shared among people, computer systems and applications (Gruber, 1993). They are usually composed of sets of concepts with relations among them and axioms involving these concepts and relations. The relations and axioms restrict the expression of concepts, therefore restricting their semantic interpretation. Sometimes a distinction is made between concepts and the terms associated with them. Ontologies can have different degrees of formalization, ranging from informal

categorizations such as library catalogs to ontologies written in formal languages with general logic constraints and axioms. Formal ontologies allow the representation of the semantic of domain concepts. They are usually composed of sets of concepts with relations among them and axioms involving these concepts and relations. The relations and axioms restrict the expression of concepts, therefore restricting their semantic interpretation. Sometimes a distinction is made between concepts and the terms associated with them (Martins, 2006).

In the context of the SW, "ontology" is an enabling technology - a layer of the enabling infrastructure - for information sharing and manipulation (Gruber, 2005).

For a particular domain, an ontology represents a richer language for providing complex constraints on the types of resources and their properties. Compared to a taxonomy, ontologies enhances the semantics by providing richer relationships between the terms of a vocabulary. Ontologies are usually expressed in a logic-based language (Cardoso, 2006).

The logic must be powerful enough to describe complex properties of objects. Adding logic to the Web means to use rules to make inferences, choose courses of action and answer questions (Berners-Lee et al., 2001).

Ontologies have been designed for purpose of enabling knowledge sharing and reuse. In that context, an ontology is a specification used for making ontological commitments. For pragmatic reasons, an ontology is written as a set of definitions of formal vocabulary. Although this isn't the only way to specify a conceptualization, it has some nice properties for knowledge sharing among Artificial Intelligence (AI) software (e.g., semantics independent of reader and context). Practically, an ontological commitment is an agreement to use a vocabulary in a way that is consistent with respect to the theory specified by an ontology. Agents are built in order to commit to ontologies (Gruber, 1993)

As stated by Gómez-Pérez and colleagues (Gómez-Pérez et al., 2004), ontologies aim to capture consensual knowledge in a generic way, and that they may be reused and shared across software applications and by groups of people. They are usually built cooperatively by different groups of people in different locations.

Ontologies can be used to increase communication both between humans and computers. The three major uses of ontologies are: 1) to assist in communication between humans, 2) to achieve interoperability and communication among software systems and 3) to improve the design and the quality of software systems (Jasper and Uschold 1999)

To communicate information, we need a medium (writing, speech), a syntax (natural language grammar) and shared semantic symbols (words, verbs, etc.). This holds for people and, also, for computer systems, but with more strict and formal syntax for the latter. The shared semantics corresponds to a model of reality, that is at least partially agreed upon by those who use it to communicate. Ontologies, have their origin in philosophy. Its first uses in computer systems were prompted by the need to share knowledge between large knowledge bases when building expert systems in Artificial Intelligence. In another application, knowledge bases about chemical elements were used in expert systems to detect them. Many other applications in

expert systems have been developed. These large knowledge bases were usually built from scratch and were very costly.

Ontologies were proposed as a way to describe the common knowledge represented in these knowledge bases and allow sharing among them in order to reduce construction costs. Not all ontologies have the same abstraction level. The most common ontologies are about a specific domain and were usually developed with a particular application in mind. Other ontologies capture more general theories, such as the theory of time intervals (Allen, 1984) that are of interest in many domains and not only help reduce the cost of building ontologies but also help to reduce gratuitous divergences among them. Still others, more in line with the original philosophic ideal, attempt to capture everything, at least in a general fashion. These are called "upper ontologies" (IEEE, 2005) and usually provide a top-level taxonomy in which to place all other concepts. Their use helps reduce divergence and eases matching of concepts between different ontologies by means of the concepts common base. Some consider the term ontology to apply to a more broad spectrum than just formal ontologies (ontologies encoded in a formal language), encompassing things like Thesauri and term glossaries. However, in order to be of use to a computer system, the ontology must be encoded in a formal language. There are several formal languages available, like First Order Logic (FOL), Knowledge Interchange Format (KIF) (Sowa, 1999) and OWL (W3C, 2004), among others. Each has different expressive power. An important issue in choosing a formal language is the tradeoff between its expressiveness power and the ability to build efficient inference engines supporting that expressiveness.

3.2.1 Ontologies Structure

As a result of the formal language used to encode ontologies, their structure can vary significantly. There is, however, a basic structure commonly used, formed by the following elements: Concepts, Relations and Axioms.

Concepts are the basic element of an ontology as they will be the object of inference in computer systems. Their definition, in formal terms, is given by the relations in which they participate and by the axioms that involve them. These restrict the possible meanings of concepts, distinguishing them from each other in a useful way for computer systems. In some cases there is a separation between a concept (or relation) and the name that is associated with it. This separation is useful in ontologies that deal with natural language, where words can have multiple meanings and so be associated with different concepts (or relations), and there can be more than one word to describe the same concept (or relation). Another use is in the translation of the ontology into several languages when the ontology is to be used by people from different countries. Most relations present in ontologies are binary relations.

Relations are used to construct taxonomies, which are a common way to organize knowledge. This relation is the base for the use of inheritance (Richmond et al, 1989) an efficient inference mechanism. This allows for the common elements of subtypes to be stored only in the super-type, therefore saving effort and space during knowledge encoding. Another common relation is

the Part-Of relation, used to describe structural decomposition (Varzi, 1996). There are several variations of this relation, differing in properties like, for example, transitivity.

With axioms, restrictions can be applied to both relations and concepts. Axioms are written in some formal logic, typically FOL, allowing the use of the full expressive power of that logic. This increases the expressiveness of the ontology and its semantic value, and also what can be inferred from it.

3.2.2 Purposes of the ontologies

Ontologies serve different purposes. One such purpose, that does not require an inference engine, a knowledge base or even a formal encoding, is to communicate a vision of the world (or of some part of it) between people. For example, they can be used to establish a common vocabulary between working groups from different backgrounds.

In computer systems, ontologies serve several purposes:

- They provide a base structure for knowledge bases and, as mentioned, a reduction in their construction costs by the reuse of already built ontologies and knowledge bases that adhere to them;
- They provide a common vocabulary for communication, either for cooperation between several systems to accomplish some complex task, or to facilitate the interaction between systems of different origins;
- They provide for a means to isolate data from the particular system used to manipulate it, as long as one can build a translator from the system's native storage format to the ontology encoding, enabling interoperability among different systems;
- They can be seen as a way to model the information structure to be used by some software system, during its development.

Ontologies also serve a purpose in the Semantic Web vision (Berners-Lee et al., 2001).They provide well defined and machine processable meaning to information available online. To this end, online resources must be associated with ontologies by means of resource annotation. Annotation involves both the description of the information according to some ontology and the linking between the description and the original resource.

Having this machine processable information available can benefit both search engines and other automated applications. Search engines can use it to improve information retrieval by distinguishing the meaning and context of the search query. It might also be possible, based on the semantic information, to provide a way in which to combine the search results in a meaningful fashion. Automated agents could use ontologies to communicate and the annotated resources to offer online services that add value by combining information and services already available. For example, to aid planning a day in family combining information from different transport companies, museums and restaurants.

3.3 ONTOLOGICAL ENGINEERING

The emergence of the Semantic Web (Berners-Lee, 1999) has caused a growing need for knowledge reuse, and has strengthened its potential at the same time. Therefore ontologies and problem-solving methods (which in some cases are considered as the precursors of Semantic Web Services) are playing an important role in this context.

Ontological engineering has garnered increasing attention over the last few years, as researchers have recognized ontologies are not just for knowledge-based systems—all software needs models of the world, and hence can make use of ontologies at design time (Chandraeskaran et al., 1999).

Ontological engineering encompasses a set of activities conducted during conceptualization, design, implementation and deployment of ontologies. Ontological engineering covers topics including philosophy, metaphysics, knowledge representation formalisms, development methodology, knowledge sharing and reuse, knowledge management, business process modeling, commonsense knowledge, systematization of domain knowledge, information retrieval from the Internet, standardization, and evaluation (Mizoguchi et al., 1995). It also gives us design rationale of a knowledge base, helps define the essential concepts of the world of interest, allows for a more disciplined design of a knowledge base, and enables knowledge accumulation. If we put ontological engineering in the context of other disciplines, many similarities and analogies arise. These similarities allow practitioners to make connections between ontological engineering and other disciplines, to bridge comprehension gaps, and to see known concepts and practices in another light. Desirable qualities for ontologies, such as being decomposable, extensible, maintainable, modular and interfacable, tied to the information being analyzed, universally understood, and translatable, are also desirable for interoperable software components, or even classes of objects in object-oriented design. Practitioners from other fields may use different terminology, but its meanings are often similar. Hence the following question naturally arises: can ontologies practitioners borrow from other disciplines?

3.3.1 Modeling and Metamodeling

Two general disciplines can help to develop ontologies at the specification and conceptualization stage: modeling and metamodeling. In practice, knowledge of these disciplines helps to:

- Organize the knowledge acquisition process;
- Specify the ontology's primary objective, purpose, granularity, and scope;
- Build its initial vocabulary and organize taxonomy in an informal or semiformal way, possibly using an intermediate representation.

3.3.1.1 Modeling.

Ontologies are specific, high-level models of knowledge underlying all things, concepts, and phenomena. As with other models, ontologies do not represent the entire world of interest. Rather, ontology designers select aspects of reality relevant to their task (Valente et al., 1999). In the domain of books, for example, the ontology designer selects one set of book attributes when developing the ontology of a library, and quite a different set when developing the ontology of bookbinding. All models follow principles and constraints, which are called concept relations and axioms. Although there are different ways to represent ontologies, ontological engineers most frequently use hierarchical modeling (at least at the conceptualization level (Devedzic and Radovic, 1999 and Lenat, 1995). They often represent concept hierarchies and taxonomies in layers, and use graphs to visually enhance the representation. The layers in ontology representation range from domain-independent (core) to task- and domain-specific. Thus, ontologies contain knowledge of appropriate hierarchical and/or layered models of the relevant world.

3.3.1.2 Metamodeling

Conceptualizing and specifying ontologies have a strong metamodeling flavor. A metamodel, or conceptual model of a modeling technique, improves the rigor of different but similar models (Fowler and Scott, 1997). Ontologies do the same for knowledge models. Without ontologies, knowledge bases representing knowledge of the same domain are generally incompatible even if they use similar knowledge models.

The good thing about using such metamodeling is that the usefulness of any specific model is not sacrifice. The ontology simply provides the skeleton for the corresponding models of the domain knowledge.

Generally, an ontology is a metamodel describing how to build models. Its components— the concepts it defines and the relations between them—are always (re)used as building blocks when modeling parts of the domain knowledge. When developing a practical software system, it helps that the tools have some built-in knowledge— a metamodel or an ontology—of the models we deploy. The metamodeling function of the ontology makes the tools intelligent. Many potentially useful parallels exist between ontological engineering and software engineering disciplines such as software architectures and software patterns, but few have been discussed explicitly or used in practical developments. Many practitioners understand similarities between ontological engineering and the object-oriented paradigm, and similarities between phases of the ontology development and software development processes, especially when looking at special-purpose software tools for developing ontologies, such as Ontolingua from Stanford University, or ODE from Polytechnic University of Madrid (Lopez et al., 1999). But practitioners can benefit from knowing more about such useful parallels. Certain software engineering disciplines and issues rarely discussed by ontology researchers can help advance ontological engineering: software architectures, programming languages and compilers, traditional software

engineering, object-oriented analysis and design, design patterns, and component- based software engineering.

3.3.2 Software architectures

Suppose you are discussing the basics of ontological engineering with a software engineer, whose field involves designing and specifying the overall system structure and underlying organization. Conveying that ontologies are architectural armatures for building knowledge bases, models, and software architectures is probably one of the best ways to help such an individual grasp the basics of ontologies. Architectural style, an important concept in the field of software architecture, characterizes a family of systems related by shared structural and semantic properties. Mary Shaw describes several common architectural styles (Shaw, 1995). A style typically defines a vocabulary of design elements, design rules (constraints) for compositions of those elements, semantic interpretation of design element compositions, and analyses on systems built in that style. Many successful designs can share a style. Styles contain condensed skeletons of the architectural knowledge gained by experienced software designers, and provide a means to reuse that knowledge. For example, layered style is suitable for applications involving distinct classes of services that can be arranged hierarchically. Designers often define layers for: basic systemlevel services, utilities appropriate to many applications, and specific application tasks. Similarly, some ontologies structure knowledge in layers to separate the user-specific knowledge from the core (and more reusable) knowledge (Mizogucgi, 1998) and (Valente et al., 1999). Other architectural styles that help define ontological engineering solutions include pipeline and data abstraction.

3.3.3 Programming languages and compilers

Special-purpose languages/tools for implementing ontologies, such as Ontolingua, CycL, and LOOM, use a frame-based formalism, a logic-based formalism, or both (Fridman-Noy et al., 1997), (Lenat, 1995) and (Lopez et al., 1999) and (Valente et al., 1999). For example, Ontolingua is a frame-based language that uses KIF (Knowledge Interchange Format), a language for publication and knowledge communication that has notation and semantics of an extended version of first-order predicate calculus. It enables writing knowledge-level specifications, independent of particular data or programming languages, and translating knowledge bases from one specialized representation language into another. Several other languages/tools for building ontologies also use such a translation approach, which enables the building of ontologies directly at the knowledge level.

This approach eliminates the need to master implementation languages and use translators to translate from the knowledge level to the implementation level. But there are problems with this approach. First-order logic is rather restricted, to enable painless translation of rich and often ill-structured or unstructured knowledge-level specifications into well-formed predicate calculus expressions. Also, there are reports of problems using specific translators. For example, when using the Ontolingua translator to produce the equivalent Loom ontology of time from a prebuilt

Ontolingua-based ontology, the resulting translation had value only as a first draft (Valente et al., 1999). Extensive manual adaptations of the translated ontology were necessary in order to make it fully applicable.

Practitioners should note more recent languages and tools make extensive use of techniques and tricks from the compiler theory to improve the quality and the capabilities of the translation process. For example, the ODE environment uses a generic translator that allows the user to specify the ontology in a user-oriented internal representation and translate it automatically into the target language (in ODE's case, into Ontolingua) (Lopez et al., 1999). The translator uses a grammar to declaratively express the conceptual model (the internal representation) in the Backus Naur Form (BNF) form. For each type of valid definition in the language, there is a table that relates the terms used in the transformation rules to the terms employed in the conceptual model. It is easy to build a new translator by merely changing the rules that identify the transformation rules of the terms to be generated, and changing the appropriate table relating the conceptualization to the implementation. It is also useful to consider other programming languages besides special-purpose languages from the perspective of ontological engineering. In addition to defining a general ontology of a programming language, with concepts like identifier, reserved word, and construct, one may abstract an ontological skeleton from any programming language. This is actually a hint for practitioners that many ontologies implicitly exist in programming languages and should not be reinvented. For example, Java has at least two obvious but rarely noted parallels with ontological engineering concepts:

• Classes from Java class libraries make up a hierarchy (with the Object class on top) that can be viewed as an extremely well elaborated ontology;

• Java bytecodes are actually an "interlingua" any Java interpreter can understand, in a common interchange format that makes Java fully interoperable and platform- independent—the same idea behind KIF and other ontology-related languages.

3.3.4 Traditional software engineering

Since the AI community develops ontologies, and uses special-purpose tools and languages to do so, many think of ontologies as a trend involving sophisticated methodology and technology. But an ontology is always about entities and relationships, and often methodology from traditional software engineering, such as using the ER model, top-down decomposition strategy, and structured system analysis, are used to represent it. For example, the Methontology framework for developing ontologies (Lopez et al., 1999) proposes a close relative of the traditional waterfall process of software development for an ontology development lifecycle. Moreover, the entire chemicals ontology developed using the Methontology framework is stored in a relational database, which can encode its ontology in its data dictionary (Mizoguchi, 1998). Fridman-Noy and Hafner discuss examples of using online lexical reference systems and electronic dictionaries as general ontologies (Fridman-Noy, 1997).

All design criteria for ontologies, such as clarity, extensibility, coherence, and minimal encoding bias also represent design criteria for software systems modules.

Ontology researchers and developers can explore a large variety of iterative and incremental traditional software development methodologies for new ideas in ontological engineering.

The processes of ontology development (Mizoguchi, 1998), (Lopez et al., 1999 and Valente et al., 1999) nearly coincide with those of object-oriented analysis and design (Fowler et al., 1997) and (Szyperski, 1998). In both cases, it is important to assemble the domain vocabulary in the beginning, often starting from the domain's generic nouns, verbs, and adjectives. Object-oriented analysis stresses different aspects than ontological analysis (Mizoguchi, 1998), but parallels are obvious. The result of object-oriented analysis is a draft of the domain ontology relevant to the application (although analysts don't call that result an ontology). And as object-oriented designers define classes, objects, hierarchies, interface functions, and system behavior, ontological engineers use intermediate representations such as semantic networks, graphs, and tables to design hierarchies and other concept relationships. Both types of specialists use templates to specify product details (Fowler and Scott, 1997) and (Lopez et al., 1999). Classes can be merged or refined, as with ontologies. Class libraries and previous design specifications often provide reuse in object-oriented design, as do previously encoded and publicly available ontologies.

One area of ontological engineering requiring additional efforts involves developing a generally accepted notation for representing ontologies. Software engineers have used several different notations in object-oriented design over the past decade, but all have converged to the Unified Modeling Language (UML) notation (Fowler and Scott, 1997) which provides a metamodel of object-oriented design. It defines graphical notation for representing classes, objects, and their relationships in four different views (logical, use-case, component, and deployment), covering all practical aspects of object oriented design. It would be nice if ontological engineers had a standard notation that everyone accepted, understood, and used in practice. Unified graphical representation for such a meta-language could help construct visually rich and easy-to-use tools (Valente et al., 1999), and would increase knowledge reuse at the design level, but unfortunately most ontology developers currently use their own notation.

From the practitioner's perspective, important differences exist between ontological engineering and object-oriented analysis and design. "Ontological" means taking a knowledge-level stance in describing a system (Chandrasekaran et al., 1999) while "object-oriented" largely refers to the means of design and implementation. In a semantic-based information retrieval system, for example, ontologies specify the meaning of the concepts to be searched for, while in the object-oriented design of such a system, ontologies represent the domain models. Object-oriented design languages like UML offer explicit design methodology and notation for all design artifacts, but ontological and metamodeling principles are only implicit in those languages (Mizoguchi, 1998). In other words, an ontology is what can be abstracted at the knowledge level from the corresponding class diagrams, object diagrams, and use-case diagrams, represented in any object-oriented notation such as UML. The role of the ontology is to convey and explicitly specify domain concepts, terms, definitions, relations, constraints, and other semantic contents that object-oriented analysis and design should rely on and support. Design patterns, described

as "simple and elegant solutions to specific problems in object-oriented software design" (Gamma et al, 1995), provide a common vocabulary for designers to communicate, document, and explore design alternatives. They contain the knowledge and experience underlying many redesign and recoding efforts to achieve greater software reuse and flexibility. Although design patterns and ontologies are not identical, practitioners should be aware these two concepts overlap, and software patterns may be used along with other sources of ontology design in practical developments. Both concepts involve vocabularies, knowledge, and "architectural armatures," and describe concepts at the knowledge level. Ontologies are more commonsense-oriented, while design patterns are more concrete. But besides activities such as software design, software patterns can also involve abstract activities such as organizational and analysis patterns (Deveddzic and Radovic, 1999) and (Gammaet al., 1995). One can draw an analogy between libraries of ontologies and catalogues of software patterns. Design pattern catalogues are not ready-to-use building blocks as are ontologies from libraries, but efforts are ongoing to make them ready-to-use blocks. It doesn't take a hard mental shift to view ontologies as abstract patterns, or knowledge skeletons of some domains. Likewise, it is easy to understand software pattern templates as knowledge of how software pattern ontologies may look.

A long-term objective of ontological engineering is to build libraries of reusable knowledge components and knowledge-based services that can be invoked over networks. Similarly, the component-based software engineering field is struggling to develop repositories of reusable, pretested, interoperable, and independently upgradable software components that enable plug-andplay design and software development. These objectives necessitate designing systems from application elements constructed independently by developers using different languages, tools, and computing platforms (Szyperski, 1998). Can ontologies be components and vice versa? Ontologies are conceptually more abstract than components, but it seems components can be parts of ontologies. It is also possible to develop a component that fully corresponds to an ontology. Different domains and ontologies can share components from repositories.

Ontologies should be, in a sense, a basis for designing and developing interoperable software components in practice, since they can precisely define the semantics of components and their parts, as well as the types of relations and communication between software components.

Ontology engineers in the context of computer science are more interested in how ontologies can be used to represent reusable and sharable pieces of domain knowledge and how they can be used in applications. In this context, ontologies are reusable and sharable artifacts that have to be developed in a machine interpretable language (Grubber, 1993) (Studer, Benjamins and Fensel, 1998).

As we have seen before, different knowledge representation formalisms (and corresponding languages) exist for the formalisation (and implementation) of ontologies. Each of them provides different components that can be used for these tasks. However, they share the following minimal set of components: classes, relations and instances.

Classes represent concepts, which are taken in a broad sense and are usually organised in taxonomies through which inheritance mechanisms can be applied.

Relations represent a type of association between concepts of the domain. They are formally defined as any subset of a product of n sets. Ontologies usually contain binary relations. The first argument is known as the domain of the relation, and the second argument is the range. Relations can be instantiated with knowledge from the domain.

Instances are used to represent elements or individuals in an ontology.

To sum up, ontological engineering corresponds to the set of activities that concern the ontology development process, the ontology life cycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them.

3.4 ONTOLOGY LANGUAGES

One of the key decisions to take in the ontology development process is to select the language (or set of languages) in which the ontology will be implemented.

In this section, we overview the most renowned Semantic Web languages, and outline their evolution trends.

3.4.1 RDF(S): RDF and RDF Schema

RDF (Lassila and Swick, 1999) stands for Resource Description Framework. It is a general-purpose language for representing information in the Web. RDF is a datamodel for objects ("resources") and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax. RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.

RDF schema (RDFS 2004) is a formal description of eligible RDF expressions. In particular, a schema defines the properties of the resource (e.g., title, author, subject, size, colour, etc.) and the kind of resources being described (e.g., books, Web pages, people, companies, etc.). RDF Schema became a W3C Proposed Recommendation on 15 December 2003, and a W3C Recommendation on 10 February 2004. The main latest updates and improvements accomplished within the context of this language are related to grammar updates, ways to link the XML serialization to formal semantics, tutorials on how to use RDF in applications, syntax and test cases (Zhdanova and Keller, 2005).

3.4.2 OIL

Ontology Interchange Language and Ontology Inference Layer (OIL) (Horrocks et al., 2000; Fensel et al., 2001) are built on top of RDF and RDFS, employing their constructs to a large extent, in order to maintain backward compatibility. OIL provides modelling primitives used in framebased and Description Logic oriented ontologies, coming along with a simple and clean semantics. It has a syntax definition using web standards such as RDF(S) and XML(S).

OIL unifies three important aspects provided by different communities: (1) formal semantics and efficient reasoning support as provided by Description Logic, (2) epistemologically rich modelling primitives as provided by the Frame-based community, and (3) a standard proposal for syntactical exchange notations as provided by the Web community (Zhdanova and Keller, 2005).

OIL is not an evolving language any longer. The natural continuer of the work carried out by the OIL team was DAML+OIL, a joint effort of the American and European ontology communities for the Semantic Web (Zhdanova and Keller, 2005)..

3.4.3 DAML+OIL

DARPA Agent Markup Language (DAML) +OIL (Horrocks and van Harmelen, 2001) is an ontology language specifically designed for the Semantic Web, created as a joint effort of the American and European ontology communities for the Semantic Web. DAML+OIL is an ontology language specifically designed for use on the Web; it exploits existing Web standards (XML and RDF), adding the familiar ontological primitives of object oriented and frame based systems, and the formal rigor of a very expressive description logic. The logical basis of the language means that reasoning services can be provided, both to support ontology design and to make DAML+OIL described Web resources more accessible to automated processes (HORROCKS, 2002). DAML+OIL exploits existing de-facto Web standards (XML and RDF), adding ontological primitives of object oriented and frame-based systems, and the formal rigor of expressive description logic. As an ontology language, DAML+OIL is designed to describe the structure of a domain. DAML+OIL takes an object-oriented approach, with the structure of the domain being described in terms of classes and properties, and the set of axioms that assert characteristics of these classes and properties. Similar to OIL, DAML+OIL is not an evolving language. The latest DAML+OIL drafts are dated from December 2001 (Zhdanova and Keller, 2005).

3.4.4 OWL

OWL (Dean and Schreiber, 2003) is the web ontology language, developed by the W3C Web Ontology (WebOnt6) Working Group. OWL is mainly based on OIL and DAML+OIL, and therefore the main features of OWL are very similar to those of OIL. OWL includes three sub languages called: OWL is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs

to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

OWL-Lite. Roughly consists of RDFS plus equality and 0/1- cardinality. It represents a migration path from other taxonomies. It is intended for classification hierarchies and simple constraints. It should be kept as simple as possible in order to facilitate the tool development.
OWL DL. Contains the language constructs but with hierarchy restrictions. It provides computational completeness and decidability, and has a maximum expressive power within

DL Description Logics fragment.

OWL Full. Composed by the complete vocabulary interpreted more broadly than in OWL DL. The language incorporates maximum expressive power and syntactic freedom, and offers no computational guarantees.

Besides the DAML+OIL style RDF syntax, the OWL specification also includes an abstract syntax, which provides a higher level and less cumbersome way of writing ontologies.

OWL became a W3C Proposed Recommendation on 15 December 2003, and a W3C Recommendation on 10 February 2004. (Zhdanova and Keller, 2005).

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not (Gómez-Pérez et al., 2004).

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete

OWL Full implementations do not currently exist (Gómez-Pérez et al., 2004).

3.5 APPLICATION OF SEMANTIC WEB

One of the most promising application area of Semantic Web technology is knowledge management (Fensel et al., 2002).

3.5.1 Knowledge Management

3.5.1.1 Semantic Web Enabled Knowledge Management

Efficient knowledge management has been identified as key in maintaining the competitiveness of organizations. Traditional knowledge management is now facing new problems triggered by the web; information overload, inefficient keyword searching, heterogeneous information integration and geographically-distributed intranet problems, to name but a few. These problems will be tackled by the modern technology known as Semantic Web Technology (Fensel, 2001).

3.5.1.2 Ontologies and P2P

Peer-to-Peer computing and the Semantic Web can actually be combined to support decentralized environments where participants can maintain individual views of the world, while sharing knowledge in ways such that administration efforts are low but knowledge sharing and finding is easy. Key to the success of combining Peer-to-Peer solutions with Semantic Web technologies is the use of Emergent Semantics. Emergent Semantics builds on lightweight and/or heavyweight ontologies that different individuals, departments or organizations have created. It considers the overlap between ontology definitions and the use of concepts and relations with actual data in order to extract shared ontologies for sets of individuals or groups of people. Intelligent tools will use such definitions to ensure that knowledge will be appropriately structured, so that it can be easily refound. Knowledge Management can occur in a distributed fashion without the overheads of central administration.

3.5.2 Electronic Commerce

Bringing Electronic Commerce to its full potential requires a Peer-to-Peer (P2P) approach. Anybody must be able to trade and negotiate with anybody else. However, such an open and flexible method for electronic commerce has to deal with many obstacles before it becomes a reality.

Mechanized support is needed in finding and comparing vendors and their offers. Currently, nearly all of this work is done manually which seriously hampers the scalability of electronic commerce. Semantic Web Technology can make it a different story: machine-processable semantics of information allow the mechanization of these tasks.

Mechanized support is needed in dealing with numerous and heterogeneous data formats. Various "standards" exist on how to describe products and services, product catalogues and business documents. Ontology technology is required to define such standards better and to map between them. Efficient bridges between different terminologies are essential for openness and scalability.

Mechanized support is needed in dealing with numerous and heterogeneous business logics. Again, various "standards" exist that define the business logic of a tradingpartner. Mediation is needed to compensate for these differences, allowing partners to cooperate properly.

The three main challenges in applying semantic web technology to Electronic commerce: 1) efficient alignment of ontologies, 2) versioning of ontologies, and 3) population of ontologies.

3.5.2.1 Ontology Mappings: Integrating Business Documents

Modern document integration tasks impose a number of requirements on the integration technology such as the following:

• The technology must transform the documents with a speed compatible with the databases producing the documents. This limits the usage of logic-based transformation techniques;

• It must allow fast plugging-in of new documents without programming effort that indicates that XSLT4 alone cannot satisfy the business integration tasks;

• Different documents and vocabularies must be aligned via their mediating conceptual models that capture semantic relations between vocabulary concepts and document elements, i.e., the hierarchy of terms and their equivalence;

• The integration model must be driven by a common process modeling ontology and a shared business integration ontology.

Accordingly, to perform the integration we need to develop ontological models for the following integration sub-tasks:

• Vocabularies that mostly represent sets of terms, sometimes enriched with topology;

• Business documents that mostly represent part-breakdown of documents into elements and their ontological models that contain a shallow concept hierarchy but a number of constraints on element values;

• Processes ontologies that contain a limited number of terms corresponding to timepoints, activities, objects etc., but a rich set of temporal axioms.

All the individual ontologies must be mapped to the mediating ontology that specifies the shared semantics of the concepts used by the integration service.

The business integration technology proposed in (Omelayenko and Fensel, 2001) assumes that XML documents might first be 'lifted up' to their RDF data models (the process known in the Semantic Web area as document annotation). Then different private RDF data models are

mapped to the shared mediating data model enriched with different constraints and formal specification of shared semantics of the concepts.

An RDF mapping technology RDFT is now being developed to represent, reuse and execute these mappings (Omelayenko et al, 2002). Specifically, RDFT provides an integration architecture, a mapping meta-ontology that specifies the mapping constructs called bridges, and the technology for map interpretation and translation to XSLT.

Concisely, the RDFT architecture assumes that three tasks (vocabulary, document and process integration) are processed separately. Each specific enterprise must separately align the vocabularies, document formats and processes with the mediating ontology. The mediating process model guides the whole integration process. Process, document and especially vocabulary maps can be frequently reused, which increases the overall efficiency of the integration. Accordingly, the whole transformation chain, from the source XML serialization and RDF model via the mediating format to the target data model and XML serialization, can be represented as a graph allowing backward-chain processing and efficient compilation to XSLT.

Although the RDFT technology was developed to solve quite a specific task, it is easy to see that basically the same tasks frequently reoccur in other Semantic Web applications. To implement the main idea of the Semantic Web of enriching Web data with machine processable semantics, we need to create conceptual models of the documents and then link them to shared domain theories (ontologies) and process these documents according to them. Hence, the business integration technology provides a solution to a typical Semantic Web task and it might be reused for other applications.

3.5.2.2 Ontology Versinoning: Content Standards

Content standards form an important enabler for electronic commerce. They specify a standard hierarchy of products and services which can be used by companies to classify their actual products. This hierarchy can be considered as a simple ontology that specifies a consensus on the products that exist. Different companies that use the same content standard can easily communicate with respect to their products. There are several standard classifications in use, e.g. UNSPSC5, which addresses a general and broad domain of products and services, RosettaNet6, which is targeted at the IT industry, and e@Class7, another broad standard that originates from Germany.

A serious threat for electronic commerce is the high change rate of the classification hierarchies and the way in which those changes are handled.

Although some parts of the UNSPSC (UNSPSC, 2007) schema might be more stable than other parts, it is clear that this amount of changing cannot be ignored. Such a high change rate can quickly invalidate a lot of the actual classifications of products. For example, the product "Binding elements" in version 8.0 is removed from the standard and three new products are added in version 8.1 ("Binding spines or snaps", "Binding coils or wire loops", and "Binding combs or strips"). This means that all products that were classified as "Binding elements" are

unclassified under the new version. This is a serious problem because of the high cost of producing the right classifications for products. Moreover, if companies use local extensions of the standard, they have to adapt those extensions to new versions too. A versioning mechanism that allows partly automatic transformation of data between content standard versions is essential. An effective versioning methodology should take care of the different types of changes in ontologies, as these might have different effects on the compatibility of data that is described by them (Klein and Fensel, 2001). An analysis of differences between several versions of content standards has yielded the following list of typical changes: class title changes, additions of classes, relocations of classes in the hierarchy (by moving them up or down in the hierarchy, or horizontally), relocation of a whole sub-tree in the hierarchy, merging of two classes (in two variants: two classes become one new class, or one class is appended to the other class), splits in classes, and pure deletions. However, current versioning techniques for content standards are often quite simple. In UNSPSC, for example, all changes are encoded as either additions, dele tions or edits (title changes). This means that the relocation of a sub-tree is specified as a sequence of "delete a list of classes" and "add a list of classes".

Semantic Web techniques can help to cope with these versioning problems. Current work on ontology versioning builds upon earlier work in database schema versioning (Roddick, 1995). Although these two are similar to some extent, ontology versioning has some characteristics that make it more complex than database schema versioning (Noy and Klein, 2003). For example, the distributed and decentralized nature of ontologies makes every coordination of changes impossible. On the other hand, the richness of the data model and the semantics that are often incorporated in ontologies, might help to find and resolve conflicts between versions. An important ontology versioning technique is the ability to compare versions of ontologies and highlight the differences. This allows changes in ontologies to be found, even if they have occurred in an uncontrolled way, i.e., possibly generated by different people in an unknown order. Another technique for ontology versioning is the specification of the intention and semantics of changes. For example, some changes are corrections of mistakes, while others represent a change in the world.

Different intentions have different consequences for the interpretation of the effects of changes. Semantics of changes specify the intended logical consequences of each change, for example that a new version of a class is a more specific than another version. Both techniques are useful for content standard versioning. Ontology comparison techniques can help companies to find and describe the differences between new versions of the standards and the old versions that were used to classify data. Descriptions of the semantics of discovered changes can facilitate the transformation of data classification. For example, in the most trivial case they can specify that a new version is a combination of two other classes; all products that were classified under the old classes can then be classified under the new class. More complicated specifications of the logical consequences, possibly with approximations, will further decrease the negative effects of the evolution of content standards.

3.5.2.3 Ontology Instantiation: GoldenBullet

Finding the right place for a product description in a standard classification system such as UNSPSC is not at all a trivial task. Each product must be mapped to the corresponding product category in UNSPSC to create the product catalog. Product classification schemes contain huge number of categories with far from sufficient definitions (e.g. over 15,000 classes for UNSPSC) and millions of products must be classified according to them. This requires tremendous labor effort and the product classification stage takes altogether up to 25% of the time spent for content management. Because product classification is so expensive, complicated, time-consuming and error-prone, Content Management needs support in automation of the product classification process and the automatic creation of product classification rules.

GoldenBullet is a software environment targeted at supporting product classification according to certain content standards (Ding et al., 2002a and Ding et al., 2002b). It is currently designed to automatically classify products based on their original descriptions and existing classification standards (such as UNSPSC). It integrates different classification algorithms from the information retrieval and machine learning areas and some natural language processing techniques to pre-process data and index UNSPSC so as to improve the classification accuracy.

3.5.3 Web Services

Semantic Web technology is still in its early stages. We are focusing on building its basic - and mostly static - infrastructure. The next step will be to produce active components that use this infrastructure to offer users intelligent services. Web services aim to support information access and e-business. Examples include UDDI, a repository for describing vendors, products and services. It uses WSDL to describe its entries, and SOAP as a protocol to define how they can be accessed. At present, none of these service description elements are based on semantic web technology. As a result, it requires tremendous human effort to perform such tasks as: searching for vendors, products and services; comparing and combining products; forming coalitions of vendors, etc. Semantic web-enabled services can provide a higher level of service by mechanizing many of these aspects. Steps in this direction are being taken by projects such as DAML and Ibrow. Within DAML, a service description language called DAML-S (Ankolenkar et al., 2001) has been developed.

This language allows formal competence descriptions that enable automatic inference as a means of selecting and combining services.

Web services described by WSDL are individual message exchanges. They can be synchronous or asynchronous one-way messages between a sender and a receiver or a pair of messages following a request/reply pattern between a sender and a receiver.

While these two patterns are sufficient in many cases, they are insufficient for more complex message exchange patterns (called public processes) like a purchase order (PO) and purchase

order acknowledgment (POA) exchange whereby the PO message as well as the POA message are acknowledged individually by low-level message acknowledgements confirming the receipt of the message. WSDL is not able to define these public processes and languages like XLANG (Thatte, 2001), WSFL (Leymann, 2001), WSCL (Banerji et al., 2001), BPML (Arkin, 2001) and BPSS (Business Process Project Team, 2001) are proposed for their definition. RosettaNet13 provides many domain-specific public processes (called Partner Interface Processes) as standard.

If trading partners try to match their complex public processes in order to conduct business with each other they might encounter a mismatch of their public processes. For example, one trading partner expects message acknowledgments but the other trading partner does not provide them. The above-mentioned languages do not support any compensation at all for these public process mismatches.

In a peer-to-peer environment no third party mediator can be asked to compensate for the process mismatches. The trading partners have to do the compensation themselves in their environments. (Bussler, 2001) presents initial work on public process mismatch compensation through the concept of process binding. Through process binding it is possible to generate additional messages, consume superfluous messages as well as change the message exchange order. Through these possibilities the mismatches can be compensated for. Presently, the process binding compensating the mismatches has to be done manually. To implement the vision of mechanized support of public process integration, the appropriate concepts as well as an approach have to be developed.

The easy information access based on the success of the web has made it increasingly difficult to find, present and maintain the information required by a wide variety of users.

In response to this problem, many new research initiatives and commercial enterprises have been set up to enrich available information with machine-understandable semantics. This Semantic Web will provide intelligent access to heterogeneous, distributed information, enabling software products to mediate between user needs and the information sources available.

Web Services deal with an orthogonal limitation of the current web. Currently, the web is mainly a collection of information but does not yet provide support in processing this information, i.e., in using the computer as a computational device. Web services can be accessed and executed via the web. However, all these service descriptions are based on semi-formal natural language descriptions.

Therefore, the human programmer needs to be kept in the loop and the scalability as well as economy of web services are limited. Bringing them to their full potential requires their combination with semantic web technology. This technology will provide mechanization in service identification, configuration, comparison and combination. Semantic Web enabled Web Services have the potential to change our life to a much higher degree than the current web already has. Bussler (Bussler, 2001) identifies the following elements necessary to enable efficient inter-enterprise execution: public process description and advertisement; discovery of services; selection of services; composition of services; delivery, monitoring and contract

negotiation. Without mechanization of these processes, internet-based e-commerce will not be able to provide its full potential in economic extensions of trading relationships.

4 TOOLS TO DEVELOP ONTOLOGIES

There are several ontology development tools for domain modeling, for building knowledge base systems, for ontology visualization, for project management or other modeling tasks. Many of the tools are research prototypes that have been built for a particular project or for an Institute/University. There has been a significant growth in the number of ontology technologies products.

After studying Michael Deny's Survey on Ontology Tools and reading the paper The Hitchhiker's Guide to Ontology Editors of Loredana Laera and Valentina Tamma we decided to do an update of the tools that are available. Some of the tools described in the Michael Deny's Survey either were no longer available (the project has finished) or have been improved. There are also new tools and new languages since there are new projects that demand so. In composing the list shown on table 7, we have selected the tools that comprise some of the following features: are robust and ready to be used; free and open source; provide support to most of the activities involved in the ontology development process and ontology practice; support Resource Description Framework (RDF), Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL); offer collaborative environment; provide multiple ontology environment; offer server-based environment with support for consistency checking; offer easy-to-use functionality for visual creation and editing; offer a query builder; support a methodology; support editing formal axioms and rules; support the growth of large scale ontologies; support versioning; promote interoperability; has a reasoner; has a graphical view; promotes easy and fast navigation between concepts; has tutorial support; and offers Plug-ins. Table 7 overviews the most importante features identified previously.

We have chosen the following tools: Protégé; OntoEdit; DOE (Differential Ontology Editor); IsaViz; Ontolingua; Altova SemanticWorks 2006; OilEd; WebODE; pOWL and SWOOP.

Protégé is one of the most widely used ontology development tool. It is free and open source. It is an intuitive editor for ontologies and there are plug-ins available to carry out some of the tasks for building an ontology.

OntoEdit is an ontology editor that integrates numerous aspects of ontology engineering. OntoEdit environment supports collaborative development of ontologies.

Doe is a simple prototype developed with Java that allows users to build ontologies according to the Bruno Bachimont proposed methodology.

IsaViz is a visual environment for browsing and authoring RDF models as graphs.

Ontolingua was built to ease the development of ontologies with a form-based Webinterface.

Altova Semantic Works is a commercial visual Semantic Web editor that offers easy-to-use functionality for visual creation and editing. It can be downloaded for a 30 day-free evaluation period.

OilEd is an editor that allows the user to construct and manipulate DAML+OIL (DAML- DARPA Agent Markup Language; OIL-Ontology Inference Layer) and OWL ontologies and which uses a reasoner to classify and check consistency of ontologies. It is provided free of charge.

WebODE is the web counterpart for ODE (Ontology Design Environment). It has support for multiple-users. This editor gives support to the methodology Methontology.

pOWL is an open source ontology management tool in a collaborative Web enabled environment.

SWOOP is a Web-based OWL ontology editor and browser. This editor has default plug-ins for different presentation syntax for rendering ontologies.

The main purpose of this chapter is to give an overview of some of the ontology tools available at the present time and to identify the tools that support methodologies for ontology development. This way, if you are starting out on an ontology project, the initial step is to find a suitable ontology editor.

4.1 PROTÉGÉ

Protégé (Noy et al., 2001) is one of the most widely used ontology development tool that was developed at Stanford University. Since Protégé is free and open source, it is supported by a large community of active users. It has been used by experts in domains such as medicine and manufacturing for domain modeling and for building knowledge-base systems. Protégé provides an intuitive editor for ontologies and has extensions for ontology visualization, project management, software engineering and other modeling tasks.

In early versions, Protégé only enabled users to build and populate frame-based ontologies in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consisted of a set of classes organized in a subsumption hierarchy, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes. Protégé editor included support for classes and class hierarchies with multiple inheritance; templates and slots; predefined and arbitrary facets for slots, which included

permitted values, cardinality restrictions, default values, and inverse slots; metaclasses and metaclass hierarchy.

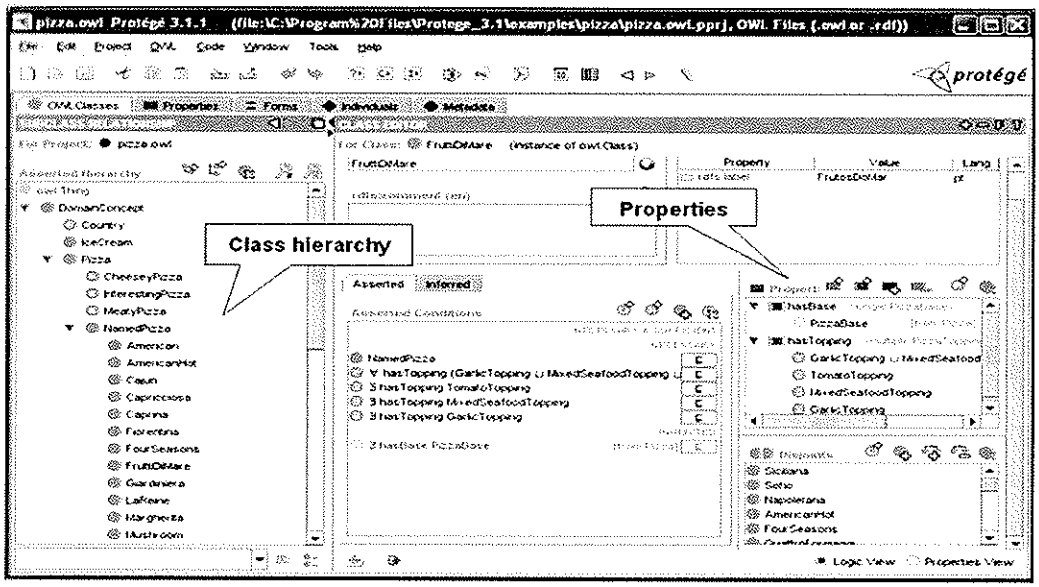

Figure 5: Protégé editor

While the first architecture of Protégé was based on frames, in 2003 it has been extended to support OWL. This extension has attracted many users captivated by the Semantic Web vision. The OWL plug-in extends the Protégé platform into an ontology editor for the OWL enabling users to build ontologies for the Semantic Web. The OWL plug-in allows users to load, save, edit and visualize ontologies in OWL and RDF. It also provides interfaces for Description Logic Reasoners such as Racer.

Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and Extended Mark-up Language (XML) Schema. The current Protégé version can be used to edit classes and their characteristics, to access reasoning engines, to edit and execute queries and rules, to compare ontology versions, to visualize relationships between concepts, and to acquire instances using a configurable graphical user interface. Protégé is a tool installed locally in a computer and does not allow collaborative editing of ontologies by groups of users.

Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-base tools and applications. Protégé is based on Java and provides an open-source API to develop Semantic Web and knowledge-base stand-alone applications. External Semantic Web applications can use the API to directly access Protégé knowledge bases without running the Protégé application. An OWL API is also available to provide access to OWL ontologies. Its extensible architecture makes it suitable as a base platform for ontology-based research and development projects. Protégé also includes a Programming Development Kit (PDK), an important resource for programmers that describe how to work directly with Protégé APIs and illustrates how to program plug-in extensions for Protégé.

Several plug-ins are available. For example, JSave (http://protege.stanford.edu/plug-ins/jsave/) is an application plug-in to generate Java class definition stubs for Protégé classes and Protégé

Web Browser is a Java-based Web application that allows users to share Protégé ontologies over the Internet. The WordNet plug-in (http://protege.stanford.edu/plug-ins/wordnettab/wordnet_tab.html) provides Protégé users an interface to WordNet knowledge base. Users can easily annotate a Protégé knowledge base using information from WordNet database. The information in WordNet can be searched by name and then be used to annotate ontologies with terms, concept IDs, synonyms, and relations.

The XML Schema (http://faculty.washington.edu/gennari/Protege-plug-ins/XMLBackend/XMLBackend.html) is a backend plug-in that transforms a Protégé knowledge base into XML. The plug-in generates an XML Schema file describing the Protégé knowledge model and an XML file where the classes and instances are stored. The UML plug-in (http://protege.stanford.edu/plug-ins/uml/) is also a backend plug-in which provides an import and export mechanism between the Protégé knowledge model and the object-oriented modeling language UML. To enable the exchange of ontologies and UML class diagrams, the UML plug-in uses the standard format for UML diagram exchange, XMI, which is supported by major CASE tools. The use of the XMI standard enables users to work with Protégé in combination with Software Engineering tools and Integrated Development Environments.

The DataGenie (http://faculty.washington.edu/gennari/Protege-plug-ins/DataGenie/index.html) is an import/export plug-in that allows reading and creating a knowledge model from relational databases using JDBC. Users can select a proper subset of a relational database to be converted into Protégé classes. Typically, during the conversion, tables become classes and attributes becomes slots. The Docgen (http://protege-docgen.sourceforge.net/) is also an import/export plug-in that allows users to create reports describing Protégé knowledge bases or ontologies. Classes, instances and documentation can be exported to various output formats such as HTML, Dynamic Hypertext Markup Language (DHTML), PDF, and XML.

Plug-ins are also available to carry out rule-based programming using the information stored in a Protégé frame-based knowledge base. Two worth mentioning examples are JessTab (http://www.ida.liu.se/~her/JessTab/) and Algernon (http://algernon-j.sourceforge.net/doc/algernon-protege.html). JessTab is a plug-in that provides a Jess console window where it is possible to interact with Jess while running Protégé. This plug-in extends Jess with supplementary features that map Protégé knowledge bases to Jess facts. Users can deploy applications that handle Protégé knowledge bases and react when patterns in the knowledge base are found. Algernon is a system implemented in Java that performs forward and backward inference of frame-based knowledge bases. Compared to Jess, Algernon operates directly on Protégé knowledge bases rather than requiring a mapping operation to and from a separate memory space.

The PROMPT plug-in (Noy and Musen, 2003) allows to manage multiple ontologies within Protégé, mainly compare versions of the same ontology, merge ontologies into one, and extract parts of an ontology.

The OWL-S Editor plug-in (http://owlseditor.semwebcentral.org/) is an easy-to-use editor which allows loading, creating, managing, and visualizing OWL-S services. OWL-S (formerly DAML-S)

is emerging as a Web service description language that semantically describes Web Services using OWL ontologies. OWL-S consists of three parts expressed with OWL ontologies: the service profile, the service model, and the service grounding. The profile is used to describe "what a service does", with advertisement and discovery as its objective. The service model describes "how a service works", to enable invocation, enactment, composition, monitoring and recovery. Finally, the grounding maps the constructs of the process model onto detailed specifications of message formats and protocols. The OWL-S Editor plug-in provides an excellent overview of the relations between the different OWL-S ontologies which are shown in an intuitive way in the Graphic User Interface (GUI) and can also be shown as a graph.

4.2 ONTOEDIT

OntoEdit (Sure et al., 2002) was developed by the Knowledge Management Group of the AIFB Institute at the University of Karlsruhe. It is an ontology engineering environment which allows creating, browsing, maintaining and managing ontologies. The environment supports the collaborative development of ontologies (Sure et al. 2002). This is archived through its client/server architecture where ontologies are managed in a central server and various clients can access and modify these ontologies. Currently, the successor of OntoEdit is OntoStudio (Figure 6) which is a commercial product based on IBM's development environment Eclipse. It can be downloaded for three months free evaluation period.

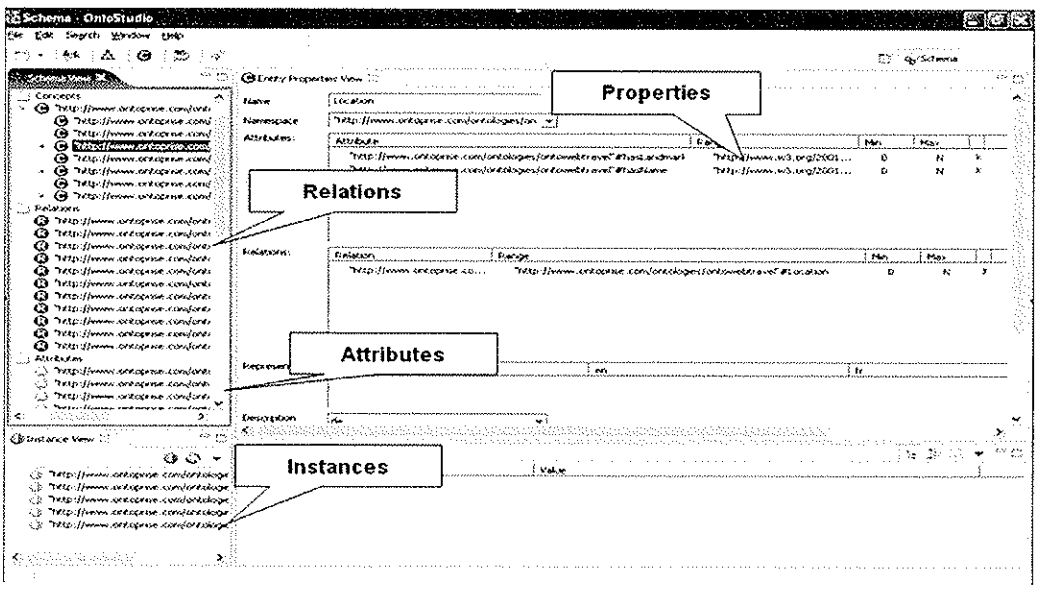

Figure 6: OntoStudio editor

OntoEdit was developed having two major goals in mind. On the one hand, the editor was designed to be as much as possible independent and neutral of a concrete representation language. On the other hand, it was planned to provide a powerful graphical user interface to represent concept hierarchies, relations, domains, ranges, instances and axioms. OntoEdit

supports F-Logic (Fuzzy Logic), RDF Schema and OIL. The tool is multilingual. Each concept or relation name can be specified in several languages. This is particularly useful for the collaborative development of ontologies by teams of researchers spread across several countries and speaking different languages. From the technical perspective, this feature is archived by using unique identifiers so that each ontological statement remains clearly defined. The names selected by users serve merely as an external representation.

OntoEdit is built on top of an internal data representation model. The data model of OntoEdit is OXML 2.0 which is frame based. OXML is defined in XML using XML-Schema. Besides concepts, relations and instances, the model can represent predicates, predicate instances, and axioms. Predicates are n-ary associations and are very similar to predicates defined in first order logic (FOL). Several types of relationships can be established between concepts, such as symmetric, reflexive, transitive,antisymmetric,asymmetric, irreflexive, or intransitive.

The internal representation data model can be exported to DAML+OIL, F-Logic, RDF(S), and OXML. Additionally, ontologies can be exported to relational databases via JDBC. OntoEdit can import external data representation in DAML+OIL, Excel, F-Logic, RDF(S), and OXML. OntoSudio can also import and export OWL files.

OntoEdit provides an API for accessing ontologies in an object-oriented fashion. The default API implementation stores ontologies in main-memory, but an additional API exists for persistent storage.

The inference engine that OntoEdit uses is OntoBroker (Decker, 1999). Using this engine, OntoEdit exploits the strength of F-Logic in that it can represent expressive rules. OntoBroker is the result of several years of research and it is now a commercial product.

Like Protégé, OntoEdit is based on a plug-in architecture. The architecture consists of three layers (Figure 7): GUI, OntoEdit core and Parser.

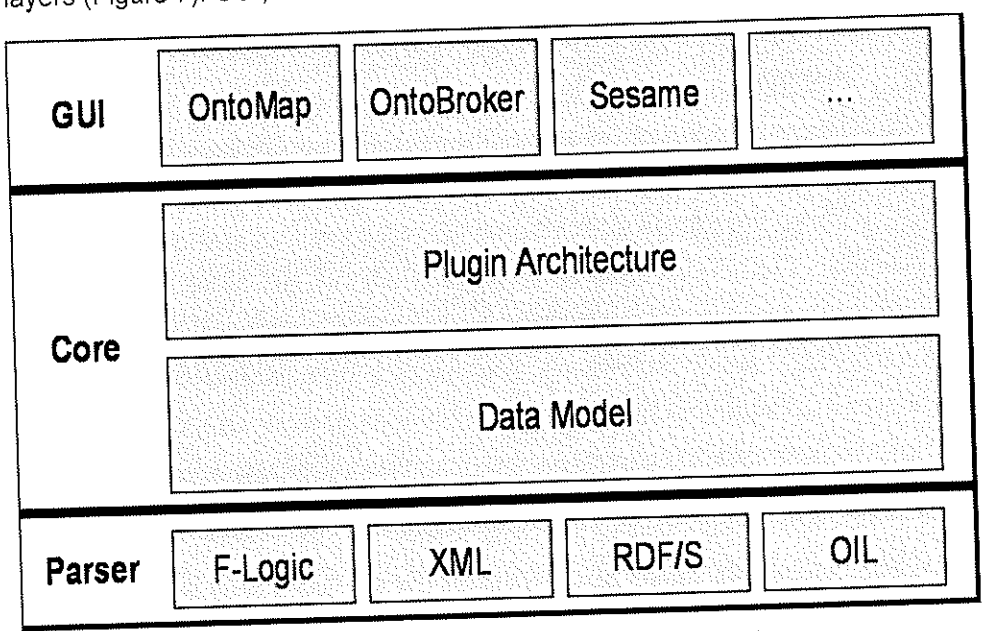

Figure 7: OntoEdit Architecture (Sure et al., 2002)

Using a plug-in architecture enables users to extend OntoEdit with specific functionalities. Since the plug-in interface is open to third parties, anyone with special requirements can deploy a component to satisfy specific needs. Several plug-ins are available. For example, the Sesame plug-in (Broekstra et al., 2002) is a generic application for storing and querying RDF and RDF Schema. Sesame allows persistent storage of RDF data and schema information. It supplies a useful query engine which supports RQL, an OQL-style query language.

4.3 DOE

DOE (Differential Ontology Editor) is a simple ontology editor and was developed by the INA (Institut National de l'Audiovisuel - France). DOE allows users to build ontologies according to the methodology proposed by Bruno Bachimont (Isaac et al., 2002).

DOE has a classical formal specification process. DOE only allows the specification part of the process of structuring ontology. DOE is rather a complement of other editors (DOE, 2006). It is not intended to be a direct competitor with other existing environments (like Protégé, OilEd, OntoEdit or WebODE), instead it was developed to coexist with other editors in a complementary way. This editor offers linguistics-inspired techniques which attach a lexical definition to the concepts and relations used, and justify their hierarchies from a theoretical, human-understandable point of view (DOE, 2006). Therefore, DOE should be used in combination with another editor. For instance, an editor that has advanced formal specification, e.g. Protégé.

DOE is a simple prototype developed in Java that supports the three steps of the Bruno Bachimont methodology (Isaac et al., 2002). The Bruno Bachimont methodology can be described in the following three steps.

In the first step, the user builds taxonomies of concepts and relations. The user has to unambiguously justify the position for each notion in the hierarchy. The user builds a definition, following four principles which come from the Differential Semantics theory (Isaac et al, 2002), i.e., 1) Similarity with Parent, 2) Similarity with Siblings, 3) Difference with Sibling and 4) Difference with Parent. For each notion, a meaning and a description has to be given.

Consequently, the user has to explicit state why a notion is similar but more specific than its parent (i.e., Similarity with Parent and Similarity with Siblings), and why this notion is similar but different from its siblings (i.e., Difference with Sibling and Difference with Parent). Hence, every concept is located in a justified and convinced position. It is possible, for the user, to add synonyms and an encyclopedic definition in a few languages for all notions in the Differential Ontology view. The main goal of this step is to reach a semantic agreement about the meaning of the labels used for naming concepts (Isaac et al., 2002).

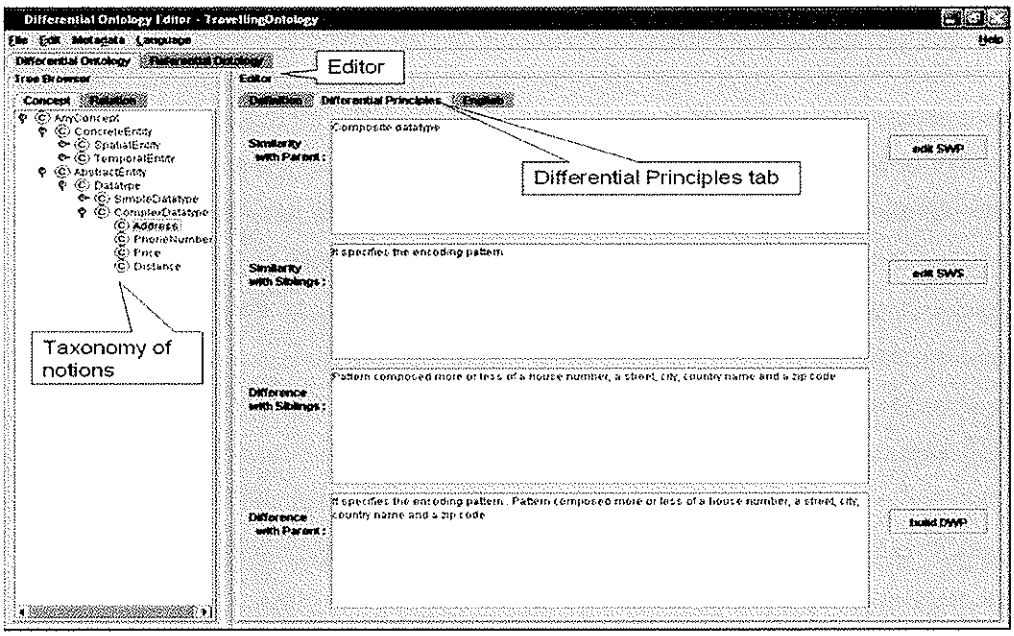

Figure 8: The differential principles bound to the notion addressed in the DOE tool (differential ontology view).

In the second step, the ontological tree obtained in the first step allows to disambiguate the notions. Consequently, the meaning is clarified for a domain-specific application. The notions become concepts behaving as formal primitives. In the referential ontology each concept refers to a set of objects in the domain (its extension) (Isaac et al., 2002). The taxonomies are considered from an extensional semantic point of view. The user can expand them with new entities (defined) or add constraints onto the domains of the relations. In this step, the user ought to do consistency checking in order to look for propagation of the arity all along the hierarchy- if specified - and inheritance of domains. As we said before, DOE main goal is to guide the user during the first steps of the process of building an ontology. It is not possible to write axioms in DOE since there is not an axiom editor.

Finally, on the third step, the ontology can be translated into a knowledge representation language. The referential concepts are compared with the possible computational operations available in the application Knowledge Based Systems (KBS). As a result, it is possible to use it in an appropriate ontology-based system or to import it into another ontology-building tool to specify it further. An export mechanism was implemented in order to translate the taxonomies into convenient exchange languages (for example, OWL). This mechanism can also be used to complete the third step

DOE allows building a differential ontology and a referential ontology, corresponding to the first two steps of the methodology proposed by Bruno Bachimont. The Differential Ontology view is divided in two frames. On the left side there is a tree browser and on the right side there is an editor. The tree browser shows hierarchically the concepts and the relations (there is a tab for concepts and a tab for relations too, like in the Referential Ontology view). The editor has three tabs: a definition tab, a different principles tab and an English tab (or other language that the user wishes to translate the ontology to). If the definition tab is selected, it will show the

"properties" of the items illustrated in the tree browser. The different principles tab is used to justify the taxonomy of notions that was build. In other words, the sense of a node is specified by the gathering of all similarities and differences attached to the notions found on the way from the root notion (the more generic) to the node that is selected on the tree browser (on the left) (Isaac et al., 2002). In the Referential Ontology view, the tree browser has the same features as in the Differential Ontology view (there are two tabs concept tab and relation tab).

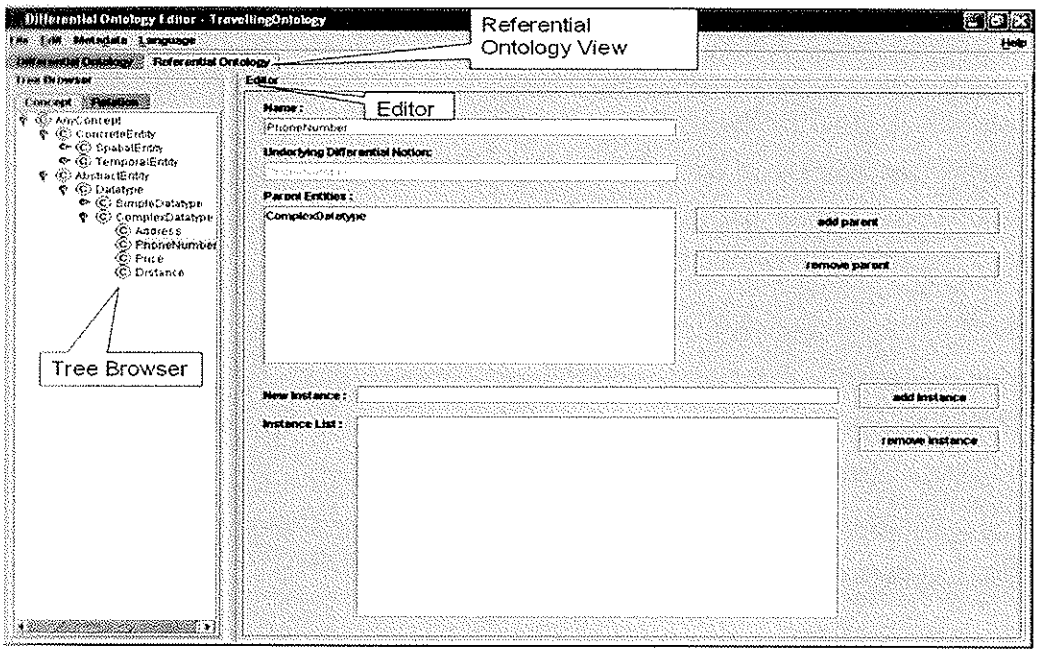

Figure 9: The Referential Ontology view with a concept tree of the Traveling Ontology.

While DOE does not have a graph view of the ontology, it includes a tree view that can show separately the concepts and the relations between the concepts. It is possible to view an ontology in two different ways: a Differential Ontology view and Referential Ontology view.

DOE can import RDFS and Ontology Web Language (OWL) formats. It is possible to export to CGXML, DAML+OIL, OIL (Ontology Inference Language) Plain Text, OIL XML, RDFS, RDF/XML, and OWL. The RDFS import/export feature is very helpful when "exchanging" files between editors, for instance OilEd, Protégé and OntoEdit and vice-versa.

DOE uses XSLT to promote interoperability (Isaac et al., 2003). XSLT is a language for transforming XML documents into other XML documents. All the exchange facilities are done with XSLT transformations.

DOE project does not have a mailing list available to the users. The installation instructions are very simple. There are no plug-ins for this editor and no user manual too.

In DOE Web page (http://opales.ina.fr/public/) there is a self-contained installer of the complete product for Windows. The user has to fill out a form in order to access the download page. The installation is fast. After downloading the product, the user has to run the Setup-DOE-v1.51.exe file and follow the instructions. To run DOE in another platform, it is required to have a Java 2 (TM) Platform, Standard Edition Version 1.3 or later (recommended v1.4.1).

4.4 IsaViz

IsaViz is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz (http://www.w3.org/2001/11/IsaViz/Overview.html) was developed by Emmanuel Pietriga. The first version was developed in collaboration with Xerox Research Centre Europe which also contributed with XVTM, the ancestor of ZVTM (Zoomable Visual Transformation Machine) upon which IsaViz is built. As of October 2004, further developments are handled by INRIA Futurs project In Situ. IsaViz also includes software developed by HP Labs (Jena 2 Semantic Web Toolkit), the Apache Software Foundation (Xerces Java 2), and makes use of the GraphViz library developed by AT&T Research (http://www.w3.org/2001/11/IsaViz/Overview.html).

IsaViz does not follow or include any methodology for building an ontology.

IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontologies to other editors, for instance, Protégé or OilEd.

The IsaViz environment is composed of four main windows: the IsaViz RDF Editor window, the Graph window, the Definition window and the Attribute window.

- The IsaViz RDF Editor window contains the main menus and a palette of tools.

- The Graph window is a ZVTM view (http://www.w3.org/2001/11/IsaViz/Overview.html) in which is displayed the RDF model represented as a 2D graph. ZVTM views display an area of an endless space as seen through a camera. This camera can be moved in the virtual space and its altitude can be changed, resulting in a 2.5D Graphical User Interface with smooth zooming capabilities. IsaViz has a user friendly interface. IsaViz has three different ways of viewing a graph. This can be a distinguishing feature when evaluating this tool with others tools. There are 3 different ways of viewing a graph: Graph View; Radar View and Property Browser. IsaViz is also recognize by its Zoomable User interface (ZUI). This interface allows rapid navigation of the graphs used to represent the RDF models.

- The Definitions window is composed of five tabs: a) namespaces; b) property types; c) property browser, d) stylesheets and e) quick input.

a) The Tab Namespaces is a table containing namespace definitions (with their optional prefix bindings). b) The Tab Property Types is a table containing property type definitions.

c) The Tab Property Browser is a text browser in which is exhibit all the properties associated with the currently selected resource. d) The Tab Stylesheets (since version 2.0) is a table containing all Graph Stylesheets (GSS) that should be applied to the model. e) The Tab Quick Input (since version 2.0) is a text area used for pasting statements expressed in RDF/XML, N-Triples or Notation3.

- The Attribute window shows the attributes of a selected item of the graph.

All the attributes shown in this window can be edited.

IsaViz can render RDF graphs using GSS, a stylesheet language derived from Cascade Style Sheet (CSS) and Scalable Vector Graphics (SVG) for styling RDF models represented as node-link diagrams. Resources and literals are the nodes of the graph (ellipses and rectangles respectively), with properties represented as the edges linking these nodes.

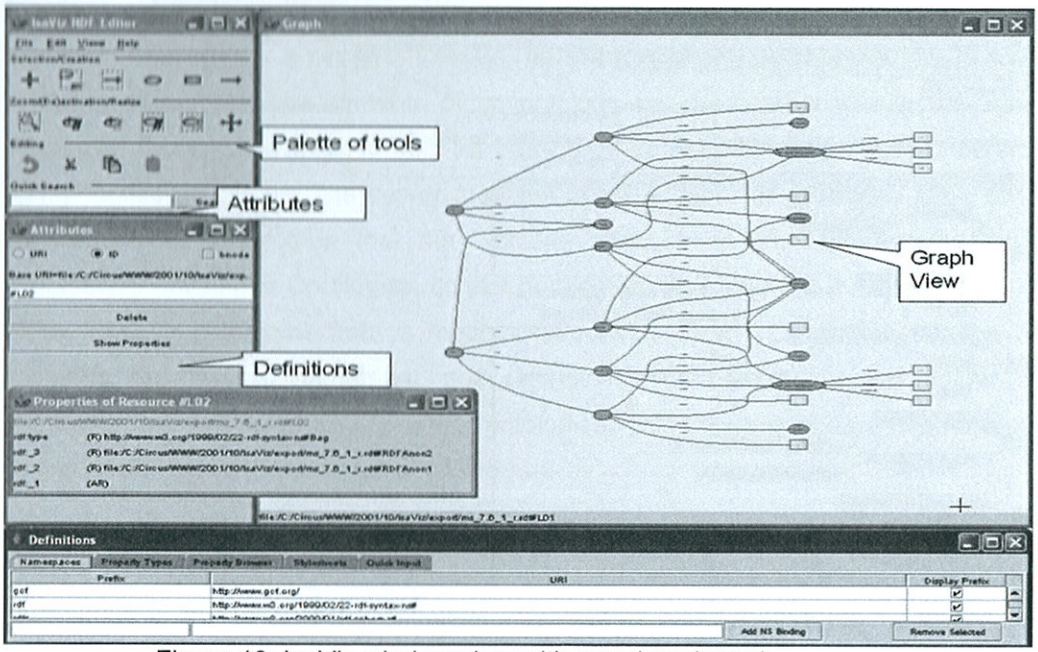

Figure 10: IsaViz window view with a node selected.

This editor has a user manual and a Web page with installation instructions. It also has a mailing list and a list of the most common problems. For that reason, it is really simple to start using this tool. The user can easily solve the problems that can emerge during installation or usage, by using the mailing list of IsaViz.

The current stable version is 2.1 (October 2004) and it is being developed an alpha version: 3.0 (December 2005).

IsaViz is implemented in Java and requires a JVM 1.3.x or later to run (1.4.0 or later is strongly recommended), Jena 1.2 (or later), GraphViz 1.8.x, Xerces 1.4.x, as well as GraphViz (http://www.w3.org/2001/11/IsaViz/Overview.html) for some features. Installation instructions are available at the editor's web page (http://www.w3.org/2001/11/IsaViz/Overview.html).

4.5 ONTOLINGUA

The Ontolingua server was the first ontology tool created by the Knowledge Systems Laboratory at Stanford University. Ontolingua was built to ease the development of ontologies with a form-based Web interface. Initially the main application inside the Ontolingua Server was an ontology editor. The Ontology Editor is a tool that supports distributed, collaborative editing, browsing

and creation of Ontolingua ontologies. Other systems were included in the environment, such as Webster, Open Knowledge Base Connectivity (OKBC) Server and Ontology merge tool. The Ontolingua Server ontology editor is also known as the Ontolingua Server frame-editor or even as the Ontolingua Server.

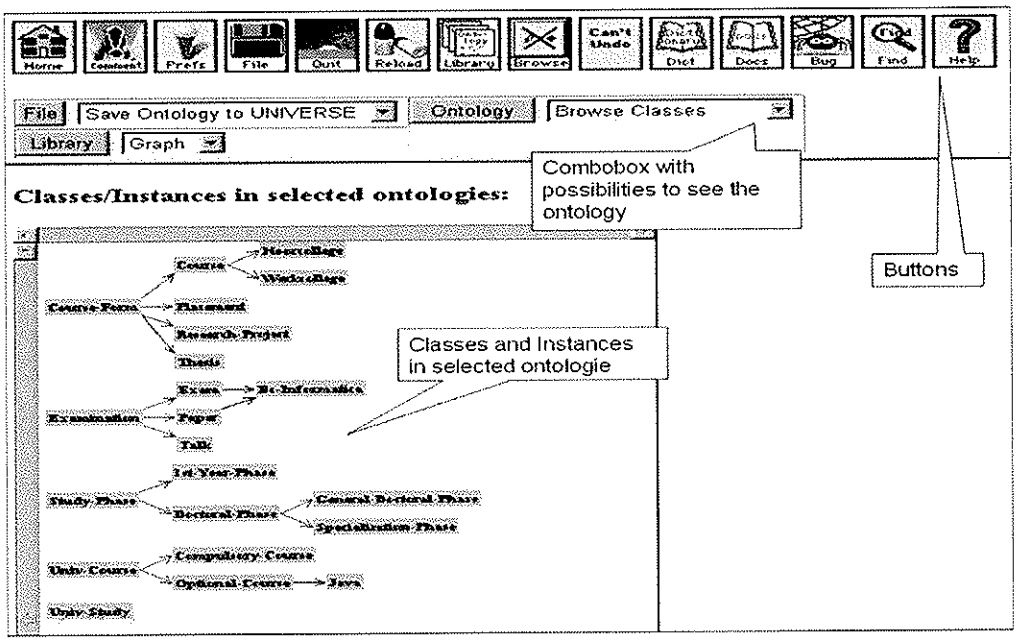

Figure 11: Ontolingua Editor view.

It is possible to export or import to the following formats: DAML+OIL, Knowledge Interchange Format (KIF), Open Knowledge Base Connectivity (OKBC), LOOM, Prolog, Ontolingua, C Language Integrated Production System (CLIPS). It is also possible to import Classic Ocelot and Protégé knowledge bases, but it is not possible to export to these formats.

If an ontology is available in the Ontolingua Server, there are two ways of using it from an external application: connecting to the OKBC Server from a remote client or using the ontology editor to translate the ontology into an implementation language.

The Ontolingua Server is organized as a set of ontology related Web applications, which are built on top of the Ontolingua Knowledge Representation (KR) System. Ontolingua uses an OKBC model with full KIF axioms. The base language is Ontolingua. This language consists of the combination of frames and first order logic, and allows representing classes. Classes are organized in taxonomies, relations, functions, formal axioms, and instances. The ontology editor is divided in two frames. The frame at the top shows the menu with the editing options available to users. The other frame shows the form to create the new class, where the user must specify the class name, and optionally the natural language documentation and superclasses of the concept. After the form is filled, the user must send its description to the server to store the definition in Ontolingua.

A session can be accessed at http://www.ksl.stanford.edu/software/ontolingua/ by any user in the group that owns the session. The other users connected to the session will be notified of the side-effects that are performed. To establish a new group, the user should use the comment button to email the group name that the user wishes to use along with an initial list of members.

The Ontolingua Server ontology editor permits to manage users and user groups who can share their ontologies. Users can work in different sessions and with different groups for each session, and then decide which ontologies can be shared with the members of each group (Farquhar et al., 1995).These multi-user sessions encourage collaborative ontology development. Nevertheless, since no locking mechanisms for ontology terms or version management functions exist in the editor, collaboration may sometimes be somewhat limited. The user can export an ontology (or a set of ontologies) as static HTML files to a local site. It is possible to compare ontologies. The contents of an ontology can be compared with any other loaded ontology. Users often want to split an ontology into a set of more easily understood, more tightly focused modules. The server provides direct support for splitting a large ontology into several smaller ontologies that may include each other. The Ontolingua Server allows assembling Ontolingua Ontologies, so that ontologies can be built in a modular way. Users can reuse existing ontologies from a modular structured library. Ontologies can be reused by inclusion, polymorphic refinement, and restriction. The primary mechanism for supporting ontology reuse is through the library of ontologies. This library acts as a central repository for reusable ontologies (Farquhar et al., 1995).

The main difference from other editors is the fact that users must have some notions of KIF and of the Ontolingua language in order to fill in the forms. Hence, if the user does not have any knowledge of KIF and of the Ontolingua language, the user will not be able to create an ontology since there is no help menu to build the KIF expression of an Ontolingua axiom.

The interface is not very user friendly since the buttons are large and primitive and the drawings inside the buttons can sometimes be ambiguous.

There is no need for an installation since this editor has Web access. The user has to be registered in order to use the system. After being registered the following services are available: Chimaera, the Ontology Editor and Webster. Chimaera helps users to reorganize taxonomies and resolve name conflicts in a knowledge-base. It is especially useful when merging KBs, but also useful as an ontology browser and ontological sketchpad. The Ontology Editor allows users to browse, create and edit ontologies using a Web browser. Webster is an online version of the Webster Dictionary, i.e., a searchable hypertext dictionary.

4.6 ALTOVA SEMANTICWORKS™ 2006

Altova SemanticWorks 2006 is a commercial visual Semantic Web editor which provides powerful, easy-to-use functionality for a visual creation and editing of RDF, RDF Schema (RDFS), OWL Lite, OWL DL, and OWL Full documents. This editor has an intuitive visual interface and drag-and-drop functionalities.

It allows users to visually design Semantic Web instance documents, vocabularies, and ontologies.

There is no evident orientation or methodology associated to this tool.

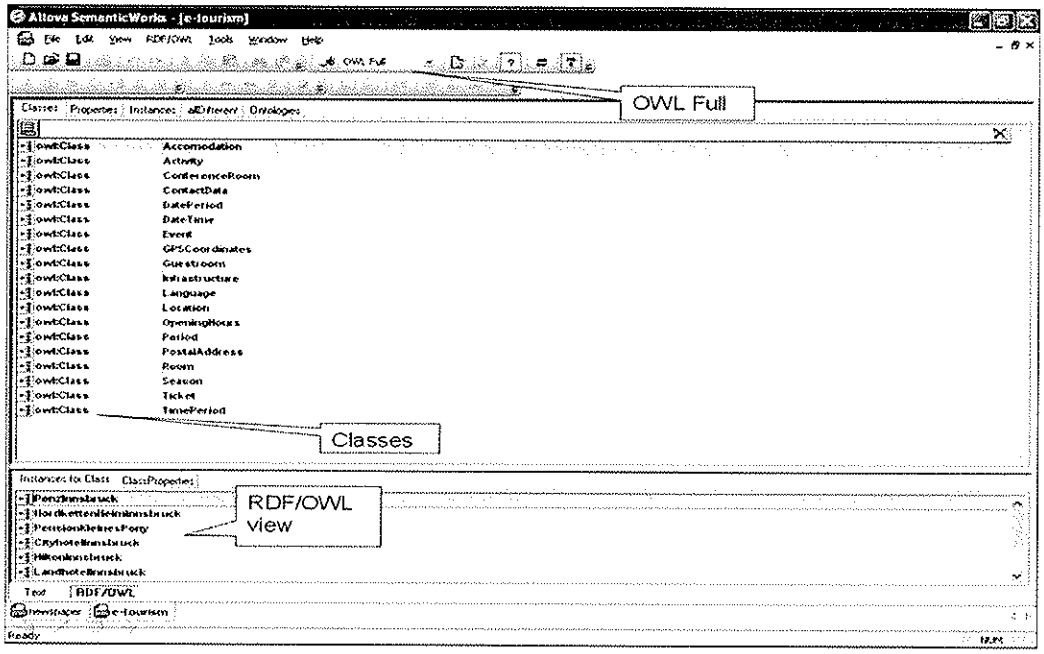

Figure 12: Semantic Works 2006 view bound to the e-tourism ontology.

This editor has the ability to manage the following files: N-triples, XML, OWL, RDF and RDFS. This tool provides powerful features for working with RDF in RDFS vocabularies and OWL ontologies.

Users are capable of printing the graphical RDF and OWL representations to create documentation for Semantic Web implementations. The user can switch from the graphical RDF/OWL view to the text view to see how documents are being built in RDF/XML, OWL or N-triples format. The RDF/XML, OWL or N-triples code is automatically generated based on the user's design. Therefore, users can learn and try out with the concepts of the Semantic Web without having to write complicated code.

The graphical display is highly configurable. The user has the possibility to adjust the width of the items in the graph, exhibit them with a vertical or horizontal orientation, adjust the distances between parent and child nodes, and change the font styles and colors used.

An intelligent right menu and context sensitive entry helpers give support to change or add details to the RDF resource according to the user choices. The entry helpers and menus only offer the choices permitted based on the RDF specification, so that users can be certain to create valid documents. Any conflicts are listed in the error window. The errors are written as links so that the user can find and repair them promptly and easily.

A full version of this editor can be installed using a self-contained installer. It is easy and fast to install. The user has to request a free evaluation key, by giving the following information: Name, Company name and e-mail. Users can test this editor for a period of 30 days after receiving the key password by e-mail.

4.7 OILED

OilEd is a graphical ontology editor developed by the University of Manchester for Description Logic Ontologies (Gómes-Pérez et al., 2004). The main purpose of OilEd is to provide a tool for ontologies or schemas editing, as opposed to knowledge acquisition, or the construction of large knowledge base of instances (Laera and Tamma, 2005). OilEd's interface was strongly influenced by Stanford's Protégé toolkit. The initial intention behind OilEd was to provide a simple editor that demonstrated the use of, and stimulated interest in, the Ontology Inference Layer (OIL) language.

The current version of OilEd does not provide a full ontology development environment. OilEd is not capable of supporting the growth of large-scale ontologies, the migration and integration of ontologies, versioning, argumentation and many other activities that are involved in ontology construction. Rather, it is the "Notepad" of the ontology editors, offering enough functionality to allow users to build ontologies and to demonstrate how the Fast Classification of Terminologies (FaCT) reasoner checks ontologies for consistency (Laera and Tamma, 2005). OilEd does not follow any methodology.

OilEd's internal data format is DAML+OIL (DAML- DARPA Agent Markup Language; OIL- Ontology Inference Layer). It is possible to import from DAML+OIL and export ontologies as RDFS, DAML OWL, RDF/XML and others formats.

Figure 13 shows OilEd Editor. The tabs on the main window give access to classes, properties, individuals and axioms. In each tab, a list on the left sides illustrates all the items. On the right panel there is the editor. This editor is used to see or to change the information concerning the entry selected in the item list on the left. The editor was reimplementated reasoner interface so that it was possible to use the DIG (DL Implementation Group) protocol. OilEd includes a DAML+OIL checker. Given the location of an ontology the application checks the syntax of DAML+OIL ontologies and returns a report of the model.

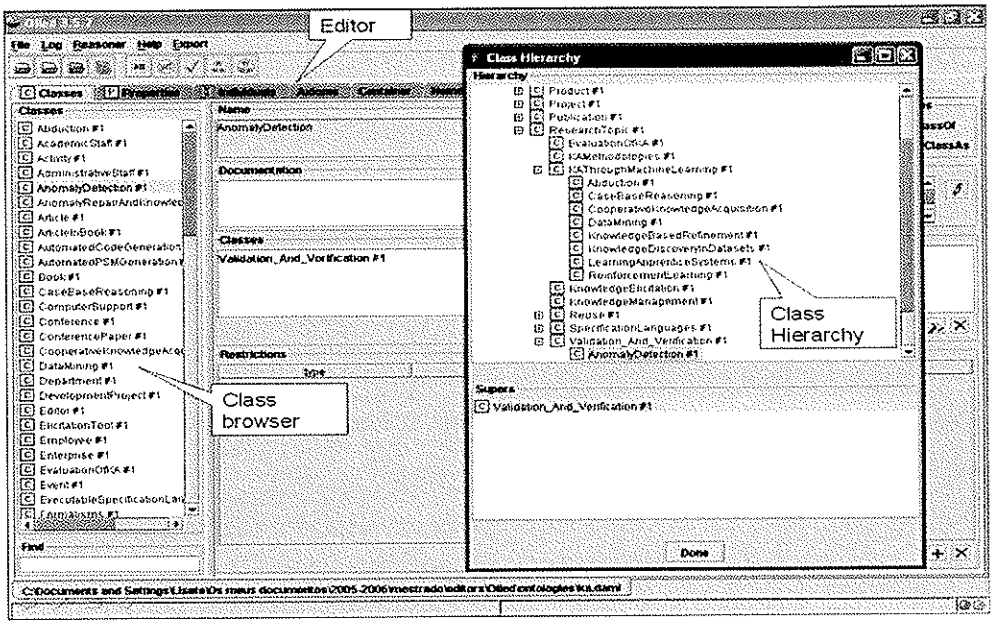

Figure 13: OilEd Ontology editor view bounded with the class AnomalyDetection.

OilEd is provided free of charge, but the user has to provide some information before downloading this editor. Registration is a simple process requiring a valid email address. The user has to be registered, in order to receive a password to revisit the main Web page to download patches and extensions when necessary.

In order to use OWL ontologies, the user must download the sesame-oil.jar library, place it in the /lib directory of the user's installation and remove the existing oil.jar. Sesame is a repository for OWL ontologies. This will add new options to the menus to open and save to Sesame repositories. OilEd expects that these Sesame repositories contain OWL files (rather than, say, DAML+OIL). This only happens since the ontologies are stored in Sesame repositories

The OilEd user manual is included in the distribution.

Angus Roberts has produced a brief tutorial introduction to OilEd

OilEd is available as an open-source project under the GPL license. The latest version of OilEd (version 3.5) can be download at http://OilEd.man.ac.uk/.

The following sample ontologies produced using OilEd are available:

- Knowledge Acquisition: The (KA)2 demo ontology from the Ontobroker project.
- Wines: Deborah McGuinness' DAML+OIL wines ontology.
- Diving: An ontology of terms from scuba diving.
- Mad Cows: An ontology demonstrating the use of the reasoner to spot inconsistent concepts.

There are the following possible packages for installation:

OilEd comes packaged with the FaCT reasoner, although alternative reasoners can be used. OilEd 3.5.7 (Windows)+reasoner. This archive contains a FaCT reasoner for Windows.

OilEd 3.5.7 (Linux)+reasoner. This updated version supports export information knowledge base models to OWL-RDF. This archive also contains the FaCT reasoner.

OilEd 3.5.7 (no reasoner). This archive does not contain a reasoner. Use this distribution only if you are not interested in using a reasoner or wish to use OilEd with an alternative DIG reasoner.

4.8 WEBODE

WebODE has been developed by the Ontological Engineering Group (OEG) from the Artificial Intelligence Department of the Computer Science Faculty (FI) from the Technical University of Madrid (UPM). WebODE is an ontological engineering workbench that provides various ontology related services (Laera and Tamma, 2005). It provides support to most of the activities involved in the ontology development process and ontology practice such as ontology edition, navigation, documentation, merge, reasoning, etc.

The WebODE ontology editor is a Web application. It was build on top of the ontology access service (ODE API).

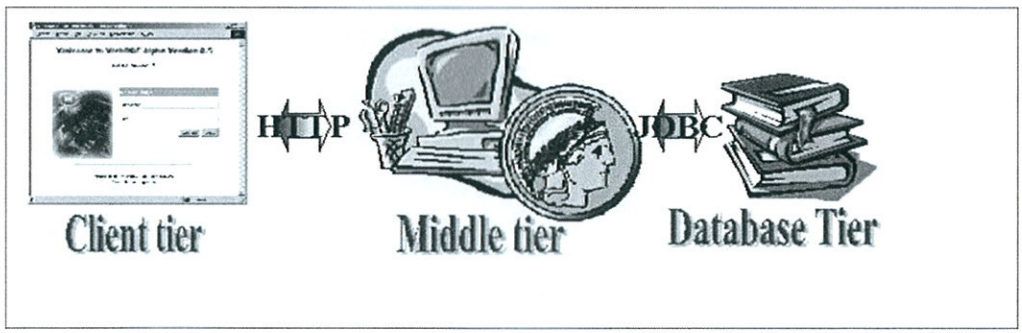

Figure 14: Architecture of the WebODE platform (Laera and Tamma, 2005)

The WebODE platform has been built using a three-tier model (illustrated in Figure 7) which include: 1) Presentation Tier, 2) Middle Tier and 3) Database Tier.

The first tier (or presentation tier) provides the user interface. This interface is provided by means of a Web browser and uses standard Web technologies. The presentation tier was implemented using HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) and XML (Extended Mark-up Language), because these technologies permit a simple interoperability between applications. Technologies like JavaScript and Java were used in order to ease the server of the weight of user validations. JavaScript provides easy and quick form validation. Java allows more complex presentation and logic schemas like graphical design or formula validation.

The second tier, the middle tier, provides the business logic. This tier is the result of the combination of two other sub-tiers: presentation sub-tier and logic sub-tier. The presentation

sub-tier is in charge of generating the content to be presented in the user's browser. It is also aimed at handling user requests from the client (such as form and query handling) and forwards them to the ODE service as appropriate. Technologies such as servlets or JSPs (Java Server Pages) were used for this purpose. The logic sub-tier provides direct access to ontologies. This direct access is made by means of a well-defined API supplied through an application server (Minerva Application Server). This server provides access to services through RMI (Remote Method Invocation) thus making application development and integration very easy.

The third tier, the database tier, contains the data. Ontologies in WebODE can be stored in any relational database. The database is accessed by means of the JDBC (Java Database Connectivity) standard.

Ontologies in WebODE are conceptualized with a very expressive knowledge model. This knowledge model is based on the reference set of intermediate representations of the Methontology methodology (Laera and Tamma, 2005). The following ontology components are included in the WebODE's knowledge model: concepts and their local attributes (both instance and class attributes), whose type can be any XML Schema type; concept groups, which represent sets of disjoint concepts; concept taxonomies, and disjoint and exhaustive class partitions; ad hoc binary relations between concepts, which may be characterized by relation properties (symmetry, transitiveness, etc); constants; formal axioms, expressed in first order logic; rules; and instances of concepts and relations.

WebODE gives support to the ontology building methodology Methontology.

WebODE contains an ontology editor, an ontology-based knowledge management system (ODEKM), an automatic Semantic Web portal generator (ODESeW), a Web resource annotation tool (ODEAnnotate), and a Semantic Web service tool (ODESWS).

The ontology editor has three user interfaces: an HTML form-based editor for editing all ontology terms except axioms and rules, a graphical user interface (OntoDesigner) and a WAB (WebODE Axiom Builder) for editing formal axioms and rules. This editor has the following ontology building services: documentation service, OKBC-based Prolog Inference engine, and ODEClean.

WebODE has automatic exportation and importation services from and into XML. WebODE has translation services to other languages and systems or to and from diverse ontology specification languages. Therefore, WebODE presents vast potential capabilities for interoperability.

Ontology export services allow generating WebODE ontologies in XML and in several other ontology languages, such as: RDF(S), OIL, DAML+OIL, OWL, and F-Logic. Ontologies can be transformed and used with the Protégé ontology editor or use the interoperability capabilities provided by this tool.

There are several ways of using WebODE ontologies inside ontology-based applications: by its Java API via a local service or application running on the same computer where the ontology

server is installed, by generating WebODE ontologies in XML and in several other ontology languages, by transforming ontologies into Protégé-2000 or into Java.

The Java API avoids accessing directly the relational database. WebODE ontologies can be accessed not only from inside the local server but also remotely with RMI and Web services (Gómes-Pérez et al., 2001).

It is possible to generate WebODE ontologies into: RDF(S), OIL, DAML+OIL and OWL. The WebODE ontologies can be used inside Protégé-2000 ontology editor.

During this process of transforming ontologies into Java, concepts are transformed into Java beans, attributes into class variables, ad hoc relations into associations between classes. This Java code can then be used to create other Java applications and uploaded in rule systems like Jess.

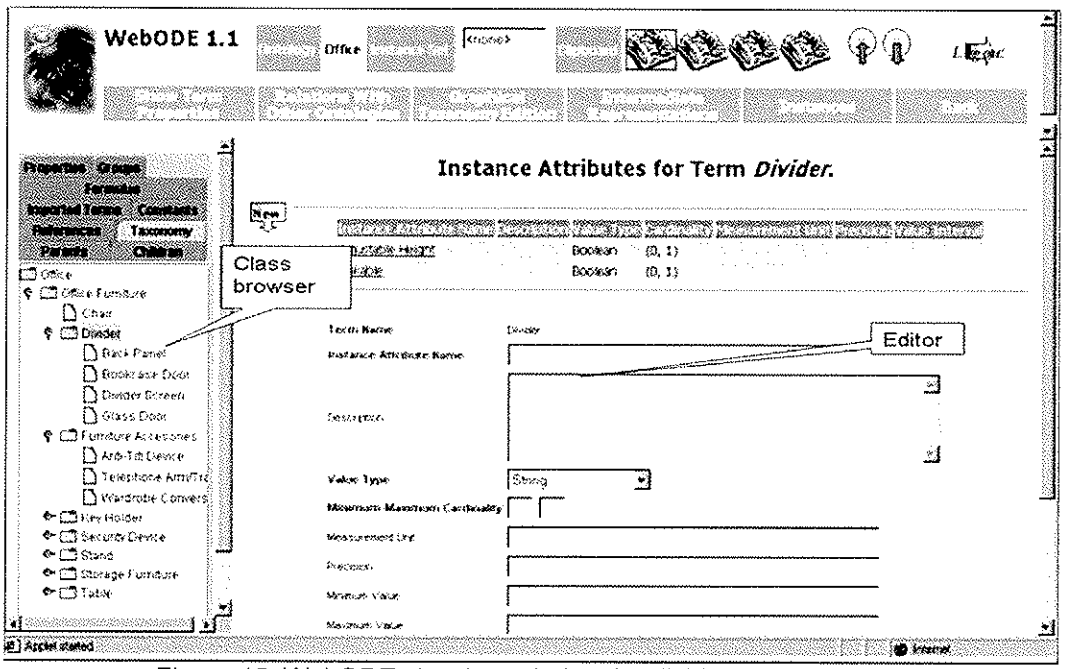

Figure 15: WebODE view bounded to the divider concept.

WebODE has support for multiple-users. User groups can be created to collaborate in the edition of ontologies. Several users can edit the same ontology without errors by means of synchronization mechanisms.

One important advantage of using this application server is that we can decide which users or user groups may access each of the services of the workbench.

This editor provides Web access and therefore, there is no need for any installation activity (http://kw.dia.fi.upm.es/wpbs/#download) .

4.9 POWL

pOWL (Auer, 2005) is a PHP-based open source ontology management tool. pOWL is capable of supporting parsing, storing, querying, manipulation, versioning, serving and serialization of RDFS and OWL knowledge bases in a collaborative Web enabled environment.

pOWL does not follow any specific methodology for developing ontologies.

pOWL supports heterogeneous data and its formal description. pOWL tries to follow the W3C Semantic Web Standards. pOWL can import and export model data in different serialization formats such as RDF/XML, and N-Triple.

pOWL (AUER, 2005) is designed to work with ontologies of arbitrary size. This action is limited by the disk space. Therefore, only some parts of the ontology are loaded into main memory. The parts loaded are the ones required to display the information requested by the user on the screen.

pOWL offers an RDQL (RDF Data Query language) query builder. pOWL has a tab that correspond to the RDQL query builder. This RDQL is an implementation of an SQL-like query language for RDF. It is possible to query the knowledge base as well as a full-text search for literals and resources. pOWL has an object oriented API. This means that all functionalities are accessed with a clean application programming interface. Models are stored in database tables. It is possible to edit and view models from different points of view.

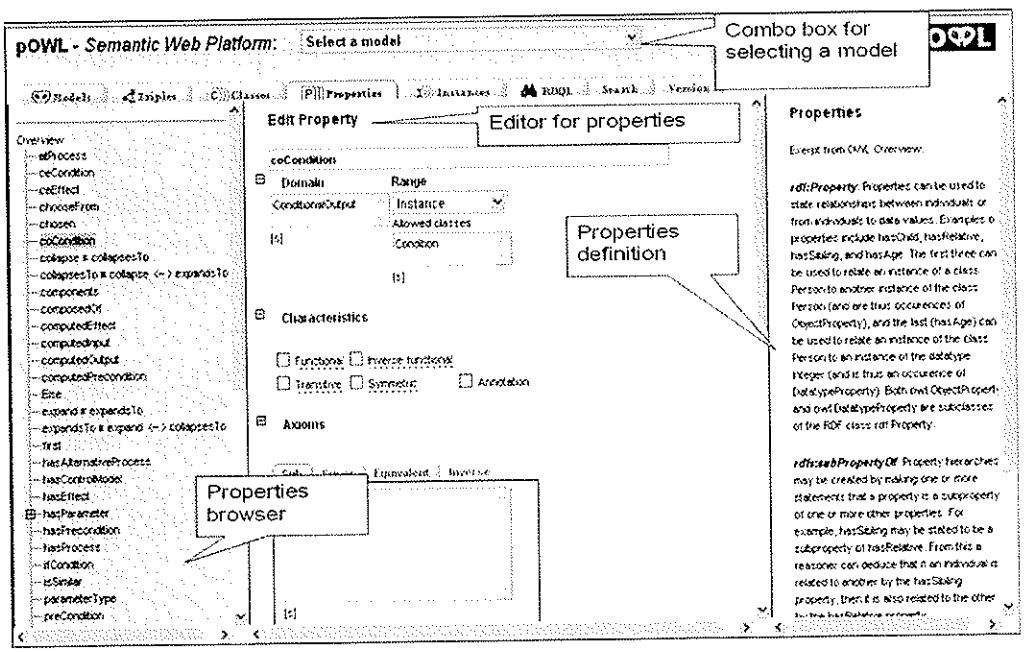

Figure 16: View of the pOWL ontology editor showing the property elements of the coCondition.

pOWL's architecture is formed by four tiers: a) pOWL store, b) RDFAPI, RDFSAPI, OWLAPI, c) pOWLAPI and d) the user interface.

a) pOWL store stores its data in a relational database. This database is used to store all the information concerning ontologies and their history.

b) In order to handle RDF, RDFS and OWL there are three different APIs: RDFAPI, RDFSAPI and OWLAPI. All classes of RDFSAPI are extended with methods for handling OWL predefined properties, Description Logic axioms and restrictions, as well as basic subsumption inference as Lassila and Swick (1999) stated.

c) The pOWLAPI includes classes and functions to build Web applications. These web applications are built on top of the APIs already described (Auer, 2005).

d) The API that permits accessing, browsing, viewing and editing data models in the pOWL store is the user interface. This interface is based on PHP pages. It is possible to have three different viewpoints of an OWL knowledge base: triple view, database view and a description logic axiom view.

The RDF statements can be viewed on the "triple" view. These statements are written following this principle: subject/predicate/object. The knowledge base can be checked as if it was an object-relational database. OWL classes are represented as tables. The columns of the table represent the properties and the rows represent the instances. On the Description Logic (DL) axiom view, the DL axioms are detected and represented using DL.

pOWL does not have a graph view but a tree view is shown on the left side panel (see Figure 16).

pOWL has multi language support and has versioning control.

pOWL has the following available documentation: user documentation, installation and administration documentation, developer documentation and application and usage scenarios. pOWL Website has six public areas: an area for bugs, an area for support requests, an area for feature requests, a public forum, a mailing list and a CVS (Concurrent Version System) repository. All edits of a knowledge base may be logged and rolled back. This will only depend on time, user and edit action.

pOWL requires a PHP enabled Web server and a database backend for storing data models. pOWL is available under GNU Public license. The complete source code is also available at http://sourceforge.net/projects/pOWL/.

4.10 SWOOP

SWOOP is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. Different ontologies can be compared against their Description Logic-based definitions, associated

properties and instances. SWOOP's interface has hyperlinked capabilities so that navigation can be simple and easy. SWOOP does not follow a methodology for ontology construction.

Users can reuse external ontological data (Kalyanpur et al., 2005). This is possible either by purely linking to the external entity, or importing the entire external ontology. It is not possible to do partial imports of OWL. There are several ways to achieve this, such as a brute-force syntactic scheme to copy/paste relevant parts (axioms) of the external ontology, or a more elegant solution that involves partitioning the external ontology while preserving its semantics and then reusing (importing) only the specific partition as desired.

It is possible to search concepts across multiple ontologies. SWOOP makes use of an ontology search algorithm, that combines keywords with DL-based in order to find related concepts. This search is made along all the ontologies stored in the SWOOP knowledge base.

With SWOOP it is possible to have collaborative annotation using the Annotea plug-in. This plug-in presents a useful and efficient Web ontology development. Users may also download annotated changes for a given ontology. The plug-in is used by the users to write and share annotations on any ontological entity. Different SWOOP users can subscribe to the server. Users can maintain different version of the same ontology since mechanisms exist to maintain versioning information using a public server.

SWOOP takes the standard Web browser as the User Interface paradigm. This Web ontology browser has a layout that is well known by most of the users. There is a navigation side bar on the left (Figure 17). The sidebar contains a multiple ontology list and a class/property hierarchy of the ontology. The center pane works like an editor. There is a range of ontology/entity renders for presenting the core content.

This editor provides support for ontology partitioning. OWL ontologies can be automatic portioned into distinct modules each describing a separate domain. There is also support for ontology debugging/repair. SWOOP has the ability to identify the precise axioms that causes errors in an ontology and there are also natural language explanation of the error. An automatic generation of repair plans to resolve all errors are provided. To better understand the class hierarchy, a "CropCircles" visualization format was implemented.

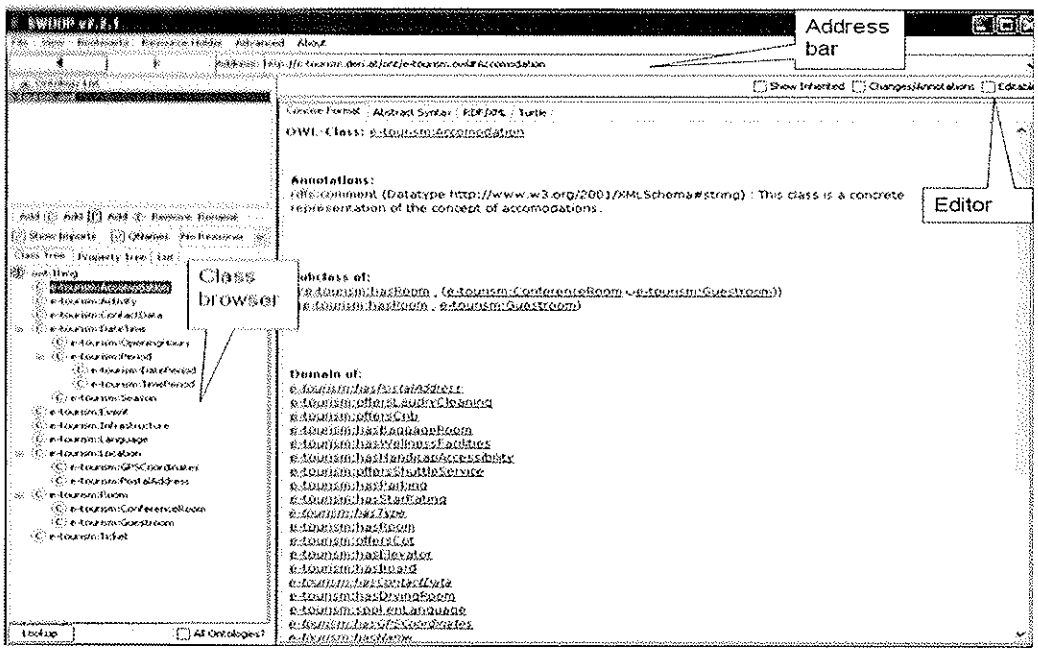

Figure 17: SWOOP view bounded to the e-tourism ontology

SWOOP is based on the conventional Model-View Controller (MVC) paradigm (Figure 17). The SWOOP Model component stores all ontology-centric information pertaining to the SWOOP workspace and defines key parameters used by the SWOOP UI objects. A SWOOP ModelListener is used to reflect changes in the UI based on changes in the SWOOP Model (Kalyanpur et al., 2005). Control is managed through a plug-in based system. This system loads new Renders and Reasoners dynamically. Therefore, it is possible to guarantee modularity of the code, and encourage external developers to contribute to the SWOOP project easily.

SWOOP uses the Wonder Web OWL API. This API provides programmatic access to data representing OWL ontologies. This API is used as the underlying OWL representation model.

SWOOP has Ontology Renderers that display statistical information about the ontology, the annotations, the DL expressivity and the OWL species level.

SWOOP has default plug-ins for different presentation syntax for rendering ontologies. For instance, there are the following presentations syntax RDF/XML, OWL and N3.

SWOOP is easy to install and has two public mailing lists available: General SWOOP (for users) and Technical SWOOP (for developers). A SWOOP application can be downloaded from http://www.mindswap.org/. After downloading the package, the user has only to run the "runme" batch file.

SWOOP is developed as a separate Java application that attempts to provide the look and feel of a browser-based application. Its architecture was designed to optimize OWL browsing and to be extensible via a plug-in architecture.

4.11 CONCLUSION

Software tools are available to achieve most of the activities of ontology development. Projects often involve solutions using numerous ontologies from external sources. Sometimes there is also the need to use existing and newly developed in-house ontologies. By this reason it is important that the editing tools for ontology construction promote interoperability. As we have stated, Protégé is used for domain modeling and for building knowledge-base systems. Protégé provides an intuitive editor for ontologies and has extensions for ontology visualization, project management, software engineering and other modeling tasks. It also provides interfaces for Description Logic Reasoners such as Racer. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. It is a tool installed locally in a computer and does not allow collaborative editing of ontologies by groups of users. There are several plug-ins available for this tool.OntoEdit offers an ontology engineering environment which allows creating, browsing, maintaining and managing ontologies. This editor supports collaborative development of ontologies. The successor of OntoEdit is OntoStudio. OntoEdit supports F-Logic, RDF Schema and OIL. The tool is multilingual. The internal representation data model can be exported to DAML+OIL, F-Logic, RDF(S), and OXML. The inference engine that OntoEdit uses is OntoBroker. OntoEdit is based on a plug-in architecture as Protégé.

DOE allows users to build ontologies according to the methodology proposed by Bruno Bachimont. This editor only permits the specification part of the process of structuring ontology. It should be used in combination with another editor. A differential ontology and a referential ontology can be built. It is possible to import RDFS and OWL formats and it is possible to export to CGXML, DAML+OIL, OIL Plain Text, OIL XML, RDFS, RDF/XML, and OWL. DOE uses XSLT to promote interoperability.IsaViz is a visual environment for browsing and authoring RDF models as graphs. IsaViz has a user friendly interface and has three different ways of viewing a graph. IsaViz is also recognized by its ZUI. This interface allows rapid navigation of the graphs used to represent the RDF models. IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, PNG and SVG.

Ontolingua was built to ease the development of ontologies with a form-based Web interface. It is a tool that supports distributed, collaborative editing, browsing and creation of Ontolingua ontologies. Ontolingua uses an OKBC model with full KIF axioms. The base language is Ontolingua. It is possible to compare ontologies. Users can reuse existing ontologies from a modular structured library. It is important that users have some notions of KIF and of the Ontolingua language. The interface is not very user friendly. It is possible to export or import to the following formats: DAML+OIL, KIF, OKBC, LOOM, Prolog, Ontolingua, CLIPS.

Altova SemanticWorks is a commercial visual editor that has an intuitive visual interface and drag-and-drop functionalities. Users are capable of printing the graphical RDF and OWL representations to create documentation for Semantic Web implementations. It is possible to switch from the graphical RDF/OWL view to the text view to see how documents are being built

in RDF/XML, OWL or N-triples format. The code is automatically generated based on the user's design. This editor has the ability to manage the following files: N-triples, XML, OWL, RDF and RDFS.

OilEd's interface was strongly influenced by Stanford's Protégé toolkit. This editor does not provide a full ontology development environment. However, allows users to build ontologies and to check ontologies for consistency by using FaCT reasoner. OilEd's internal data format is DAML+OIL. It is possible to import from DAML+OIL and export ontologies as RDFS, DAML OWL, RDF/XML and others formats.

WebODE is a Web application. This editor supports ontology edition, navigation, documentation, merge, reasoning and other activities involved in the ontology development process. User groups can be created to collaborate in the edition of ontologies. WebODE has automatic exportation and importation services from and into XML. This tool supoorts Methontology methodology.

pOWL (Auer, 2005) is a PHP-based open source ontology management tool. pOWL is capable of supporting parsing, storing, querying, manipulation, versioning, serving and serialization of RDFS and OWL knowledge bases in a collaborative Web enabled environment. This editor was designed to work with ontologies of arbitrary size. pOWL offers an RDQL query builder. pOWL has versioning control. pOWL supports heterogeneous data and its formal description.

SWOOP is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. It is possible to have collaborative annotation with the Annotea plug-in. Users can reuse external ontological data.

To sum up, there are commercial ontology tools (Semantic Works and OntoStudio), there are ontology tools that demand learning/knowing a specific language (Ontolingua and OilEd) and there are ontology tools that are more graphic (IsaViz and Semantic Works). Other tools are Web-based application (WebODE, pOWL and SWOOP) or follow a methodology (DOE and WebODE).

There are several available ontology tools that can help to build an ontology. Some tools only support common edition and browsing functionalities. Other tools provide ontology documentation, ontology import/export for different formats, graphical view of ontologies, ontology libraries and attached inference engines.

In this chapter, we compared a selection of some of the available ontologies tool and environments that can be used for ontology construction, either from scratch or by reusing other existing ontologies.

As this overview shows, there are many ontology development tools for ontology construction

Tool	Product or Project Web Page
Apollo	http://apollo.open.ac.uk/index.html
CoGITaNT	http://cogitant.sourceforge.net/
DAMLImp (API)	http://codip.grci.com/Tools/Components.html
Differential Ontology Editor (DOE)	http://opales.ina.fr/public/
Disciple Learning Agent Shell	http://lalab.gmu.edu/
DUET	http://codip.grci.com/Tools/Tools.html
Enterprise Semantic Platform (ESP) including Knowledge Toolkit	http://www.semagix.com/
GALEN Case Environment (GCE)	http://www.kermanog.com/
ICOM	http://www.cs.man.ac.uk/~franconi/icom/
Integrated Ontology Development Environment	http://www.ontologyworks.com/
IsaViz	http://www.w3.org/2001/11/IsaViz/
JOE	http://www.cse.sc.edu/research/cit/demos/java/joe/
KAON (including OIModeller)	http://kaon.semanticweb.org/
KBE -- Knowledge Base Editor (for Zeus AgentBuilding Toolkit)	http://www.isis.vanderbilt.edu/Projects/micants/Tech/Demos/KBE/
LegendBurster Ontology Editor	http://www.georeferenceonline.com/
LinKFactory Workbench	http://www.landc.be/
Medius Visual Ontology Modeler	http://www.sandsoft.com/products.html
NeoClassic	http://www-out.bell-labs.com/project/classic/
OilEd	http://oiled.man.ac.uk/

Table 5: List of some of the representative ontology tools

Tool	Product or Project Web Page
Onto-Builder	http://ontology.univ-savoie.fr/
OntoEdit	http://www.ontoprise.de/com/ontoedit.htm
Ontolingua with Chimaera	http://www.ksl.stanford.edu/software/ontolingua/
Ontopia Knowledge Suite	http://www.ontopia.net/solutions/products.html
Ontosaurus	http://www.isi.edu/isd/ontosaurus.html
OntoTerm	http://www.ontoterm.com/
OpenCyc Knowledge Server	http://www.opencyc.org/
OpenKnoMe	http://www.topthing.com/
PC Pack 4	http://www.epistemics.co.uk/
Protégé-2000	http://protege.stanford.edu/index.html
RDFAuthor	http://rdfweb.org/people/damian/RDFAuthor/
RDFedt	http://www.jan-winkler.de/dev/e_rdfe.htm
SemTalk	http://www.semtalk.com/
Specware	http://www.specware.org/
Taxonomy Builder	http://www.semansys.com/about_composer.html
TOPKAT	http://www.aiai.ed.ac.uk/~jkk/topkat.html
WebKB	http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/generalDoc.html
WebODE	http://delicias.dia.fi.upm.es/webODE/
WebOnto	http://kmi.open.ac.uk/projects/webonto/

Table 6: List of some of the representative ontology tools

Feature	Protégé	OntoEdit	DOE	IsaViz	Ontolingua	SemanticWorks 2006	OILEd	SWOOP	pOWL	WebOde
Versioning	✓	✓	×	×	×	×	×	✓	×	×
Collaborative ontology edition	✓	✓	×	×		×	×	✓	✓	✓
Graphical class/properties and taxonomy	✓	✓	✓	×	✓	✓	✓	✓	✓	✓
Graphical tree view	✓	×	×	✓	✓	✓	✓	×	×	✓
Support growth of large scale ontologies	✓	✓	×	✓	✓	✓	×	✓	✓	✓
Querying	×	✓	×	×	×	×	×	×	×	×
Consistency Checking	✓	✓	×	×	×	×	✓	×	×	✓
OWL Editor	✓	✓	✓		✓	✓	✓	✓	✓	✓
Supports Methodologies	×	×	Bruno Bachimont		×	×	×	×	×	Methontology
User Interface (multi-language, intuitive)	✓	✓	×	✓	×	✓	✓	✓	✓	✓

Table 7: Most representative features

The following table (Table 7) lists the features that we considered more important when deciding which ontology tool to use: versioning; collaborative ontology edition; graphical class/properties taxonomy; graphical tree view; support the growth of large scale ontologies; querying; friendly user interface; consistency check and OWL editor.

The feature versioning keeps track of the version evolution that the ontology suffers.

The collaborative ontology edition is a very useful feature since it allows users to edit ontologies in a collaborative manner.

The graphical class/properties taxonomy provides an interesting view of the classes and of the properties of the ontology. This feature permits viewing classes and properties with graphically.

The graphical tree view shows a generic graph view of the ontology. It is possible to have an overview of the structure of the ontology.

If the ontology editor supports the growth of large scale ontologies, then the user can be sure that it is possible to build an ontology by using only one editor.

The feature concerning querying allows to query the knowledge base.

Consistency checking is important in order to look for propagation of the arity all along the hierarchy and inheritance of domains.

There are only two tools that only support one specific methodology, Doe and WebODE.

It is important that the user interface is friendly and similar to others interface. As a result, the user does not need to spend to much time getting to know the tool.

5 SEED PROJECT

Tourism has become one of the world's largest industry players and its growth shows a consistent year to year increase with highly competitive business for tourism destinations all over the world. Competitive advantage is no longer natural, but increasingly driven by science, information technology and innovation.

The internet is already the primary source of tourist destination information for travelers. The number of people turning to the Internet for vacation and travel planning has increased more than 300% over the past five years. It has outpaced traditional sources of information on tourist destinations within a short period of time. One major cause for the growth of e-tourism is that it extends existing business models, reduces costs, and expands and introduces new distribution channels.

The Semantic E-tourism Dynamic Packaging System (SEED) research laboratory was established in 2005 by Prof. Jorge Cardoso with the help of Eng. Jorge Fernandes from Expedita (Expedita, 1996). The SEED research laboratory is a consortium of professionals in disciplines related to SW technologies and Business Process Modelling. This laboratory is specialized in the practical application of advanced technologies to modern information systems in order to solve problems such as data integration, search, indexing, process modelling, ontologies and workflows.

Initially, the SEED laboratory had as main objective the development of SW technologies and solutions for the tourism industry. In 2006, the laboratory established two main areas of research Business Process Modelling and SW. Research projects focus in the areas of Workflows, Business Processes Analysis, Information Extraction and Integration, Web Services and the SW.

SEED project can be divided in the following areas: 1) data extraction, 2) data models mapping, 3) ontology for e-tourism, 4) packaging architecture, 5) semantic rules and 6) tourism industry IT/IS.

5.1 DATA EXTRACTION

If an enterprise wants to build an automatic system to create packages of products, it must have access to data from all products. Products data integration is easy when there is some B2B protocol between enterprises. Since different enterprises use different ways to provide product information, the system must support multi-sources and different data types.

The objective of this project was to develop a platform that allows dynamic integration of different information sources and channel data, based on a query, to a single output. This project was Bruno Silva responsibility.

The system must gather information from typical B2B (Business-to-Business) and B2C (Business-to-Clients) formats (semi-structured for B2B and unstructured/semi-structured for B2C) and wrap the obtained data to an (semi-structured) integration format. The project will involve dealing with three conceptual technologies: extractors, wrappers, and mediators, which are described in the following sections.

Besides handling different data types, this tool must be able to:

- Manage data sources;
- Connect to remote and local data sources;
- Know how (and where) to get an specific information;
- Respond to data retrieval queries;
- Handle query and extraction exceptions.

5.2 EXTRACTORS , WRAPPERS AND MEDIATORS

An extractor is a software component that extracts data from Web sites. The role of the extractor is to convert information implicitly stored as an HTML document which consists of plain text with some tags, into information explicitly stored as part of a data-structure. HTML documents do not contain any type of information.

Using wrappers to extract information from the Web was one of the most used way so far. A wrapper is in charge of transforming the extracted information into the target structure that has to be used outside of the wrapper according to the user's needs. It should take advantage of generic conversion tools that can directly map extracted string into say dates, zip codes or phone numbers.

Mediator systems provide users with uniform access to various data sources. Mediator often must reconcile different representations of the same concept. For example, one system might

measure altitude in meters from the earth's surface while another measures it in miles from the earth's center. In the future, application developers may define interfaces in terms of abstract attributes using "self-description" - for example, Altitude (datatype=integer, units=miles).

5.3 Data Models Mapping

Today's enterprise face critical needs in integrating disparate information spread over several data sources inside and even outside the organization. SW technologies, such as ontologies, play an important role in semantic integration of data.

The area concerning Data Models Mapping was of Toni Rodrigues and Pedro Costa responsibility. The objective of this project was to develop a user-friendly interactive mapping tool that allows a user to map syntactic data in a XML format to concepts of an existing ontology. Although this project is itself part of the overall SEED Project and so is closely related with e-Tourism, this tool must nevertheless generate mapping rules that allow converting any XML structured syntactic data to instances of any existing ontology, in OWL language, regardless of its application domain.

5.4 Ontology for E-Tourism

The Semantic Web is an extension of the current Web that adds technological infrastructure for better knowledge representation, interpretation, and reasoning (Berners-Lee e t al., 2001). Ontologies form one of the most important layers in the Semantic Web architecture. They can enhance the functioning of the Web in many ways. Ce-TO is a domain-specific ontology that is being developed with a travel-centric approach. It reflects the consensus among people for its existence, maintenance, and effectiveness. This ontology defines concepts related to the complementary e-tourism offer.

5.5 Packaging Architecture

Dynamic packaging has been introduced as an innovative technology allowing for the automated online configuration and assembly of package travel products for individual customers. A dynamic packaging infrastructure requires integrating data in an automated way for querying in a uniform way and across multiple heterogeneous systems containing tourism

related information. The technology includes the ability to combine multiple travel components in real time and provides a single, fully priced package. One big challenge to develop dynamic packaging applications is to find a solution to cope and integrate the non-standard way of defining e-tourism products and services.

This project main goal is to develop a platform to enable dynamic packaging using the latest Internet technologies, such as Semantic web, Ontologies, Web services, and Web processes.. Miguel Gouveia is in charge of this project.

5.6 SEMANTIC RULES

In the travel industry most organizations do not formaly identify and store rules. Instead, although travel managers and travel agents use rules periodically they exist only in the software code that runs packaging applications. Rules are 'lost' in application code. As a result, the people that are directly in contact with the rules that dictate what sort of travel packages should be created at a given time of year under specific market conditions, are not travel managers and travel agents but rather the IS staff who convert packaging requirements into lines of code. Moreover, when rules are embedded in application code it becomes difficult to locate and change business logic and each alteration requires recompiling application code. Separating packaging rules from application code allows packaging policies to be easily communicated and understood by all employees and rules can be managed in isolation from application code. Packaging rules can be expressed using formal languages. Examples of languages to define rules include Unified Modeling Language (UML), the ILOG rules language the Business Rules Markup Language (BRML), and Rule Markup Language (RuleML). The major goal is to study the how semantics packaging rules can be used to create dynamic packages. This project main goal is to create semantic rules for the tourism industry. Jorge Sousa is responsible for this project.

5.7 TOURISM INDUSTRY IT/IS

Evidence indicates that the effective use of information technology is crucial for tourism business' competitiveness and prosperity, as it influences their ability to differentiate their offerings as well as their production and delivery costs. Because tourism is an information-based industry it is one of the natural leading industries on the Internet. For example, it is anticipated that most sectors in the travel industry throughout the world will have a web site on the Internet. Thus, it is vital for every tourism destination and travel business to embrace the use of information technology and exploit its potential.

Tourism Information Systems (TIS) are a type of business systems that serve and support e-tourism and e-travel, such as airlines, hoteliers, car rental companies, leisure suppliers, and travel agencies. These systems rely on travel related information sources to create tourism products and services. The information present on these sources can serve as the springboard for the development of a variety of systems, including dynamic packaging applications, travel planning engines, and price comparison applications.

Influenced by the booming tourism sector and the increasing spectrum of travel alternatives, travel agents started to assemble customized tourism itineraries or packages, in order to fulfil the individual needs of travellers in a fast growing market.

With the growth of demand for customized tourism itineraries, travel agents, tour operators, and intermediaries seek new technologies that provide their personnel and clients the flexibility to put together unique dynamic packages from a range of alternatives, without having to be aware of the intricacy of contract rules and pricing issues.

The concept of dynamic packaging is based on an individual consumer request, including the ability to combine, multiple travel components such as flights, hotels, car rentals, and any other tourism related component in real time and provides a single, fully priced package, requiring only one payment from the consumer and hiding the pricing of individual components. The products available to the customer can be stored in local inventories or external heterogeneous data sources.

The recent developments of technologies, such as knowledge bases, ontologies, and semantic Web, can enable to build a new breed of dynamic packaging solutions. Using these latest technologies, we present a new packaging model, the knowledge-based dynamic packaging model. The model aims at pro-actively offering wrapped packages which include destination information, information from social networks and public websites. For knowledge-based dynamic packaging applications to be successful it is indispensable to study their architecture. Therefore, we undertake a study of our approach to dynamic packaging application development by presenting a packaging architecture based on three main layers: Data Integration layer, Packaging layer and Presentation layer.

The data integration layer is responsible for tourism information systems data integration. Data integration is a challenge for dynamic packaging applications since they need to query across multiple heterogeneous, autonomous, and distributed tourism data sources produced independently by multiple organizations in the travel industry. To develop a knowledge-based dynamic packaging application it is important to identify each data source according to its type of data since the type of data will influence our selection of a solution to achieve data integration. Due to the high level of autonomy and heterogeneity of tourism information systems, dynamic packaging systems cannot be successfully developed by considering only syntactic and structural integration. One important aspect that needs to be contemplated to develop a new breed of dynamic packaging systems is semantic heterogeneity in order to reduce the potential failures that may occur when integrating tourism information systems. The development of suitable ontologies for the tourism industry can serve as a common language

for tourism-related terminology and a mechanism for promoting the seamless exchange of information across all travel industry segments.

The packaging layer supports several packaging models, such as the prepackaging, pre-designed framework packaging, the component packaging and the dynamic packaging model. As we have already mentioned, our work adds to this layer a new model which is motivated by a knowledge-based approach. The proposed new model analysis input information describing consumer experiences/reviews and individual user profiles gathered through social networks and personalized user accounts in order to create tailor-made packages with maximum customization.

The presentation layer is fundamental to determine if a knowledge-based dynamic package will be successful or not. This layer is in charge of wrapping dynamic packages. The question of how to wrap the package in order to attract the traveller is essential for turning lookers into bookers. Creating a suitable individual package, regarding that one can value this soonest after the trip, can only be achieved if the presentation layer is able to create a clear picture of the product in the customers mind.

Through its collaborations with other research institutes partners, the SEED is able to achieve significant research at an international level. Partners include the Technical University of Eindhoven (The Netherland) and the WU Wien (Austria).bnb (Cardoso and Fernandes, 2005).

6 DEVELOPING ONTOLOGIES USING THE ZACHMAN FRAMEWORK

Ontologies are software products that are used for the purpose of enabling knowledge sharing and reuse among people and software systems. They form one of the most important layers in the Semantic Web architecture (Berners-Lee, 2001).

But what are ontologies? Ontologies are similar to taxonomies but use richer semantic relationships among terms and attributes, as well as strict rules about how to specify terms and relationships (Cardoso, 2005). In computer science, ontologies have emerged from the area of artificial intelligence (AI). They have generally been associated with logical inferencing and, recently, have begun to be applied to the Semantic Web. An ontology is a shared conceptualization of the world which consist of definitional aspects such as high-level schemas and assertional aspects such as entities, domain vocabulary and factual knowledge- all connected in a semantic manner (Sheth, 2003).

But why are ontologies so important? Ontologies are important since they 1) promote and facilitate interoperability among information systems, 2) provide a shared and common understanding of a domain that can be communicated between people, and heterogeneous and widely distributed application systems. They are reusable and sharable artifacts that have to be developed in a machine interpretable language (Gruber, 1993, Studer et al., 1998). Ontololgy engineers are interested in how ontologies can be used to represent reusable and sharable pieces of domain knowledge and how can they be used in applications.

But how can we build an ontology? At the beginning of the 1990s, ontologies were built using mainly AI modeling techniques based on frames and first order-logic (FOL) (Gómez-Pérez et al., 2004). However, other knowledge representation techniques based on description logics have been also used to build ontologies (Baader et. al., 2003). In the context of the Semantic Web new description logic languages have emerged such as Ontology Interchange Language and Ontology Inference Layer (OIL), DARPA Agent Markup Language (DAML), DAML+OIL and Ontology Web Language (OWL). According to Asunción Gómez-Pérez and colleagues, (Gómez-Pérez et al, 2004) there are important connections and implications between the knowledge modeling components used to build ontologies, the knowledge representation

paradigms used to represent formally the components, and the languages used to implement the ontology under a given knowledge representation paradigm. An ontology built with frames or description logics (DL) can be implemented in several languages since these frames or DL are independent from any language. These frames or DL work as guidelines during the development process of an ontology. It is stated that other techniques widely used in software engineering (SE) and databases for modeling concepts, relationships between concepts, and concept attributes could also be appropriated for building ontologies because these techniques enforce a structure to the domain knowledge and constrain the interpretation of terms (Gómez-Pérez et al., 2004).

A number of different techniques for SE have been proposed for develop software, each exhibiting strengths and weaknesses, but all have a series of generic phases in common. According to Pressman, software delivers the most important product of our time: information. Software is developed or engineered; it is not manufactured in the classical sense. Similarly, ontologies also "deliver" information and are not manufactured. Many of the ontologists have knowledge concerning engineering. Therefore, some of them have borrowed from the SE, methodologies and principles to support the ontology development process. It is possible to state that the methodologies for ontology development have their roots on software methodologies since the ontology development process follows several of the models and techniques from SE. Some of the questions that arise when selecting a language or a tool for ontology construction are the same as when building software. There are several methodologies available. All of them have advantages and disadvantages.

Until now, several ontologies have been developed by different groups, under different approaches, and with different methods and techniques. The set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies is known as ontological engineering (OE) (Gómez Pérez et al., 2004).

While the ontological engineering and the SE have several aspects in common, we can find important differences. These differences force us to adapt and extend existing methodologies from the SE field that might fulfil the needs of OE. For example, OE requires a stricter and explicit description of the relationships between concepts, while SE targets mainly the description of the components of the system and not always targets the relationships between components. Furthermore, ontologies are closer to humans and software is closer to machines. Therefore, it seems natural that methodologies to develop ontologies should account for human behaviour and language considerations.

According to Enderton (Enderton, 1972), ontologies are often equated with taxonomic hierarchies of classes, class definitions, and the subsumption relation, but ontologies need not be limited to these forms. Ontologies are also not limited to conservative definitions, that is, definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world. For example, ontologies provide means of inferencing knowledge by adding different types of relationships, such as hierarchy parent child relationships. It also

considers other types of relationships like equivalence, homographic, hierarchical and associative relationships. To model and organize concepts to semantically illustrate knowledge about the world, relationships, constraints and rules are well thought-out.

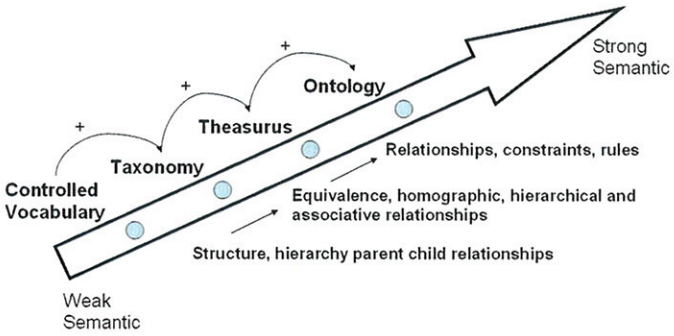

Figure 18: Levels of Semantic (Cardoso, 2005)

From all of the SE methodologies it is possible to point out the Zachman Framework (ZF). This framework is a popular Enterprise Architecture tool. According to Diann Daniel (Daniel, 2007) the goal of enterprise architecture is to create unity. In its simplest terms, enterprise architecture is the process of aligning a business's strategic vision with its information technology. It connects different business units for synergistic communication and collaboration, making it easier for users to share knowledge. This is also the main goal of an ontology.

Peter Heinckien, chief enterprise architecture at Toyota Europe, states that the enterprise architecture must chart, define and standardize technology, data and business processes. These are main goals of the development of ontologies too. The enterprise architecture transforms tech-speak into language of business solutions (Daniel, 2007).

The ZF can be divided into two sections. The first section concerns the logical behavior of the system (first three rows) and the other section the physical behavior of the system (the other three rows). It is also possible to call the first section as the "human view" and the second as the "machine view". Therefore, we believe that this is the methodology that needs to be considered when building ontologies since it portraits two different views. These views are considered very important to the ontology development since the other methodologies were very "machine view".

Another advantage of using this framework is the fact that instead of representing the process of development as a series of steps, it is organized around the points of view taken by the various players. We believe that by replacing the technical activities and the support activities (activities that were done in parallel) suggested by the Methontology methodology it is possible to provide a methodology that offers more than one point of view.

However, the management activities that this methodology proposes are very important since they include scheduling, control and quality assurance. The scheduling activity identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion. The control activity guarantees that scheduled tasks are completed in the manner

intended to be performed. The quality assurance activity assures that the quality of each and every product output is satisfactory. These activities are similar to the ones done in the earlier stages of the software development concerning organization/planification of the project.

Therefore, the management activities should be implemented before the ZF and the technical and support activities should be replaced by the ZF, as the following figure shows.

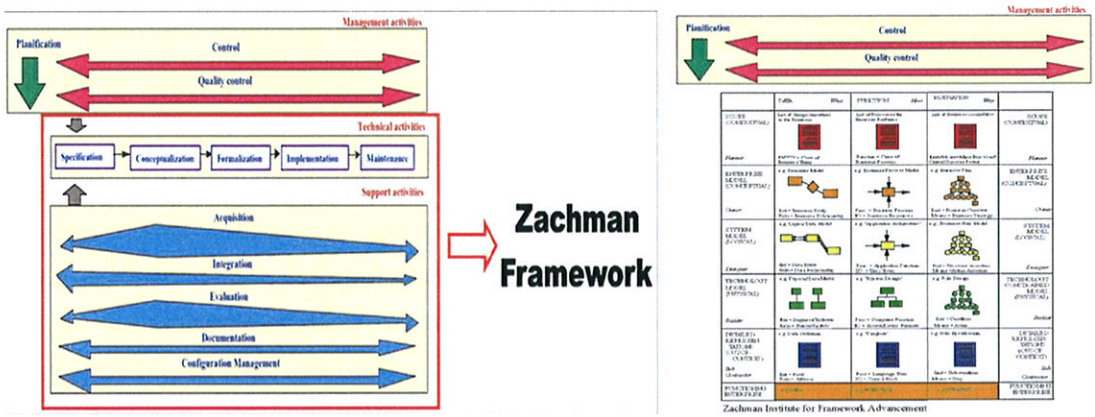

Figure 19: Replacement of the technical and support activities by the Zachman Framework

Our new methodology is organized as followed: 1) management activities and 2) ZF since before starting the ontology development we need to specify requirements for the project of building an ontology.

The replacement of the technical and support activities can be mapped to the ZF. Since these two activities are done in parallel, it is possible to draw the following diagram:

Zachman Framework	Technical Activities	Support Activities
A (Scope)	Specification	Start Acquisition, Integration, Evaluation, Documentation and Configuration Management
B (Enterprise Model)	Conceptualization	Acquisition and Evaluation are the main activities
C (System Model)	Formalization	Continue with Acquisition, Integration, Evaluation, Documentation and Configuration Management
D (Technology Model)	Implementation	
E (Detailed Representation)	Maintenance	

Table 8: Table that maps the perspectives with the activities identified by Methontology Methodology

As the table above show, it is possible to map the technical and support activities to the perspectives (rows) of the ZF. However, there are also columns that represent different areas of interest for each perspective. This is not considered in the Methontology methodology. Let us take a look at the first perspective and the data column and the function column. The first row and the first column, begins by specifying the concepts important to the domain. In the second

column, instead of concepts important to the domain, relationships concerning the domain are specified.

6.1 THE ZACHMAN FRAMEWORK

In 1987, John Zachman published a different approach to the elements of system development (Zachman, 1987). The ZF draws upon the discipline of classical architecture to establish a common vocabulary and set of perspectives, a framework, for defining and describing today's complex enterprise systems.

Enterprise Architecture provides the blueprint, or architecture, for the organization's information infrastructure. Instead of representing the process as a series of steps, he organized it around the points of view taken by the various players. These players included: 1) the Chief Executive Officer (CEO) or whoever is setting the agenda and strategy for an organization, 2) the business people who run the organization, 3) the systems analyst who wants to represent the business in a disciplined form, 4) the designer, who applies specific technologies to solve the problems of the business, and finally, 5) the system itself. The ZF represents each of these perspectives as a row in his matrix. He then defined columns in the matrix to represent the kinds of things people should be looking at. These include functions and data, as addressed by most methodologies.

In addition, however, Mr. Zachman has set up columns to represent locations (where business is conducted), the people and organizations involved, events which cause things to happen, and the motivations and constraints which determine how the business behaves.

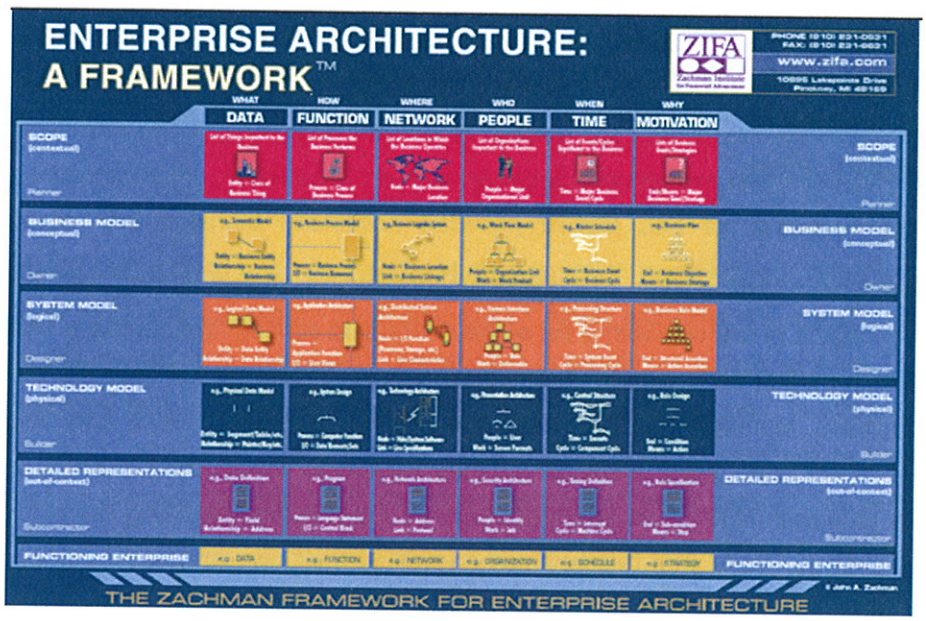

Figure 20: The Zachman Framework (ZIFA)

Zachman's Framework is organized as 36 cells arranged in a six by-six matrix-six rows and six columns. It is a two dimensional schema, used to organize the detailed representations of the enterprise (O'Rourke, et al., 2003). The six rows of the framework represent six perspectives, as viewed by the planner, owner, designer, builder, subcontractor, and functioning enterprise. The framework's six columns represent different aspects: things, processes, connectivity, people, timing and motivation (O'Rourke et al., 2003).

6.1.1 Perspectives

A perspective is the result of an ordered logical method used to break an issue or topics into unique viewpoints or frames of reference. An overview of the perspectives is given.

1. Scope (Planner's view): This is simply a list of major business goals. Defines the "universe of discourse" relative to motivation.

2. Enterprise Model (Owner's view): This is a model of the business objectives and strategies of the Enterprise that constitute the motivation behind Enterprise operations and decisions.

3. System Model (Designer's view): This is a logical model of the business rules of the enterprise in terms of their intent (ends) and the constraints (means).

4. Technology model (Builder's view): This is a physical specification of the Business Rules. The rules are not presently factored out from their implementations and therefore are found as cardinality and optionally in the data models as procedural code or a policy specification.

5. Detailed representations (Subcontractor's view): Here a particular language is chosen, and the program listings, database specifications, networks, and so forth are all produced.

6. Functioning enterprise: Finally, a system is implemented and made part of an organization.

6.1.2 Aspects

The columns in the ZF represent different aspects of interest for each perspective. An aspect is the result of an ordered logical method used to break an issue or topic into defined basic parts (Select Business Solutions, 2006)

As shown in Figure 20, these are:

1. Data: Each of the rows in this column addresses understanding of and dealing with an enterprise's data. This begins in row one with a list of the things that concern the company and affect its direction and purpose. Row two, is a contiguous model of the things seen by the participants in the business. Many-to-many and n-ary relationships may be present, reflecting the way the business views them. Also, relationships may be shown which themselves have attributes. Row three provides more of an information-based perspective, resolving many-to-many and n-ary relationships, along with relationships containing their own attributes. Indeed, attributes are more exhaustively defined, and unique identifiers are specified. Entities are generalized to more closely reflect the underlying structure of the business and its relationships. In row four, entities are converted to table definitions, object classes, hierarchy segments, or

whatever is appropriate for the kind of data base management system to be used. This is tantamount to creating the data definition language statements. In row five, the tables are actually implemented on physical disk drives, using the underlying organization of the database management system. This is where table spaces are defined, disk packs are allocated, and so forth. The actual database itself is created and initial data are converted and loaded for row six. This column is associated with the interrogative "what" (Hay, 1997).

2. Function: The rows in the function column describe the process of translating the mission of the enterprise into successively more detailed definitions of its operations. Where row one is a list of the kinds of activities the enterprise conducts, row two describes these activities in a contiguous model. Row three portrays them in terms of data transforming processes, described exclusively in terms of the conversion of input data into output data. The technology model in row four then converts these data conversion processes into the definition of program modules and how they interact with each other. Pseudo-code is produced here. Row five then converts these into source and object code. Row six is where the code is linked and converted to executable programs. This column is associated with the interrogative "how" (Hay, 1997).

3. Network: This column is concerned with the geographical distribution of the enterprise's activities. At the strategic level (row one), this is simply a listing of the places where the enterprise does business. At row two, this becomes a more detailed communications chart, describing how the various locations interact with each other. Row three produces the architecture for data distribution, itemizing what information is created where and where it is to klbe used. In row four, this distribution is translated into the kinds of computer facilities that are required in each location, and in row five, these facilities requirements are translated into specification of particular computers, protocols, communications facilities, and the like. Row six describes the implemented communications facilities. This column is associated with the interrogative "where" (Hay, 1997).

4. People: The fourth column describes who is involved in the business and in the introduction of new technology. The model of people is a simple list of the organizational units and each unit's mission. In row two, this list is fleshed out into a full organization chart, linked to the function column. Here also, requirements for security are described in general terms. In row three, the potential interaction between people and technology begins to be specified, specifically in terms of who needs what information to do his job. What roles do each play and what data are necessary for those roles? Along with this are specific definitions of security requirements, in terms of who (which role) is permitted access to what. In row four, the actual interface between each person and the technology is designed. In this row, issues of interface graphics, navigation paths, security rules and presentation style are addressed. In row five, this design is converted into the outward appearance of each program, as well as the definitions of access permissions in terms of specific tables and/or columns each user can have access to. In row six, you have trained people, using the new system. This column is associated with the interrogative "who" (Hay, 1997).

5. Time: The fifth column describes the effects of time on the enterprise. It is difficult to describe or address this column in isolation from the others, especially column two. At the strategic (row one) level, this is a description of the business cycle and overall business events. In the detailed model of the business (row two), the time column defines when functions are to happen and under what circumstances. Row three defines the business events which cause specific data transformations and entity state changes to take place. In the technology model (row four), the events become program triggers and messages, and the information processing responses are designed in detail. In row five, these designs become specific programs. In row six business events are correctly responded to by the system. This column is associated with the interrogative "when" (Hay, 1997).

6. Motivation: As Mr. Zachman describes it, this is concerned with the translation of business goals and strategies into specific ends and means. This can be expanded to include the entire set of constraints that apply to an enterprise's efforts. In row one, the enterprise identifies its goals and strategies in general, common language terms. In row two, these are translated into the specific rules and constraints that apply to an enterprise's operation. In row three, business rules may be expressed in terms of information that is and is not permitted to exist. This includes constraints on the creation of rows in a database as well as on the updating of specific values. In row four, these business rules will be converted to program design elements, and in row five they will become specific programs. In row six, business rules are enforced. This column is associated with the interrogative "why" (Hay, 1997).

According to Crawford Parrish (Parrish, 2004), the ZF is a logical structure intended to provide a comprehensive representation of an information technology enterprise. It allows for multiple perspectives and categorization of business artefacts. The ZF gathers and refines principles from older methods. It has a structure (or framework) independent of tools and methods used in any particular IT business. The framework defines how perspectives are related according to certain rules or abstractions (Parrish, 2004)

6.2 A NEW METHODOLOGY BASED IN THE ZACHMAN FRAMEWORK

In this section a description of the Zachman Ontological Framework (ZOF) is done and an overview of how our solution is structured so as to provide guidelines to assist the ontological engineering in making choices at a variety of levels, from high level structure of the ontology, to the fine detail of whether or not to include some particular distinction.

Most of the methods and methodologies for building ontologies are focused on the development activities, specially on the ontology conceptualization and ontology implementation, and they do not pay too much attention to other important aspects related to management, integration, evolution and evaluation of ontologies.

The ZOF methodology is structured as followed: (1) Management Activities that include (a) a planning task, (b) a control task and (c) a quality control task and (2) the ZOF. These two activities are done in parallel.

	Data	Function	Motivation
Scope (Contextual) Planner	List of things important to the domain	List of activities important to the domain	List of competency and purpose questions and answers
Enterprise Model (Conceptual) Owner	Controlled Vocabulary	Describe Internal Structure of concepts	Domain Rules
System Model (Logical) Designer	Taxonomy	Domain and range of properties	Rule Definition
Technology Model (Physical) Builder	Thesaurus	Select an ontology tool and language	Formalization of rules
Detailed Representations (Out-of-context) Subcontractor	Analyse other known Ontologies related to the domain	Analyse relations between the concepts of known ontologies and define instances	Analyse other Business Related Rules
Functioning Enterprise	Modelled Ontology	Formalized Ontology	Rules Document

Figure 21: New methodology build according to the Zachman Framework

As with the ZF, this new framework for ontology development has six perspectives. For now, we will only consider three columns since we decided that they were the most important aspects during the ontology development and since we believe they are the minimal required aspects for now. Furthermore, almost all the methodologies presented in section 2 address and represent these aspects. These methodologies all try to answer the following questions: "What", "How" and "Why". The rows represent the different perspectives of the domain (a different approach from the ones presented in the methodologies overviewed), while the columns represent different aspects of the domain.

The other tree columns were not considered since they represented different kinds of aspects that we believed that were not as essential as these. An ontology does not have a geographical space where it has to happen. Therefore, the network column is not needed since this column is concerned with the geographical distribution of business activities. This ontology will help promote interoperability between systems that are located in different areas. An ontology does not have organizational units and there is no need to identify each unit's goal. Therefore, we

believe that this column does not need to be considered. The column Time was not considered since time is not a relevant aspect of the ontology development.

To sum up, in our opinion, these columns were not considered since the information concerning them is not significant for this domain ontology.

6.2.1 Perspectives

The perspectives proposed by the ZF are all considered in this new methodology. There are six perspectives: (1) Scope (Contextual), (2) Enterprise Model (Conceptual), (3) System Model (Logical), (4) Technology Model (Physical), (5) Detailed Representations (Out of Context) and (6) Functioning Enterprise. Since we are developing ontologies we decided to identify the rows with only one concept. The concepts are: for the first row – contextual, for the second row-conceptual, for the third row – logical, for the fifth row – physical, for the sixth row – out of context and the last row has the same name. These concepts had already been identified by John Zachman. Each row had 3 names associated, we simply selected the one we though described the row best.

A brief description of each of the perspectives is given.

1. Contextual: Definition of the domain features and the ontology purpose. This is necessary to establish the context in which the ontology will be used. In order words, defines the domain direction and purpose, defining the boundaries of the ontology. A well-characterised requirements specification is important to the design. The use to which the ontology is put has a great effect on the content and style of that ontology. Informal questions will be asked and the ontology must be able to answer since they will be used to check that the ontology fits its purpose.

2. Conceptual: This perspective defines - in ontological terms- the nature of the ontology, including its structure, concepts, relationships and so forth. After identifying the key terms that exist in the domain, their properties and the relationships that hold between them appear. The relationships describing the domain are captured. At this row, it is important to bear the results of the first row, that of requirements gathering, in mind.

3. Logical: Defines the ontology in more rigorous terms than in row 2, basically taking the model to a greater level of detail. This row represents the conceptual row in some formal language, for e.g., object models or logic.

4. Physical: Constraints, rules and other relationships are considered. A thesaurus is built and properties are defined in a more detailed manner. This row is concerned in describing how the technology will be used to address information processing needs.

5. Out of Context: A language and tool for ontology development is chosen based on the characteristic of the project. Other ontologies and Business Rules concerning the domain are analysed. It is always worth considering what somenone has done and check if it can redefine and extend existing sources for our domain.

6. Functioning Enterprise: Finally, an ontology is implemented and made part of a system. The evaluation is made by validating the frame with respect to the ontology and documentation produced. The physical manifestation of the end product (ontology).

To sum up, the rows represent the different perception of the domain according to the business goals. However, it is important to remind that the models within a single row should be consistent with one another.

6.2.2 Aspects

The ZOF represents different areas of interest for each perspective. The columns describe dimensions of the ontology development effort. Even though there are six columns we only considered, for now, three of them. The first column (Data) identifies the concepts of significance, and the relationships between them, within the domain in study. The second column (Function) focuses on the processes of the domain that are performed in order to support itself and its consumers. The Motivation column translates business goals, strategies, and constraints into specific implementations. Within a column the models are adapted to reflect the views of the different perspectives (contextual, conceptual, logical, out of context, functional enterprise) for each row.

An overview of the columns selected is given below.

Data Column: Each of the rows in this column addresses the concepts of a particular domain. This begins in row one with a list of things important for the domain and affects its contextualization. Row two, is a contiguous model of the things seen by the participants in the domain. The terms that compose the domain are described. Relationships are considered presented in row three, reflecting the way the domain is structured. Hierarchical parent child relationships are identified. It is more of an information-based perspective, resolving many-to-many and n-ary relationships, along with relationships containing their own attributes. Equivalence, homographic, hierarchical and associative relationships are identified. Concepts are generalized to more closely reflect the underlying structure of the domain and its relationships. In row four, terms are formalized. This is the first step to create the data definition language statements. In row five, this information is actually implemented. In the first three rows the data is analysed in a more logical way and its conceptualization is gradual. The last three rows formalise the data analysed and considered relevant to the domain. In row five, a modelled ontology is shown

Function column: The rows of the function column describe the business processes important to the domain. These business processes help to identify some of the properties of the terms identified. Row one aggregates all the activities presented in a particular domain. Row two describes the properties that each of these activities considers. In row three, the type of the properties are identified. The physical perspective, row four, selects an ontology tool and a language to implement the ontology. It answers to the question "how will we build the ontology?". The information gathered in row three is formalized and implemented. Row five maps the ontology into an ontology language using a suitable development tool. Row six

integrates the modelled ontology ready to be used in an operation system. Similarly to the object-oriented approach, functions and data tend to be addressed together. The result is an implemented ontology.

Motivation column: The rows in this column translate the domain goals and strategies into specific ends and means. This can be expanded to include the entire set of constraints that apply to a domain. In row one, the goals and strategies are identified using a natural language. In row two, these are translated into specific semantic rules and constraints that apply to a domain (business rules). In row three, these business rules are formalized. This includes constraints on the creation of relationships between terms. In row four, these business rules will be converted into a more formalised language. In row five a study of other business rules is made. This study helps to find new rules or to validate the rules that we already have formalize. In row six, business rules are enforced and a rule document is presented.

As this overview shows, these are the three columns and the six perspectives that we believe to be be crucial to the ontology development process since they allow acquisition, contextualization, conceptualization, formalization and implementation of a lot of information concerning the domain in analyse, as Jones identified (Jones at al., 2000). We believe that it is possible to extend the columns and consider the other aspects in a near future.

6.3 ZOF AND OTHER METHODOLOGIES

John Zachman has developed a "framework" for examine the body of knowledge as a whole, and see what sorts of general conclusions it is possible to draw from it (Hay, 1997). A framework is a set of assumptions, concepts, values, and practices that constitutes a view of the reality. We want to examine a certain domain and extract the information needed to provide a shared and common understanding of a domain that can be communicated between people, and heterogeneous and application systems.

ZF was selected as a basis for this new methodology since it has the following advantages:

- It is a proven methodology in software engineering since it has widely been used with success;
- It includes a step-by-step development plan and work breakdown structure because it is organized in rows and columns. It is possible to start the development in any of the cells;
- It is a tool for sharing and reusing enterprise information among people since each row is a perspective and each column an aspect that is easily understood;
- It explicitly shows that there are many views that need to be addressed when developing a product;
- It provides a reminder of the issues that need to be considered namely contextual, conceptual, logical , physical and out-of-context features;

- Different perspectives mean different views that mean different ways of looking, using and treating information;
- It has a comprehensive taxonomy since you do not have to be an engineer to understand it and use it;
- It is easy to change and maintain since it is organized in rows and columns.

ZF gathers and refines principles from older methodologies so as ZOF. After analyzing the methodologies presented previously, we mapped some of the activities that they proposed to a table.

	Data	Function	Motivation
Scope	• 2-Collect Data (IDEF5) • 2-Capturing Knowledge-Enterprise Model Approach • 1- Define a set of motivating scenarios – TOVE • 2-Knowledge acquisition – Methontology		• 1-Organize and define project- IDEF5 • 1-Identify the purpose of the ontology- Enterprise Model Approach • 2-Informal Competency questions- TOVE • 1-Specification- Methontology
Enterprise Model	• 3-Analyze data- IDEF5 3-Conceptualization- Methontology		
System Model	• 3-Define the terminology of the ontology- TOVE 4-Develop initial ontology- IDEF5		• 4-Formally redefine the competency questions
Technology Model	• 5-Define and validate ontology- IDEF5 • 6-Evaluation- Methontology	5-Define the semantics and constraints on the terminology using FOL- TOVE	
Detailed Representation	• 4-Integration – Methontology	• 5-Implementation- Methontology	• 6 and 7 Evaluation and documentation- Methontology
Functioning Enterprise	DATA	Modeled Ontology	Ontology

Table 9 : Overview of the methodologies for ontology development when compared to ZF.

This table shows how ZOF was influenced not only by the ZF but also by other methodologies for ontology development. As table 9 shows it is possible to map some of the activities proposed by the methodologies presented in section 2.1. to some of the cells that compose the ZOF. Therefore, ZOF follows some of the principles that the methodologies presented contain.

Each of the rows in the first column is concerned with understanding and dealing with data. Even though all the methodologies indicate this activity as one of the most important, they do not give any guidelines on how to collect data. Only TOVE methodology suggests defining a set of motivating scenarios concerning the domain. However, other types of representation need to be incorporated since these motivating scenarios are not sufficient. On the other hand, they are a good starting point. Hence we will adopt this approach.

The second column is not clearly separated in methodologies presented in section 2 from the first column. The activities concerned with collecting data and describing the processes occur together. In this framework that does not occur. We separate the data from the processes. Therefore, if we need to adjust or update any of these (data, processes), it is easier and faster. Another advantage comes from the fact that it is simpler to detail them.
The third column, Motivation, is concerned with the purpose of the ontology, why are we developing this ontology. In order to identify the purpose, TOVE, methodology suggests to define informal competency questions and to answer them. We will adopt them since they are easy to understand and a way to reveal the reasons, goals and strategies of the business.

6.4 RUNNING EXAMPLE

In this section we will show a running example of how the Zachman Ontology Framework (ZOF) can be used to build an ontology. It should be noticed that ZOF is to be generally complemented by management activities. Since these activities are out of scope of our work, for further information on this subject the reader is referred to the Methontology methodology proposed by Assunción Gómez-Pérez (Gómez-Pérez et al., 2004).

Our running example has been implemented and deployed at *Expedita-Arquitectura e Gestão de Sistemas de Informação Lda. Expedita* is an IT organization that has developed several projects for the tourism industry. The company participates in the project through its administrator, Eng. Jorge Fernandes, the co-PI of the project under which this work has been carried out.

Searching for better attending their clients, group Pestana in collaboration with Expedita started a project which main goal is to obtain an on-line tool to be used internally, by the internet site of the group and by customers for planning activities during their stay in any of the hotel group. By using this tool, customers can know what the offers are and make their reservations. The domain of the application is tourism, namely the complementary offer that is available. We plan to use this domain for web applications that present the complementary offers available or simply provide information about the resources/experiences available. Naturally the concepts describing different types of experiences (romance, in family, wellness) and main food categories (international, regional) will figure into our domain. Also the notion of a good combination for a day in family or a romance evening is going to be considered. At the same time, it is unlikely that this domain will include the concept accommodation even though this concept ais somewhat related to the idea of resources. These will happen since it is mainly a domain model for complementary offer to accommodation. In our running example, we will explain how we developed an ontology step by step for the complementary offer domain.

6.4.1 Ontology Development

Ontological engineering requires the definition and standardization of a life cycle ranging from requirements specification to maintenance, as well as methodologies and techniques that drive the ontology development. So, the ZOF methodology includes: (1) the identification of the ontology development process, the methodology itself, which specifies the steps for performing each activity, the techniques used, the products to be output, and how the ontology is going to be evaluated.

6.4.1.1 First Row

In this row it is necessary to establish the context that the ontology models. This row is concerned with the things that define the nature and purpose of the project. It defines the "universe of discourse" of the ontology. In other words, the ontology contextualization's main goal is to identify the ontology primary objective, purpose, granularity level, and scope.

External requirements are identified and purpose and competency questions are asked.

DATA COLUMN

The first row should be a list of terms, concepts, instances, attributes important to the domain. In order to obtain this list, a study should be done by analysing motivating scenarios since the development of ontologies is motivated by scenarios related to the applications that will use the ontology. Such scenarios describe a set of ontology's requirements that the ontology should satisfy after being formally implemented. A motivating scenario also provides a set of intuitively possible solutions to scenario problems. These solutions give a first idea of the informal intended semantics of the concepts and relations that will later be included in the ontology (Gómez-Pérez et al., 2004).

Gruninger and Fox (Gruninger and Fox, 1995) (TOVE Project) propose identifying intuitively the main scenarios, that is, possible applications in which the ontology will be used. Web pages, documents, common sense and travel agencies are examples of motivating scenarios.

These are some of the terms that were gathered when analysing motivating scenarios: bus, car, levadas, shopping, regional cooking, nature, romantic, activities in family, cultural activities, spa, bar, restaurant, concert, footballmacth, wellness, important events, sightseeing.

FUNCTION COLUMN

The second column concerns the list of activities important to the domain. In ontological terms, this row models the activities that are known as tourism activities. These activities are related to the activities belonging to the complementary offer domain. The tourist does not need to be worried with the accommodation nor the flight since he is already aware where he wants to be. The tourist is concerned with the activities that he can do while staying at a certain place.

...To rent a car...

...To ask for information about a car, boat trip, excursion...

...To schedule a trip about a car, excursion, boat trip...

...To buy a ticket for an excursion, boat trip...

...To plan a boat trip, an excursion...

...To enjoy a day in family...

...To reserve a table for two for a romantic evening...

...To have a cultural visit...

...To dine or to have lunch at a special restaurant by the sea...

...To go to a concert...

...To have a day in a spa...

List 1: An excerpt of the list of some of the activities identified.

This list shows some of the desires customers have when looking for complementary offers in the tourism industry. Since accommodation is already arranged they are only worried about the ways they can sucessfuly spend their time.

MOTIVATION COLUMN

The purpose of the business for which the ontology is going to be developed, its goals and strategies are identified in the third column.

A set of natural language questions called competency questions (Gruninger and Fox, 1995) and purpose questions is elaborated. Competency questions are those to be answered by the ontology once the ontology it is constructed. Purpose questions are used to determine the scope and need of the ontology.

These questions and their answers are both used to extract the main terms and their properties, relations and formal axioms of the ontology.

Purpose Questions/Answers

- What is the domain the ontology will cover?

The ontology will cover all the complementary offers that can be selected by guests staying at Pestana Hotel.

- For what are we going to use the ontology?

The ontology will be used to provide a consensual knowledge model of the tourism

- For what types of questions the information in the ontology should provide answers?

The ontology should provide answer to questions like: What kind of events can I attend to during summer time?, Where can I rent a car? Where can I go for a massage? Where can I go for a romantic dinner? What kind of activities can I do with my family?

- Who will use the ontology?

The Ontology will help to integrate information concerning complementary offer that will be use by employees in charge of helping the guests during their stay.

Competency Questions/ Answers

- What activities can the family enjoy together?

Examples of activities include: to walk, to attend to a Culture Event, an afternoon tea or a boat trip.

- What kind of events can I attend to during the month of February?

You can assist to a Carnival Party.

- Where can I go to have a light meal?

You can go to the Magic Tea House.

Judging from the list of questions, the ontology will include the information on various types of experiences and resources.

Despite the fact that the answers to purpose questions may change during the ontology-design process at any given time, they help limit the scope of the model. On the other hand, the answers to competency questions help to identify requirements.

6.4.1.2 SECOND ROW

In the first row we identified the key concepts and the activities that are associated to this domain. We also limited the scope of the model and identified requirements with the competency and purpose questions.

Helena Sofia and João Martins (Pinto and Martins, 2004) call this stage specification (that is also the first stage in the ontology development life cycle). According to them, the second stage, conceptualization, is the stage that describes, in a conceptual model, the ontology to be built, so that it meets the specification made in the previous step. The conceptual model of an ontology consists of concepts in the domain (identified in the Data column) and relationships among those concepts (identified in the Function column). If we compare the ZOF to the ontology development life cycle described by Helena Sofia and João Martins (Pinto and Martins, 2004), we can say that the first and second row of the ZOF are equivalent to the specification and conceptualization stage identified in the ontology development life cycle.

According to Gruber 5 (Gruber, 1993) principles for designing ontologies to be used in knowledge sharing, conceptualization should be specified at the knowledge level without depending on a particular symbol level encoding. These two rows are written in a Natural

Language and will become more and more formal in the other rows. In ontological terms, this row models the nature of the ontology, including its structure, functions, organization and so forth. This row describes business functions as perceived by people performing them. It is the conceptual perspective.

DATA COLUMN

A controlled vocabulary is built in the data column. A controlled vocabulary is a precise, unambiguous, non-redundant list of concepts (Cardoso, 2005). A definition is given for each concept identified previously.

Term	Definition	Term	Definition
Bus	a vehicle carrying many passengers; used for public transport;	Levadas	A levada (Portuguese for "led") is an irrigation channel or aqueduct on the island of Madeira in the Atlantic Ocean
Car	a motor vehicle with four wheels; usually propelled by an internal combustion engine	Shopping	Shopping is the purchase of goods and services from retailers. In some contexts it is considered a leisure activity as well as an economic one.
regional cooking	is the act of applying heat to food in order to prepare it for ingestion. It encompasses a vast range of regional methods, tools and combinations of ingredients to alter the flavor or digestibility of food.	Nature	the natural physical world including plants and animals and landscapes etc
romantic	activities involving two characters falling or being in love	spa,	Used to describe the facilities offered at venues with not just swimming pool, but usually sauna, steam, Jacuzzi etc
activities in family	an association of people who share common beliefs or activities	bar	a counter where you can obtain food or drink
cultural activities	Activities of or relating to the	restaurant	is an establishment that

	arts and manners that a group favors.		serves prepared food and beverages to order, to be consumed on the premises
Concert	is a live performance, usually of music, before an audience	wellness	The state of feeling well in the body, mind and spirit.
Footballmacth	Is a formal contest in which two teams compete	important events	Something important that happens at a given place and time

Table 10: An excerpt of the controlled vocabulary

The controlled vocabulary produced can be used as a list of concepts that provide a standard vocabulary of words of the domain in study. These concepts have been carefully selected in order to avoid having multiple terms for the same subject.

FUNCTION COLUMN

From the list of activities produced previously it is possible to acknowledge knowledge concerning the way how the concepts are related. The properties and type of properties of the concepts identified are described. For instance, a bus has a departure place and an arrival place. These properties belong to the term bus. Another example is the concept *important event*. Every event has always a time and place. So, time and place are properities of the concept event. It is also possible to conclude that there are different types of events. If we look for events on the web they appear categorized. Instead of having a concept that gathers all the events it is better to categorize them. This categorization can be done according to their description. This property, description, helps to define the event and to classify it. If we are going for a concert, its description should say what kind of music we will hear. Thus, the concert is a music event. Music events can be classified as cultural events since are related to the arts and manners of a group, community, or to the shared knowledge and values of a society.

MOTIVATION COLUMN

In the previous row, competency and purpose questions were answered. Competency questions help to identify the scope of the ontology. The scope is the complementary offer in the e-Tourism domain. This domain is a gigantic business field. There are rules that "govern" each of the different fields of this domain. This column main goal is to identify, formalize and implement the rules that "govern" this domain.

In this row, it is important to analyse the answers to the competency questions and determine some of the rules that govern this domain. These rules can be gathered in a document so that it can be easily updated and accessed.

For instance, according to answer to the first competency question it is possible to state that there are rules to classify activities. Four types of activities are suggested: walk, to attend to a concert, to have an afternoon tea and to have a boat trip. All of these activities can be enjoyed in family. To reserve a table for two is not a family activity since it only involves two persons. A table for two should be classified as a Romantic Activity.

Rule 1: An activity is a family activity if and only if can be done in family.

A family is a group of more than two persons that work as a social unit and that can be related by birth, marriage, or adoption and that have once reside together.

Another rule example is the one concerned with the events that happen in February. The property time/date is important here. This property will help to identify the events that happen at a certain time.

Rule 2: An event is only an event if and only if it has a date and time.

This is one of the many rules that we are able to determine when analyzing the answers to the competency questions.

6.4.1.3 THIRD ROW

The third row is the logical row. In this row it is decided what is technically and physically achievable. This row introduces details of how the terms may be implemented in an operational environment. A Taxonomy is built and relationships between concepts become clear. The goal of this task is to describe in detail all the relationships included in the taxonomy and to produce a relationship table. A rule table is built in the third column. Methontology proposes to include the following information in the rule table: name, Natural Language description, the expression that formally describes the rule, the concepts, attributes and relations to which the rule refers.

DATA COLUMN

Taxonomy is a subject-based classification that arranges the terms in a controlled vocabulary into a hierarchy. The concepts that compose the controlled vocabulary are arranged into taxonomy (Cardoso, 2005). A concept is described by making explicit its relationships with other concepts. The list that composes the controlled vocabulary is enhanced with parent-child relationships (generalization). The generalization relationship is a taxonomic relationship between a more general description and a more specific description that builds on it and extends it. The more specific description is fully consistent with the more general one (it has all properties, members, and relationships) and may contain additional information. The more general description is called the parent; an element in the transitive closure is an ancestor. The more specific description is called the child; an element in the transitive closure is an ancestor (Rumbaugh et al., 1998).

New terms emerge when we try to organize the terms in a taxonomy. Resource and Experience are the two main concepts of this ontology that group the terms identified earlier. A Resource is

a natural or human wealth which can be used to satisfy human needs. An Experience is composed by one or more resources that when combined with each other allows something significant to happen, that is, that it will be highlighted from the ordinary events and that is remembered like that.

Figure 22: Taxonomy of the complementary ontology

Experience and Resource are superclasses. Experience has four subclasses and Resource has three subclasses. Each of the Resource subclass has two or more subclasses and so on. The term resource gathers the "goods" that are concrete, that are real-world objects while experience gathers what people are sometimes looking for when accommodated in a hotel for vacancies. This taxonomy was presented to Expedita experts and was reformulated until all the persons involve were agreed.

FUNCTION COLUMN

Once we have defined the classes, it is necessary to describe the internal structure of concepts. There are several types of object properties:1) "intrinsic" properties, 2) "extrinsic" properties, 3) parts, if the object is structured; these can be both physical and abstract "parts" and 4) relationships to other individuals; these are relationships between individual members of the class and other items.

We decided that we would pay more attention to the class experience than to the resource class since other ontologies have already described some of the resources that we identified. The novelty is the class experience. This class can be understood like a category of activities pre-created or a class day combines different types of resources so that it fulfills needs of a customer.

This class has the following properties: haslocation, hastime, hasprice and hasticket.

MOTIVATION COLUMN

Rules identified previously need to be updated and described in a more detailed way. Concepts and relationships used in the rule

In the previous row we identified a rule that says that an activity is a family activity if and only if can be done in family. This rule should be updated to the following one: *an experience is an in family experience if and only if it can be done in family.*

n this row, all the business rules captured are organized in a table.

For each rule definition, Methontology proposes to include the following information: name, Natural Language description, the expression that formally describes the rule, the concepts, attributes and relations to which the rule refers, and variables used in the expression. Methontology proposes to specify rules expression using the template *if <conditions> then <consequent>.* The left-hand side of the rule consists of conjunctions of atoms, while the right-hand side is a single atom.

An example of this type of rule is the following one:

Name:Ferry Rule

Description: Every ferry that departs from Madeira is arranged by the company Lobo Marinho

Expression: $\forall(x,y,z)\ MadeiraLocation(x)\ and\ ferry(y)\ and\ departureplace((y,x))then\ companyname\ (y,"Lobo\ Marinho")$

Concepts:Ferry, MadeiraLocation

Attributes:company name

Relations: departureplace

Variables: x, y

6.4.1.4 FOURTH ROW

The fourth row concerns the physical perspective. It describes how technology may be used to address the information processing needs identified previously

DATA COLUMN

A Thesaurus is a networked collection of controlled vocabulary terms with conceptual relationships between terms. According to the National Information Standards Organization

(NISO 2005), there are four types of relationships that are used in Thesaurus: equivalence, homographic, hierarchical, and associative.

An equivalence relation says that a term t_1 has the same or nearly the same meaning as t_2. Two terms t_1 and t_2 are called homographic if term t_1 is spelled the same way as term t_2, but has a different meaning. The hierarchical relationship has already been considered in the taxonomy. The associative relationship is used to link terms that are closely related in meaning but not hierarchical. An example of an associative relationship can be as simple as "is related to" as in terms t_1 "is related to" term t_2.

In order to taxonomy became a thesaurus the following three types of relationships must be considered: equivalence, homographic and associative.

There is not any homographic relationship. It is possible to establish the following associative relationships:

A cultural experience is associated to the cultural events (entertainment resources).

A romantic experience is associated to the gastronomy (entertainment resources) or to the leisure activities (entertainment resources) or to the healthandbeauty (entertainment resources).

A wellness experience is associated to the healthandbeauty (entertainment resources).

A cultural experience is associated to the cultural events (events resource).

These relationships will be described using the asserted conditions editor.

For instance, a cultural experience is associated to the cultural events is described as followed.

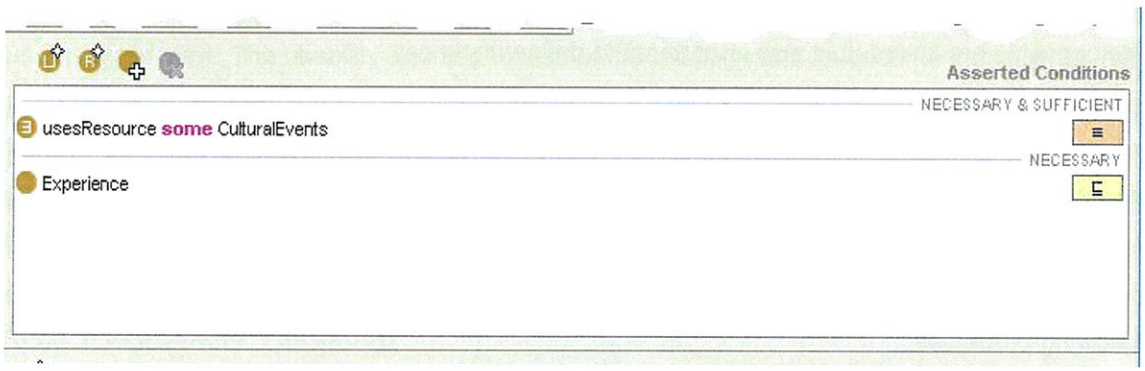

Figure 23: Asserted Conditions editor expressing an associated relationship.

Concerning the equivalence relationships it is possible to establish the following ones:

Wellness is equivalent to HealthandBeauty.

Cultural is equivalente to cultural events..

FUNCTION COLUMN

This row is concerned with the physical perspective. An ontology tool and a language should be selected so that the knowledge acquired becames clear and organized for being processed by computers so that they can make inferences from it.

MOTIVATION COLUMN

Rules identified and detailed previously are formalized using axioms so that they can be implemented in an ontology development tool later. In this row the formalization of rules allows to know and validate how terms can be combined into valid statement and how sanctioned inferences can be made.

6.4.1.5 FIFTH ROW

This row is concerned with the out-of-context. It is almost always worth considering what someone else has done and checking if we can refine and extend existing sources for our particular domain.

DATA COLUMN

Concerning the Data Column, it is very useful to study other ontologies and use the knowledge obtained to validate our ontology. We studied the travel ontology and notice that this ontology overviews some of the resources concept. The travel ontology considers the following concepts: hotel, restaurant and airtravel (Gordon et al., 2005)

FUNCTION COLUMN

In the function column we should studied the ontologies that are related to this domain and how the concepts are linked. It is also importante to define instances.

This cell main goal is to define instances of classes in the hierarchy. Defining an individual instance of a class requires: 1) choosing a class, 2) creating an individual instance of than class, and 3) filing in the properties values. For example, we can create an individual instance a spa day to represent a specific experience, a wellness experience. A day spa is an instance of the class wellness experience representing all the wellness experiences. This instance has the following properties:

Name: A spa day (instance of HealthandBeauty)

Location: Funchal- Choupana Hills (instance of HealthandBeauty)

BeginTime:11:00

EndTime:18:00

Ticket_number: 1111 (instance of Ticket)

Price: 200€

MOTIVATION COLUMN

Like in the Data column, it is important to analyse the rules that govern the other tourism domain and see if any of rules make sense in our ontology.

6.4.1.6 SIXTH ROW

The sixth row concerns the Functioning Enterprise. In this row the ontology is implemented and made part of an organization. This row is also concerned with the evaluation, that is, judge the quality of the ontology.

DATA COLUMN

The end product is a modeled ontology. It is necessary to organize the documentation associated to the terms represented in the ontology. This is an important activity since not only improves its clarity, but also facilitates maintenance, use and reuse.

FUNCTION COLUMN

An implemented ontology is integrated and a report of what was done, how it was done and why it was done should be done. This report will help maintenance (update and correct the implemented ontology).

MOTIVATION COLUMN

The end product is a rule document with all the rules that govern the domain.

The modeled ontology should be use to assessment. This ontology should be judge from the user point of view. The usability and usefulness of the ontology and its documentation when (re) used or shared in application should be analysed. A technical evaluation should be done (judge the ontology and documentation against the framwork). Verification will guarantee its correctness according to the accepted understanding about the domain. Validation guarantees that it corresponds to what it is supposed to, according to the specification requirements.

6.4.2 Results and Validation

The interest of building ontologies using a framework like the one presented is to reduce the knowledge acquisition time during the ontology development. ZOF works as a guideline for the ontology development. This new framework documents all of the steps taken. Therefore, it is possible to easily evaluate/validate the ontology and obtain the documentation needed. There is also an informal and a formal phase. The informal phase starts in the first row and the formal starts in the third row. The column concerning Data is the column concerned with the data that will became information. The information will evolve to knowledge. The column concerning Function is the column where the behaviour of the system is formalized. Relationship between the concepts and properties are described. The third column, Motivation column is related to the business goals and strategis. It is also related to the rules that "govern" the domain. These

rules restricted and help to describe the domain and the way it "works". In a near future we will try to expand ZOF to the six columns that compose the ZF.

7 COMPLEMENTARY OFFER ONTOLOGY

Ce-TO is the name we gave to the ontology built using the ZOF. It stands for Complementary e-Tourism Offer. This ontology is the solution for data integration between different web applications since it offers a share, organized, and common understanding of the data, allowing better integration, communication, and interoperability of inter and intra organizacional tourism information systems.

7.1 OVERVIEW

Ce-TO has three main concepts: Experience, Resources and Ticket. It is structured this way since we believe that there are two types of customers. The customer that knows what he wants to do and a customer that wants to find out what can he do. The resource concept gathers all the resources available while the experience concept describes the type of experience that a customer can have. If he is looking for a car rental then he will search information concerning the resource concept. If he does not know what he can do, he will search by categories.

Most tourists select a flight and accommodation when they organize their vacancies. They do not plan the activties that they will be doing during their stay. Therefore, this is ontology will be use to promote the complementary offer. It will be available on an on-line tool to be used internally, by the internet site of Group Pestana and by customers of this Hotel.

In the contextual row, information about the domain in study is obtained.

When most of the information has been acquired, we have a lot of unstructured information that must be organized. Conceptualization organizes and structures the acquired information using external representations that are independent of the implementation languages and environments. Specifically, this phase organizes and converts an informally perceived view of a domain into a semiformal specification using a set of intermediate representations that the domain expert and ontologists can understand (controlled vocabulary anc concept properties).

After obtaining a sizable number of terms, we built a concept classification tree using relations such as subclass-of. A class C is a subclass of parent class P if and only if every instance of C is also an instance of P.

These intermediate representations fill the gap between how people think about a domain and the language in which ontologies are formalized.

7.2 ONTOLOGY DESCRIPTION

This ontology is called Ce-TO and stands for Complementary e-Tourism Offer. Ce-TO can be divided into two parts: one that describes concrete, real-world objects and "relationships between them" (Resource), and one that defines more abstract business aspects of the world of complementary offer (Experience). Following this delineation, the most general view of the proposed ontology is depicted in Figure 24.

Figure 24: Ce-TO graph overview

7.2.1 Concrete Part

The first and most general notion in this part is the class Resource with its properties (name, company name, schedule, price, etc). This class has three subclasses: Gastronomy, Entertainment and Transportation.

Gastronomy has the follwing subclasses: Bar, Snack-bar, Restaurant and Others. A bar is a place where a tourist can have a drink. A snack-bar is place where a tourist can have a light meal, like a toast and a milk shake. A restaurant is a place where the tourist can have a meal using forks. Other gathers all the places that are not bars, snack-bars or restaurant. For example, fast food and food shops.

The class entertainement has three subclasses: Events, LeisureActivities and HealthandBeauty. SportEvents and CulturalEvents are the subclasses of the class Events. We believe the class events can be expanded. We did it expanded since this ontology is a simple example of an ontology built with ZOF. An example of a cultural event is a concert and an example of a sportevent is a football match. In order to obtain this information the system has to look for information concerning events happening at a certain place and time and pay attention to the description made of the event. The description will help classify the event.

The LeisureActivities class has four subclasses. They are: SightSeeing (very popular in Madeira), Shopping, Gym and Levadas (very popular in Madeira). These activities can all be done in family and are activties that are usually request by customers staying at Madeira.

The HealthandBeauty class is a new business area that has gained a lot of fans. This class has two subclasses: Spa and Hairdresser.

The transportation class handles means of travel. We have distinguished four of them: LandTransportation, AirTransportation, SeaTransportation and EcologicalTransportation. This notation is similar to the one used in the real-worl. The ecological transportation is concerned with the means of travel that are nature friend (for example, bike, horse).

Figure 25: Resource Concept (Class) with its subclasses

Figure 25 shows all the subclasses of the Resource concept. This concept can be defined as all the natural or human wealth which can be used to satisfy human needs. There are companies that provide these resources. For instance, in Madeira, the company that does the trip to Porto Santo by ferry is Lobo Marinho. If a tourist queries the system about a ferry trip to Porto Santo, the application ought to look for information about it.

7.2.2 Abstract Part

The abstract part is related to the Experience concept. We can not measure and experience. It is not possible to quantify it. It is only possible to qualify it.

An Experience is composed by one or more resources that when combined allows something significant, that is, that it will be "destacado" from the ordinary "acontecimentos" and that is remember like that.

The experience concept is related to resource concept since they combine the different resources available. The class experience has four subclasses: InFamily, Wellness, Romantic and Cultural as figure 26 shows

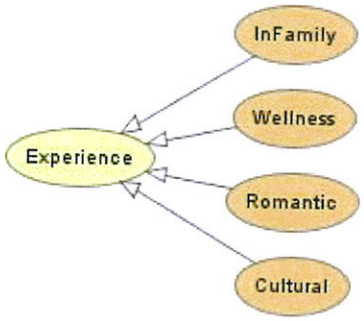

Figure 26: Experience concept overview.

A InFamily experience can combine all the events of the class resources and all the leisure activties. It is possible, to go shopping InFamily or to walk in a Levada.

A Wellness experience can be a dinner at a really fine restaurant or a cup of tea at the Magic Tea House. It can also mean a spa day or to have a hair cut.

A Romantic experience can be a boat trip or a candle dinner.

A Cultural experience experience can be the combination of a visit to a sightseeing spot and a concert at the Theater.

7.2.3 Ticket Concept

The ticket concept gathers information concerning the resources. It is also the proof that a resource has been selected and paid. The ticket can identify the customer and the resource. A ticket is a commercial document showing that the holder is entitled to something.

8 CONCLUSIONS AND FUTURE WORK

Tourism is one of the sectors of the World economy with the best outlook.The increase in time for leisure activities and its social importance promotes the expansion of the Tourism Industry. The rapid growth of the Internet and the continual adoption of innovative technology have led to serious changes in the travel industry.

The World Tourism Organization (WTO, 2005) predicts that by 2020 tourist arrivals around the world would increase over 200%. Tourism has become a highly competitive business for tourism destination over the world. Competitive advantage is no longer natural, but increasingly driven by science, information technology and innovation (Cardoso, 2005).

Semantic Web has a huge potential for e-Tourism since it provides: 1) semantically enriched information searching, 2) integration and interoperability, 3) personalized and context-aware recommendations and 4) internationalization. Its vision is to have web data linked so that it can also be used by machines for automation, integration and reuse across various applications.

When looking at information systems and the web from a tourist perspective it is possible to state that there are two types of information. One type of information is well structured in repositories and allows access via web, but is normally isolated and rarely includes detailed information. The other type of information is the web as a whole with its many small, detailed pieces of information, about music festivals, leisure activities, etc.

The increased celerity associated with the evolution of tourism calls for technical innovation to generate superior consumer services such as market overview or price comparison.

The SEED project mainly focuses on narrowing the gap between those two types of information. The objective is to connect the isolated pieces of information in order to assist the user finding and understanding the information sources and in order to allow individualized use of tourism offers. To achieve this purpose technology of the SW was employed.

There are several methodologies that support the ontology development. It is not possible to evaluate which methodology is best since it depends of project goals. Some of the methodologies presented had drawbacks. The main drawbacks were: 1) lack of conceptualization process before implementing the ontology, 2) some management and support activities are missing, 3) there are typically separated stages to produce first the informal

description of the ontology, and then its formal embodiement in a ontology language and 4) none of the methodologies gives an advice on how to identify ontological concepts. Therefore, we decided to build a new methodology. This new methodology is based on the Zachman Framework (ZF) and tries to address the drawbacks identified. ZF has the following advantages: 1) it is a proven methodology in software engineering since it has widely been used with success, 2) it includes a step-by-step development plan and work breakdown structure because it is organized in rows and columns. It is possible to start the development in any of the cells, 3) it is a tool for sharing and reusing enterprise information among people since each row is a perspective and each column an aspect that is easily understood, 4) it explicitly shows that there are many views that need to be addressed when developing a product, 5) it provides a reminder of the issues that need to be considered namely contextual, conceptual, logical , physical and out-of-context features, 6) Different perspectives mean different views that mean different ways of looking, using and treating information, 7)It has a comprehensive taxonomy since you do not have to be an engineer to understand it and use it and 8) it is easy to change and maintain since it is organized in rows and columns.

ZOF has two perspectives related to the conceptualization stage (1st drawback). This stage happens before implementing the ontology. The 2nd drawback is addressed by adopting the managementa and support acitivies of the Methontology methodology. Initially an informal description of the ontology is built. This informal description is related to the formal description and is embodied in a ontology language (3rd drawback). ZOF gives an advice on how to identify ontological concepts (analyse motivating scenarions and the activities important to the domain).

A study of the ontology tools available was done. This study was very helpful for the selection of the ontology language and tool. It also showed that only two of the tools studied support a specific methodology (Doe and WebODE).

Several tourism ontologies were considered for reuse, before we built our own ontology. In e-tourism different ontologies have been developed for different areas. However, none of them meets our needs to describe complementary offers. We identified terms relevant to the domain and then expanded it to ontology by adding relations and properties. Finally, we identified three main categories that are the most important to the domain. The ontology main focus is the description of complementary offers available (resources and experiences). The intention of a user to query a tourism portal is to find relevant resources of complementary offer, for example, transportation or to find a "package" that gives the possibility to experience something. Many tourists prefer to select a type of experience instead of combining several resources (transportation and cultural event).

Therefore, we defined a category for concrete real world objects (resources) and a category for experiences (combination of resources). This domain is still an extremely challenging work for the future.

Ce-TO is a complementary e-tourism ontology that will be available at an "e-kiosk". This "e-kiosk" main goal is provide information concerning the complementary offer available. This ontology is a small ontology but we would like to extend it a near future.

9 REFERENCES

Agenticities (2007), http://www.agentcities.org (last accessed in the 7of December of 2007)

Allen J, (1984), Towards a general theory of action and time. Artificial Intelligence 23:123-154.

Allen J F, (1983) Maintaining Knowledge About Temporal Intervals. Communications of the ACM, 26(11):832–843, 1983.

Ankolenkar A, Burstein M, Hobbs J, Lassila O, Martin D, McIlraith S, Narayanan S, Paolucci M, Payne T R, Sycara K and Zeng H. (2001) *DAML-S: A Semantic Markup Language For Web Services.* In: Semantic Web Working Symposium (SWWS).

Antoniu G and van Harmelen F (2004), A Semantic Web Primer. MIT Press

Arkin A (2001), Business Process Modeling Language (BPML), Working Draft 0.4,

Aspiréz JC, Corcho O, Fernández-López, M, Gómez-Pérez A (2003), WebODE in a nustshell. AI Magazine.

Auer S (2005), pOWL – A Web Based Platform for Collaborative Semantic Web Development. In Proceedings of the Workshop on Scripting for the Semantic Web, Heraklion, Greece, May 30, 2005.

Baader F, McGuiness D, Nardi D, Patel-Schneider P (2003), The Description Logic Handbook: Theory, implementation and applications. Cambridge University Press, Cambridge, United Kingdom

Bachimont B, Isaac A., and Troncy R (2002), Semantic Commitment for Designing Ontologies: A Proposal. In Asuncion Gomez-Pérez and V. Richard Benjamins, editors, 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW'2002, volume LNAI 2473, pages 114-121, Sigüenza, Spain, October, 1-4 2002. Springer Verlag.

Bechhofer S, Horrocks I, Goble C, and Stevens R (2001), OilEd: a reason-able ontology editor for the Semantic Web, Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence.

Benjamin P, Menzel C, Mayer R, Fillion F, Futrell M, De Witte P, Lingineni M, (1994) , IDEF5 Method Report, Prepared For Armstrong Laboratory, ALIHRGA, Wrght-Pattersin Air Force Base, Ohio 45433, Revision Date: September, 21, 1994.

Bernaras A, Laresgoiti I, Corera J (1996) Building and Reusing ontologies for electrical network applications. In: Wahlster W (ed) European Conferences on Artificial Intelligence (ECAI'96). Budapest, Hungary. John Wiley and Sons, Chicester, United Kingdom, pp 298-302

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

Berners-Lee T, Miller E, (2002) The Semantic Web lifts off.

Berners-Lee T, Hendler J and Lassila, O. (2001) The Semantic Web. Scientific American. May 2001.

Berners-Lee T, (2007) Interview session between eWEEK Chief Technology Analyst Jim Rapoza and Tim Berners-Lee, creator of the Web and head of the World Wide Web Consortium.

Berry, T, (2004) A Standard Business Plan Outline (http://www.bplans.com/dp/)

Blásquez M, Fernández-López M, Garcia-Pinar JM, Gómez-Pérez A (1998) Building Ontologies at the Knowledge Level using the Ontology Design Environment. In: Gaines BR, Musen MA (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98). Banff, Canada.

BPMI.org, 2001.

Broekstra J, Kampman A., and Van Harmelen F, (2002), Sesame: A generic architecture for storing and querying RDF and RDFSchema. Horrocks, I., Hendler, J., eds. In Horrocks, I., Hendler, J., eds.: Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002), Sardinia, Italy, LNCS, Springer, Vol. 2342, pp. 54–68.

Business Process Project Team (2001) ebXML Business Process Specification Schema (BPSS), Version 1.01, ebXML, 2001.

Bussler C, (2001) The Role of B2B Protocols in Inter-Enterprise Process Execution. In

Cardoso J (2006) "Developing an OWL Ontology for E-Tourism", in Semantic Web Services, Processes and Applications. 2006, Springer, ISBN: 0-38730239-5

Cardoso J (2005) Semantic Web: Technologies and Applications. Unpublished Paper, Universidade da Madeira, Portugal.

Cardoso J, (2005) E-Tourism Creating Dynamic Packages using Semantic Web Processes.

Cardoso J, Fernandes J, (2005) SEED Report, for Semantic Web FactBook, AIS SIGSEMIS.

Cardoso J, Sheth A, (2006),Semantic Web Services, Processes and ApplicationsSemantic Web Services, Processes and Applications, Developing an OWL Ontology for E-tourism chapter 10.

Chandrasekaran B, Jonhson TR, Benjamin VR (1999) Ontologies: what are they? Why do we need them? IEEE Intelligent Systems & their applications, Special Issue on Ontologies

Chandrasekaran B, Josephson JR, Benjamins VR ,(1999), What are ontologies, and why do we need them? IEEE Intelligent Systems 14, 1 (Jan./Feb. 1999), 20–26.

Daml (2006) http://www.daml.org or http://www.daml.org/ontologies/178

Daniel D (2007), Five Steps to Managing Innovation, http://www.cio.com/article/125700/Five_Steps_to_Managing_Innovation

Denny M, (2004) Ontology Tools Revisited, http://www.xml.com/pub/a/2004/07/14/onto.html?page=2

Daniel D, (2007), The rising importance of E.A., March 2007

Hay DC, (1997), Essential Strategies, Inc. The Zachman Framework: An Introduction

De Hoog, R (1998), Methodolgies for building Knowledge Based Systems: Achievments and Prospects, In: Leibowitz J (ed) Handbook of Expert Systems. CRC Press Chapter 1, Boca Raton, Florida

Dean, M, Schreiber, G (2003) OWL Web Ontology Language Reference. W3C.

Decker, S., Erdmann, M., Fensel, D., and Studer, R. (1999) Ontobroker: Ontology-based access to distributed and semi-structured information. In: R. Meersman et al. (eds.), Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, Kluwer Academic Publishers, Boston, 1999. pp. 351–369.

Deri (2004) http://www.deri.at/research/projects/e-tourism/2004/d10/v0.1/20040719

Devedzic, V. , (2002) Understanding Ontological Engineering, Communications of the ACM archive, Volume 45 , Issue 4 (April 2002) ISSN:0001-0782, pp: 136 - 144.

Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M. C. A., et al. (2002). Golden-Bullet in a Nutshell, FLAIRS-2002: *The 15th International FLAIRS Conference* (pp. 403-407).

DOE (2006), The Differential Ontology Editor, http://opales.ina.fr/public/index.html

Enderton HB (1972), A Mathematical Introduction to Logic, San Diego, CA: Academic Press.

ERCIM http://www.ercim.org/publication/Ercim_News/enw51/berners-lee.html

E-Tourism Working Group (2004), http://e-tourism.deri.at/, 2004

Évora (2007) http://www.visitevora.pt/pt/conteudos/onde_o_alentejo_acontece/cultura/MigObj_Viver_a_historia.htm

EWeek http://www.eweek.com/article2/0,1895,2143643,00.asp

Expedita (2007), http://www.expedita.com/expedita/index.php?option=com_content&task=view&id=25&Itemid=127

Farquhar, A., Fickas, F., and Rice, J. (1995), The Ontolingua Server: a tool for collaborative ontology construction , Proceedings of the 10yh Banff Knowledge Acquisiton for Knowledge based system workshop (KAW 95).

Fensel, D, van Harmelen F, Horrocks I, McGuiness DL, Patel Schneider PF (2001) OIL: An ontology infrastructure for the Semantic Web. IEEE Intelligent Systems & their applications 16 (2):38-44

Fernández- López, M., (1999) Overview of Methodologies for building Ontologies, Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, August 2, 1999.

Fernández-López M, Gómez-Péres A, Juristo N (1997), Methontology: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontolgical Engineering of AAAI. Stanford University, California, pp 33-40.

FIPA (2007), http://www.fipa.org/

Fodor, O., Werthner, H. (2004). Harvesting lightweight Ontologies out of legacy XML sources. In Proc. of the First International Conference on Knowledge Engineering and Decision Support (ICKEDS'04). Porto, Portugal.

Fodor, O. & Werthner, H. (2005) Harmonise – a Step towards an Interoperable e-Tourism Marketplace. *International Journal of Electronic Commerce 9/2.*

Fowler, M. and Scott, K.(1997) UML Distilled: Applying the Standard Object Modelling Language.Addison-Wesley, Reading, MA, 1997.

Fridman-Noy, N., Hafner, C.D. (1997) The state of the art in ontology design: a survey and comparative review. AI Magazine (Fall 1997), 53–74.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995.

Goméz, Mariano Fernandez, Asuncion Gomez-Perez, and Natalia Juristo (2007) METHONTOLOGY: from Ontological Art Towards Ontological Engineering. In Proceedings of the AAAI 97 Spring Symposium Series on Ontological Engineering, pages 33–40. The AAAI Press, 1997.

Gómez-Pérez A, Fernández-López M, Corcho, O (2004) Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)

Gómez-Pérez A, Fernández-López M, de Vicente A (1996) Towards a method to conceptualize domain ontologies. In: van der Vet P (ed) ECAI'96 Workshop on Ontological Engineering. Budapest, Hungary, pp 41-52

Gómez-Pérez, A. , Fernández-López, M., & Corcho, O. (2004), Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, pp. 293-362, 2004.

Gómez-Pérez, A., Fernández-López, M., Corcho, O., and Aspiréz, J. (2001) WebODE: a sacalable ontological engineering workbench, First International Conference on Knowledge Capture (K-CAP 2001) Canadá.

Gómez-Pérez, A., Fernández-López, M., Corcho, O., (2004) Ontological Engineering, ISBN 1-85233-551-3 Springer-Verlag London Limited 2004, 2nd printing, 2004, pp 107-130.

Gordon, M., Kowalski, A, Paprzycki, M, Pelech, T, Szymczak, M, Wasowic, T, (2005) Ontologies in a travel support system. In: D. J. Bem et. al. (eds.) *Internet 2005*, Technical University of Wroclaw Press, 2005, 285-300,

Grau, B. C. (2004), A Possible Simplification of the Semantic Web Architecture In Proceedings of the Thirteenth International World Wide Web Conference (WWW-2004)

Gruber, T (2005), Ontology of folsksonomy: A mash-up of apples and oranges, http://tomgruber.org/writing/ontology-of-folksonomy.htm

Gruber, TR, (1993) A translation approach to portable ontology specification

Gruninger M, Fox, M (1995) Methodology for the design and evaluation of ontologies In Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6.1-6.10

Gruninger, M., and Fox, M.S. (1995), "Methodology for the Design and Evaluation of Ontologies", Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal.

Guarino N, Giarretta P (1995), Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: Mars N (ed) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS'95). University of Twente, Enschede, The Netherlands.

Hay, D., (1997) Zachman Framework Introduction, www.tdan.com/i001fe01.htm

Horrocks I, (2002), DAML+OIL: A reason-able Web Ontology Language, http://web.comlab.ox.ac.uk/oucl/work/ian.horrocks/Publications/download/2002/edbt02.pdf

Horrocks I, Fensel D, Harmelen F, Decker S, Erdmann M, Klein M (2000), OIL in a Nutshell. In : Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp1-16.

Horrocks I, van Harmelen F (eds) (2001) Reference Description of the DAML+OIL (March 2001) Ontology Markup Language. Technical report. http://www.daml.org/2001/03/reference.html.

http://standards.ieee.org/reading/ieee/std_public/description/busarch/1178-1990_desc.html

http://www.idef.com/IDEF5.html

http://www.w3.org/TR/1999/RECrdf-syntax-19990222

IEEE (1990) http://standards.ieee.org/reading/ieee/std_public/description/busarch/1178-1990_desc.html

IEEE (2005) Standard Upper Ontology. http://suo.ieee.org (last accessed Dec 2005).

IEEE Std 1178-1990 IEEE Standard for the Scheme Programming Language -Description,

IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29, 3 (Aug. 1999), 422–439.

Jasper, R, Uschold M (1999) A Framework for Understanding and Classifying Ontology Applications.

Jones D., Bench-Capon, T., Visser, Pepijn, (2000) Methodologies for ontology development.

Jones, D., Bench-Capon, T. and Visser, P., (2000) Methodologies for ontology construction, Department of Computer Science, University of Liverpool, Liverpool, U.K., L69 7ZF

Kalyanpur, A., Parsia, B., and Hendler, J. (2005), A tool for working with web ontologies, in:In Proceedings of the International Journal on Semantic Web and Information Systems, Vol. 1, No. 1, Jan - March, 2005.

Klein, M, Fensel, D, (2001) Ontology Versioning on the Semantic Web ,SWWS Stanford, July 30, 2001 Vrije Universiteit Amsterdam Goal

Kuno, H. Lemon, M. Pogossiants, G. Sharma, S Williams S.(2001): Web Services Conversation

Laera, L., Tamma, V. (2005), The Hitchhhiker´s Guide to Ontology Editors. This paper is based on and updates the extensive evaluation of ontology editors produced by the Ontoweb project in the Deliverable 1.3, "A survey on ontology tools" and can be found at http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip

Lake, D., (2001) American Go Online for Travel Information, in CNN.

Lassila, O., and Swick, R.R. (1999): Resource description framework (RDF) Model and syntax specification. Recommendation, W3C, February 1999.

Lasssila O, Swick, R (1999) Resource Description (RDF) Model and Syntax Specification. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax/

Lenat DB, Guha, RV (1990), Building Large knowledge-basesd Systems: Representation and Inference in the Cyc Project. Addison-Wesley, Boston, Massachusetts.

Lenat, D.B. (1995) CYC: A large-scale investment in knowledge infrastructure. Commun. ACM 38, 11 (Nov. 1995), 33–38.

Leymann F. (2001) Web Service Flow Language (WSFL 1.0), May 2001. http:// www-4.ibm.com/ software/solutions/webservices/pdf/WSFL.pdf.

Lopez F (1999), Overview of Methodologies for building ontologies

Lopez, M.F., Gomez-Perez, A., Sierra, J.P., Sierra, A.P. (1999) Building a chemical ontology using methontology and the ontology design environment. IEEE Intelligent Systems 14, 1 (Jan./Feb. 1999), 37–46.

Martins, L A, (2006), Using Grammar Inference Techniques in Ontology Learning (Dissertação)

Mereology http://plato.stanford.edu/entries/mereology/ (last accessed Dec 2005).

Uschold M and King M (1995) Towards a Methodology for Building Ontologies. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing held in conduction with IJCAI 1995, 1995.

Mizoguchi R, Vanwelkenhuysen J, Ikeda M (1995) Task Ontology for reuse of problem solving knowledge. In Mars N (ed) Towards Very large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS'95).

Mizoguchi, R.(1998) A step towards ontological engineering. Proceedings of The 12th National Conference on AI of JSAI (June 1998), 24–31.

Mondeca (2007) http://www.mondeca.com (last acessed in the 7 of December of 2007)

Noy N.F., McGuiness, D. L.,(2001) Ontology Development 101: A Guide to Creating Your First Ontology, Knowledge Systems Laboratory, Stanford University, http://www.ksl.stanford.edu/kst/what-is-an-ontology.html

Noy, N F, Klein, M (2003)Tracking Complex Changes During Ontology Evolution

Noy, N.F., and Musen, M.A. (2003), The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, International Journal of Human-Computer Studies, 2003

Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., and Musen, M.A. (2001), Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems, 16, 60--71.

Oldakowski, R., and Bizer, C. (2004): RAP: RDF API for PHP, To be published in Proceedings of the "1st International Workshop on Interpreted Languages", 2004.

Omelayenko, B. and Siebes, R. (2002) Semantic Web Application Areas.

Omelayenko, D. Fensel, (2001) An Analysis of B2B Catalogue Integration problems: Content and Document Integration, In: Proceedings of the International Conference on Enterprise Information Systems (ICEIS-2001), July 7-10, 2001, p. 945-952

Omelayenko, (2002) RDFT: A Mapping Meta-Ontology for Business Integration, In: Proceedings of the Workshop on Knowledge Transformation for the Semantic for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW-2002), 23 July, 2002, p. 77-84

OpenTravel (2007) http://www.opentravel.org

OWL, Web Ontology Language (OWL). 2004, World Wide Web Consortium (W3C).

Parrish M, T. Ott, C. Lance-Jones, G. Schuetz, A. Schwaeger-Nickolenko and A.P. Monaghan, (2004) Loss of the Sall3 gene leads to palate deficiency, abnormalities in cranial nerves, and perinatal lethality, *Mol. Cell. Biol.* **24** (2004), pp. 7102–7112.

Pinto, H.S., Martins, J.P., (2004) Ontologies: How can they be Built? Knowledge Information Systems 6(4).

Pressman, Roger S. (2001), Software Engineering, A practioner's approach, 5th edition, MacGraw Hill, 2001, ISBN 0073655783, pp20-43.

Protégé http://protege.stanford.edu/publications/ontology_development/ontology101.pdf (last accessed in the 7of December of 2007)

Protégé, (2005) Stanford Medical Informatics

RDFS (2004) Resource Description Framework (RDF) http://www.w3.org/TR/rdf-schema/

Richmond H. Thomason and John F. Horty (1989) Logics for Inheritance Theory. In M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, editors, Proceedings of the 2nd International Workshop on Non-monotonic Reasoning, volume 346 of Lecture Notes in Artificial Intelligence, pages 220–237. Springer-Verlag, 1989.

Roddick J F, (1995), A survey of schema versioning issues for database systems, Information and Software Technology Volume 37, Issue 7, 1995, Pages 383-393

Rumbaugh, J., Jacobson, I, Booch, G. (1998)The Unified Modeling language reference manual

SCIAMhttp://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

SEED Report http://dme.uma.pt/jcardoso/Research/Papers/R&D%20project%20report-SEED.pdf (last accessed in the 7of December of 2007)

Shaw, M. (1995) Patterns for software architectures. In: J.Coplien, D. Schmidt (eds), Pattern Languages of Program Design. Addison-Wesley, Reading, MA (1995), 453–462.

Sheth, A. (2003) Semantic Metadata for Enterprise Information Integration. DM Review. July 2003

Select Business Solution (2006)

Sorensen, R., (1995) A Comparison of Software Development Methodologies Software Technology Support Center, http://www.stsc.hill.af.mil/crosstalk/1995/01/Comparis.app

Sowa, JF (1999), Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing CO., Pacific Grove, California.

Staab S, Schnurr HP, Studer R, Sure Y (2001) Knowledge Process and Ontologies. IEE Intelligent Systems

Stanford University. Knowledge Interchange Format. http://wwwksl.stanford.edu/knowledge-sharing/kif/ (last accessed Dec 2005).

Studer R, Benjamins VR, Fensel D (1998), Knowledge Engineering: Principlies and Methods. IEEE Transactions on Data and Knowledge Engineering 25 (1-2):161-197.

Sure, Y., Angele, J., & Staab, S. (2002), OntoEdit: Guiding Ontology Development by Methodology and Inferencing, In Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics ODBASE 2002, October 28 - November 1, 2002, University of California, Irvine, USA, volume 2519 of LNCS, pp. 1205-1222. 2002.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer R., & Wenke, D. ,(2002), OntoEdit: Collaborative Ontology Development for the Semantic , International Semantic Web Conference, June 9-12 2002, Sardinia, Italy, LNCS, Springer, Vol. 2342, pp. 221–235. 2002.

SW (2001), http://www.w3.org/2001/sw/

Swartout B, Ramesh P, Knight K, Russ T (1997) Toward Distributed Use of Large-Scale Ontologies. In Farquhar A, Gruninger M, Gómez-Pérez A, Uschold M, van der Vet P (eds) AAAI'97 Spring Symposium on Ontological Engineering. Stanford University, California, pp138-148.

Swartout B, Ramesh P, Knight K, Russ T (1997), Toward Distributed Use of Large-Scale Ontologies. In: Fraquhar A, Gruninger M, Gómes-Pérez A, Uschold M, van der Vet P (eds) AAAI'97 Spring Symposium on Ontological Engineering. Stanford University, California, pp138-148

Swartout,W. Tate, A. (1999) Ontologies, Guest Editors' Introduction, IEEE Intelligent Systems 14, 1, Special Issue on Ontologies (Jan./Feb. 1999), 18–19.

Szyperski, C. (1998) Component Software: Beyond Object-Oriented Programming. ACM Press/Addison-Wesley, New York, NY/Reading, MA, 1998.

Thatte, S. (2001) XLANG web services for business process design. http://www.gotdotnet.com/team/xml wsspecs/xlang-c/.

Troncy, R. & Isaac, A. (2002), Semantic Commitment for Designing Ontologies: A Tool Proposal. Poster Session at: 1st International Conference on the Semantic Web, ISWC'2002, Sardinia, Italia, June, 9-12 2002.

UNSPSC (2007) http://www.unspsc.org/

Updegrove A., (2005), The Semantic Web: An Interview with Tim Berners-Lee, http://www.consortiuminfo.org/bulletins/semanticweb.php

URL http://www.mindswap.org/papers/Swoop-Journal.pdf

Uschold M, Grunninger M (1996) Ontologies: Principles, Methods and Applications. Knowledge Engineering Review pag.93-155

Uschold M. (1996), Building Ontologies: Towards A Unified Methodology. In: Watson I (ed) 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems. Cambridge, United Kingdom.

Uschold M., King M(1995) Towards a Methodology for Building Ontologies. In: Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Haring. Montreal, Canada, pp 6.1-6.10

Uschold, M. (1996) "Building Ontologies: Towards A Unified Methodology", Proc. Expert Systems 96, Cambridge, December 16-18th.

Uschold, M. and M. Gruninger (1996), Ontologies: Principles, methods and applications. Knowledge Engineering Review.

Valente, A., Russ, T., MacGregor, R., Swartout, W (1999) Building and (re)using an ontology of air campaign planning, IEEE Intelligent Systems 14, 1 (Jan./Feb. 1999), 27–36.

Varzi, AC, (1996) Parts, wholes, and part-whole relations: The prospects of mereotopology, Data and Knowledge Engineering, Volume 20, Issue 3, November 1996, Pages 259-286 http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6TYX-3VTK31T-2&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=862d53e92997747edefd8c6a705854fb

VLDB (2001). Rome, Italy, September 2001.

W3C (2004) OWL. http://www.w3.org/2004/OWL/, (last accessed Dec 2005).

Whitgift, D.,(1991) Methods and Tools for Software Configuration Management, 1991.

World Tourism (2005) http://www.world-tourism.org

WTO (2005), World Tourism Organization, http://www.unwto.org/index.php

Yager T., (2002) The Future of Application Integration, http://www.infoworld.com/article/02/02/22/020225feintro_1.html

Zachman, J (1987), A Framework for Information Systems.

Zhdanova, A, Keller, U (2005), Proceedings of the 2nd World Enformatika Congress (WEC'05), Choosing an Ontology Language.

Zifa - http://www.zifa.org (last acessed in the 7 of December of 2007)