Ontology mapping for discovery of geographical services

Lydia Marleny Prieto-León
January 2006
Ontology mapping for discovery of geographical services

by

Lydia Marleny Prieto-León
lmprietol@yahoo.com

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfillment of the requirements for the degree of Master of Science in Geoinformatics.

Thesis Assessment Board

Chairman: Professor Dr. Menno-Jan Kraak
First supervisor: Ir. Rob Lemmens
Second supervisor: Dr. Rolf de By
External Examiner: Drs. Marian de Vries

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHEDE, THE NETHERLANDS
Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.
To Laura and Arthur
Acknowledgements

I would like to express my gratitude to the following persons for their invaluable contribution during this study:

To Koert Sijmons, who made the dream of coming to the Netherlands true and was always there as a friend and advisor.

To Ir. Rob Lemmens, who gave me kind guidance and support. Thanks for all the time spent in long meetings giving shape to this research and for sharing with me ideas and possibilities. Also to Ir. Rolf de By for the constructive discussions and his supportive suggestions.

For the priceless support of Gustavo Zárrate, who encouraged me to continue working and hold me up during these difficult times.

To my beautiful daughter Laura, who filled me with love and inspired me every moment as a mother and best friend. And to my cute Arthur, who was given by God as our gift in this Earth.

To my mother Mariela, my sister Doris and my nephew David who encouraged me with love thru the distance.

To the Netherlands Fellowship Programme that supported me financially to pursue my studies.

The support of my institution, Universidad Distrital Francisco Jose de Caldas in Bogotá, Colombia was invaluable for the fulfilment of my objectives.

To my friends and classmates who provided me with constant support and a wonderful environment to work and study.

Enschede, January 2006.

Lydia Marleny Prieto León
Abstract

Organizations have realized that knowledge is the current and real competitive advantage since many years ago. And knowledge has to be exposed and shared to be exchanged and improved with the contributions of communities’ participants. To tackle this necessity of sharing knowledge, not only inside the organization but across organizational boundaries, research centers have been relying on ontologies. Ontology is considered the mean for defining common understanding between communities, and so far, it has been used intensely in some successful projects, such as the Genoma project.

On the other hand the Service Paradigm followed by Service Oriented Architecture (SOA) implementations such as web services, requires a stage for discovering feasible services to solve a request from a consumer. Ontologies offer a good approach to improve the searching process through the use of a common understanding of concepts and restrictions; the purpose is to speed up the discovery process with a search with meaning and with semantics, and not just from a syntactical point of view.

The big new challenge is that with the generalized use of large distributed environments, such as the internet, a proliferation of ontologies has come even in the same domain areas. So how to harmonize, integrate or just navigate between all these similar but different ontologies and realize the improvement to the discovery phase of the Service Paradigm?

Ontology mappings allow the “translation of concepts” between ontologies. A mapping can be considered as the construction of a reference or global ontology that let the free navigation between ontologies.

The goal of the present research is to go deeper in some mapping techniques and intend to build a virtual ontology that represents the mapping between two well-known ontologies and a proposed reference ontology of geo-operations. Some guidelines to build ontological solutions and develop ontological mappings are part of the final results of the exploration.

**Thesis keywords:**
Ontology, mapping, service discovery, taxonomy, geo-operations, SOA.
# Table of Contents

ACKNOWLEDGEMENTS..............................................................................................................5  
ABSTRACT ...................................................................................................................................7  
TABLE OF CONTENTS .................................................................................................................9  
LIST OF FIGURES ........................................................................................................................11  
LIST OF ACRONYMS ....................................................................................................................13  

## 1 INTRODUCTION ....................................................................................................................15  
1.1 MOTIVATION ......................................................................................................................15  
1.2 PROBLEM DEFINITION .....................................................................................................15  
1.3 RESEARCH OBJECTIVE .....................................................................................................16  
1.4 RESEARCH QUESTIONS .....................................................................................................16  
1.5 RELATED WORK ................................................................................................................16  
1.6 THESIS OUTLINE ..............................................................................................................17  

## 2 SERVICE ARCHITECTURE CONCEPTS .............................................................................19  
2.1 INTRODUCTION ................................................................................................................19  
2.2 SERVICE APPROACH ........................................................................................................19  
2.3 LOOSE COUPLING ..............................................................................................................20  
2.4 SERVICE-ORIENTED ARCHITECTURE (SOA) ................................................................21  
2.4.1 SOA Operations ............................................................................................................22  
2.4.2 Benefits of SOA ............................................................................................................23  
2.5 DISCOVERY OF SERVICES ...............................................................................................23  
2.5.1 Metadata ......................................................................................................................23  
2.5.2 UDDI .............................................................................................................................24  
2.6 CURRENT DRAWBACKS OF SERVICE DISCOVERY ..........................................................24  
2.7 CONCLUSION ....................................................................................................................25  

## 3 TAXONOMIES AND ONTOLOGIES OF GEO-OPERATIONS .............................................26  
3.1 INTRODUCTION ................................................................................................................26  
3.2 SEMANTIC WEB ................................................................................................................27  
3.2.1 WHAT IS THE PROBLEM? ..............................................................................................27  
3.2.2 SEMANTIC WEB PROPOSAL .........................................................................................28  
3.2.3 HOW SHOULD THE SEMANTIC WEB WORK? .............................................................29  
3.3 WHAT IS A TAXONOMY? ...................................................................................................30  
3.4 GIS TAXONOMIES ............................................................................................................30  
3.5 GEO-OPERATIONS AND GEO-SERVICES .....................................................................31  
3.6 THE CORE ELEMENT: A TAXONOMY OF GEO-OPERATIONS .......................................32  
3.7 WHAT ARE ONTOLOGIES? ................................................................................................33  
3.8 FROM TAXONOMIES TO ONTOLOGIES ..........................................................................33  
3.9 DESCRIPTION LOGICS .......................................................................................................34  
3.10 RESOURCE DESCRIPTION FRAMEWORK (RDF) ..............................................................35  
3.11 WEB ONTOLOGY LANGUAGE (OWL) BASIS .................................................................35  
3.12 ONTOLOGY DEVELOPMENT ENVIRONMENT .................................................................37
3.13 Development Methods for Ontology Based Solutions ........................................ 38
3.14 Concluding Remarks ......................................................................................... 39

4 Case Study: Location-Based Services ................................................................. 41
4.1 Introduction ......................................................................................................... 41
4.2 OpenGIS Location Services (OpenLS) ............................................................... 42
4.3 ESRI Geo-Processing Commands ........................................................................ 46
4.4 The TRAVELER System ...................................................................................... 47
  4.4.1 Motivating Scenario ....................................................................................... 47
  4.4.2 Using UML to model the system ................................................................... 47
4.5 Building Location-Based Geo-Operation Ontologies Using OpenLS .................. 50
4.6 Conclusion .......................................................................................................... 52

5 Mapping and Reasoning with Ontologies ............................................................. 53
5.1 Introduction ......................................................................................................... 53
5.2 Ontology Integration ............................................................................................ 53
5.3 Mapping Between Concepts ................................................................................ 54
5.4 OWL Constructs for Mapping ............................................................................. 55
5.5 Mapping Method .................................................................................................. 56
5.6 Ways of Mapping .................................................................................................. 57
5.7 What is the Role of a Reasoning Engine? ............................................................. 58
5.8 Conclusion .......................................................................................................... 59

6 Results and Discussion .......................................................................................... 61
6.1 Introduction ......................................................................................................... 61
6.2 Mappings Between the Ontologies ...................................................................... 61
6.3 Guidelines for Ontology Mapping ........................................................................ 76
6.4 Ontology Based Discovery of Geo Services ....................................................... 77
6.5 Conclusion .......................................................................................................... 78

7 Conclusions and Recommendations ...................................................................... 79
7.1 Conclusions ......................................................................................................... 79
7.2 Recommendations and Future Work ................................................................. 81

REFERENCES .......................................................................................................... 83
List of Figures

Figure 2.1 IT Architecture Development (Source: Marks/Warrell) ............................................. 20
Figure 2.2 Service trading in SOA (Based on Gottschalk, 2002) .................................................. 22
Figure 3.1 URI and URL in the web .................................................................................................. 27
Figure 3.2. Semantic web as a Universal Medium for the exchange of data Service .................. 28
Figure 3.3. Semantic web operational schema ................................................................................. 29
Figure 4.1. Open LS Information Model (Source OpenGIS Consortium, 2005) ......................... 42
Figure 4.2. Open LS Schema Documents (XSD) ............................................................................ 43
Figure 4.3. XLS encoding of a typical directory service to find restaurants within 500 m of my University .................................................................................................................. 44
Figure 4.4. Open LS Route Service request parameters ................................................................. 45
Figure 4.5. Use Cases for the Case Study ......................................................................................... 48
Figure 4.6. Activity Diagram for the Use Case Find Nearby Restaurants ........................................ 49
Figure 4.7. Activity Diagram for the Use Case Find local events (In Enschede) .......................... 50
Figure 4.8. Ontology model for an example service: the gateway service ..................................... 51
Figure 4.9. OpenLS Gateway service modeled using asserted conditions ..................................... 52
Figure 5.1. Mapping concepts (Adapted from Noy [2003]) .............................................................. 55
Figure 5.2. OPERA get position operation with its asserted conditions ........................................ 61
Figure 5.3. Asserted and inferred hierarchies to find the user position in ESRI ontology ............... 63
Figure 5.4. Asserted and inferred models for finding the user position displayed with OWLViz plugin ........................................................................................................................................ 64
Figure 5.5. OPERA buffer Asserted (left) and Inferred (right) models displayed with OWLViz ...... 67
Figure 5.6. OPERA Find Nearby Restaurants Asserted and Inferred Models displayed with OWLViz ................................................................................................................................. 67
Figure 6.7. Ontology mapping strategy (Adapted from Kalfoglou [2003b]) .................................... 76
Figure 6.8. Ontology mapping Architecture (Adapted from Kalfoglou [2003]) ............................... 77
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT</td>
<td>Abstract Data Types</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AOI</td>
<td>Area Of Interest</td>
</tr>
<tr>
<td>DL</td>
<td>Description Logics</td>
</tr>
<tr>
<td>ESRI</td>
<td>Environmental Systems Research Institute</td>
</tr>
<tr>
<td>GI</td>
<td>Geographical Information</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>GMS</td>
<td>GeoMobility Server</td>
</tr>
<tr>
<td>GSI</td>
<td>Geographical Service Infrastructure</td>
</tr>
<tr>
<td>IS</td>
<td>Information Systems</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>KB</td>
<td>Knowledge Base</td>
</tr>
<tr>
<td>LBS</td>
<td>Location-Based Services</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>OWS</td>
<td>OGC Web Services</td>
</tr>
<tr>
<td>POI</td>
<td>Point Of Interest</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>Racer</td>
<td>Renamed ABox and Concept Expression Reasoner</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SDI</td>
<td>Spatial Data Infrastructure</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery, and Integration</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifiers</td>
</tr>
<tr>
<td>W3C</td>
<td>Word Wide Web Consortium</td>
</tr>
<tr>
<td>WCS</td>
<td>Web Coverage Service</td>
</tr>
<tr>
<td>WFS</td>
<td>Web Feature Service</td>
</tr>
<tr>
<td>XLS</td>
<td>XML for Location Services</td>
</tr>
<tr>
<td>XMI</td>
<td>XML Metadata Interchange</td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Services Modeling Ontology</td>
</tr>
<tr>
<td>WMS</td>
<td>Web Map Service</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Services Modeling Language</td>
</tr>
<tr>
<td>WWW</td>
<td>Word Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

An increasing number of geographical web services is available to the public through the web. Today, complex applications are built, integrating ready to use components, in a distributed environment enabled by Spatial Data Infrastructures (SDIs). Catalogs provide searchable repositories of information descriptions, but the mechanisms to support this task of discovery and retrieval are still insufficient [Klien et al, 2004]. It is not easy for a user to find the right service, i.e. with the right functionality and possibilities for his/her specific purpose [Lemmens et al, 2004].

To find the right service, a user needs to discover it using known functionality, attributes and standards available today. Matchmaking is done to compare the requirements of a single service given by the user with the service capabilities published by the service provider. For a service to be discovered (by humans supported by machine e.g. intelligent agents), will have to be described in a way that is human and machine understandable, and with capabilities that makes reasoning possible.

It has long been realized that the web could benefit from having its content understandable and available in a machine processable form, and it is widely agreed that ontologies play a role in providing much enabling infrastructure to achieve this goal [Baader et al, 2003]. Ontologies capture domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused, shared and operationalized across applications and groups [Bouquet et al, 2002].

Ontologies contain a taxonomy and a set of inference rules. The taxonomy defines classes of objects and relations among them [Lemmens et al, 2003]. Web based ontologies overcome the limitations of descriptions based on textual specifications, adding semantic information, ways to structure the concepts and automatic reasoning mechanisms.

1.2 Problem Definition

Metadata standards [ISO, 2003] provide means of describing content at different levels of detail. Textual overviews, feature catalogues and conceptual models are the common ways of describing data contents in metadata [Ahonen-Rainio, P, 2002]. Standards for descriptions of services such us WSDL are characterized by name, by free text description of the service, and by specification of input and outputs of the service. But this approach falls short to fully express the user needs. Problems of semantic heterogeneity caused by
synonyms and homonyms can arise during free-text search in catalogs [Klien et al, 2004]. This is because our language is full of ambiguities, different terms may refer to similar concepts, and the same terms may refer to different concepts.

1.3 Research Objective

The objective of this research is to test the usability of geo-operation ontologies during the discovery of geographical services. To achieve this objective the accomplishment of the following specific objectives is proposed:

1. Design and implement a mapping mechanism between operation ontologies to support reasoning based discovery

2. Develop guidelines for successful implementation, management and use of ontology-based reasoning mapping

1.4 Research Questions

The research questions have been formulated as follows:

- How is service discovery done and what are the shortcomings faced by web services users?
- How to model service functionality through service type classifications (taxonomies)?
- What techniques are available for doing mapping of web service capabilities?
- What considerations are required for doing reasoning with geo-operation ontologies?
- How to use mapping methods and tools and what guidelines can be defined for its implementation?

1.5 Related work

Support several applications for accessing multiple ontologies has been addressed by [Gahleitner, Wolfram, 2004]. They proposed a common interface for distribution and reuse of ontology mapping information based on the concept of semantic bridges is part of a communication framework that provides semantically enriched exchange facilities. Universal semantic unification of virtual integrated digital archive concepts is also described in [Costilla et al, 2004]. This work proposed a web integrated architecture based on mediator and wrappers containing an ontological kernel for providing a universal semantic unification.

A collaborative ontology development environment is also discussed in [Bouquet et al., 2004], integrating numerous aspects of ontology engineering and a methodology for guiding the ontology development process. By means of an application called OntoEdit it is possible
to develop an ontology based portal which aims the presentation of information to human and software agents taking advantages of semantic structures. They faced the conflicting needs for easy of use and construction of complex ontology structures. The problem of finding an agreement on the meaning of heterogeneous semantic models is presented by [Bouquet et al., 2003]. This approach shows how the semantic coordination can be solved using hierarchical classifications that could be applied for conceptualizing domain ontologies describing services and data types.

Klien et al [2004] are actually working in an architecture for ontology-based discovery and retrieval of geographic information. So far they have defined components such as Enhanced Cascading Catalog Service and the Reasoner component and plan to develop a query scenario in which the user is able to formulate a question using terms from the familiar shared vocabularies. Although they focus more on the architecture, this work seems to be a promising potential for collaboration efforts between the participant institutions.

1.6 Thesis Outline

This thesis is organized in the following chapters:

Chapter 1. Introduction
This chapter provides an overview of the research. The background of the study, the research problem, the objectives and the research questions are presented.

Chapter 2. Service Architecture Concepts
This chapter exposes key concepts involved in service architecture and discovery. The current implementations used for web service discovery, their benefits and shortcomings.

Chapter 3. Taxonomies and Ontologies of Geo-operations
Foundations about taxonomies and ontologies is presented. The process for building geo-taxonomies is explained through an example.

Chapter 4. Case study: Location-based Services
This chapter reviews concepts on location-based services, and contextualize the case study, describing the use cases and activity diagrams that model the scenarios.

Chapter 5. Mapping and reasoning with Ontologies
Mapping of concepts is presented and ontology reasoning mechanisms are explained.

Chapter 6. Results and Discussion
An overview of the results generated by the study is presented. The whole process followed to implement the mapping using geo-operation ontologies is presented using a case study in the location-based services domain.
Chapter 7. Conclusions and recommendations
To finish a discussion on the findings is presented, as well as recommendations and ideas for further research.


2 Service Architecture Concepts

2.1 Introduction

A System Architecture can be thought as the process and the discipline to develop efficient and effective Information Systems. Additionally System Architecture is in charge of aligning the technological processes with the business goals of organizations. System Architecture represents the wise use of the technological platform to provide the services expected by the users and their needs (McGovern et al, 2004).

Service-Oriented Architecture (SOA) is a logical result of outsourcing trends of not critical processes during the 90’s decade. SOA formally separates services, which are the functionalities that have to be provided, from consumers, who are systems that need the functionality. A contract is used to facilitate this separation; in this way consumers have to deal with not technical issues included in the contract and are separated from its implementation details.

Nowadays SOA represents the most promising architecture to be conformed by every technological implementation. But SOA can not be equated with Web Services, which are a realization of SOA.

With these set of background notes, this chapter pursues to explore the concepts that support Service Discovery which is one of the operations involved in the SOA paradigm; its characteristics, current approaches and disadvantages, and the ontological proposal are reviewed.

2.2 Service approach

Services can be found in the daily life: travel services, credit services, banking services, internet services and so on. Each of these services can be used in combination with the others thought the definition of agreements; an agreement is defined to control the interchange of execution orders of processes and the resulted products, the communication between the actors involved in the transaction, and aspects related with the Quality of Service (QoS) such as reliability, cost, response time, etc.

Bieber and Carpenter (2001) define a service as a “contractually defined behavior that can be implemented and provided by any component for use by any component, based solely on the contract”.


The special nature of services is their compose ability; for example: a travel agency can call a credit institution for a credit validation for a customer, then order a debit from a banking entity, and finally send an electronic message through internet. This characteristic allows the composition of service chains that tackle the necessity of solving complex business procedures supported by technological platforms.

In this context, the service concept is the next step in the evolution of software from functions and packages, objects and classes, and components (Weerawarana et al., 2005). It provides an abstraction of specific pieces of software from a functional decomposition, with the possibility of thinking about them in terms of services available on the network. This advantage frees the mind about details about implementation, programming models, and technology platforms (Endrei et al., 2004).

SOA surpass the constraints found in any opaque procedural system, and offers a mechanism based on a set of orchestrated sequences of messages, transformations, routes, and processing events. Its support technologies, such as XML, SOAP, and WSDL, let applications to declare and expose both the message contents and the functional components that operate on the messages.

The process of solution improvement offered between loose coupling and adaptability is outlined in figure 2.1. The SOA paradigm facilitates the design, development, implementation, and support of highly transparent and modular process-oriented applications and workflows that also conform to the typical IT departments' rigorous operational performance standards.

2.3 Loose coupling

Loose coupling enables services, to be joined together without care of differences in technology platforms or implementation models. This characteristic allows composing partial or full functionalities of discovered services on a demand basis. Loose coupling exists between a requester of a service and the service itself, if there is no a priori
knowledge about each other in terms of formats, protocols, models, or technology used by them.

Loose coupling directly intends to overcome the lack of interoperability between current distributed system technologies such as CORBA, J2EE, and COM. In contrast with trying to work with different and incompatible object models, loose coupling is actually relying on message-oriented, network oriented, and platform independent services available on the network.

Nevertheless loose coupling requires a wrapper that includes information about the interface that a service provides to its potential consumers. This wrapper has to be a standard between the actors of the transaction on the network.

Loose coupling benefits the consumer because consumer applications are protected from changes in the service provider implementation, and the consumer has a broader spectrum of providers. The point of agreement is the contract and its accomplishment.

### 2.4 Service-Oriented Architecture (SOA)

The Service Oriented Architecture (SOA) is a software architecture of distributed computing that takes software resources as a collection of services available on a network that can communicate with each other. A service can be considered as a function that is well-defined, self-contained, and does not depend on the context or state of other services. This communication can involve data transfer or two or more services coordinating an activity.

SOA offers a flexible and open platform, network oriented, based on message interchanging that support the exchange of information and execution of processes on heterogeneous systems.

SOA follows the “Publish, Find, Bind” paradigm (Figure 2.2). This paradigm is also known as service trading, and involves three roles: the service requester who is the interested user in receiving the candidate services’ profiles and the product of the selected service; the service registry in charge of announcing the available services, and the service provider, which is the direct responsible of executing the offered and selected service.

This paradigm addresses the discovery of available services instances through the publication of service descriptions, lookup of potential services for the requester, and invocation of the chosen service.

According to Endrei et al. (2004) SOA enables distributed computing architectures with long expected characteristics such as:

---

1 Software Architecture: coherent set of abstract patterns guiding the design of each aspect of a larger software system.
• Reduce cost and increase reuse.
• Faster time-to-market services.
• Easier integration and complexity management.
• Leverage of existing solutions.

![Service trading in SOA](Based on Gottschalk, 2002)

2.4.1 SOA Operations

2.4.1.1 Publish

The initial operation of SOA is to publish the service in a repository to announce the capabilities of the service. This operation generates an input in the service registry to be queried by the consumers. This operation allows to specify services with precision and with greater structure, because the service would eventually be invoked by parties that are not from the same administrative space as the provider of the service.

2.4.1.2 Find

The find operation allows SOA applications to deal with the challenge about discovering the services suitable and available for a specific purpose. The first thought about this issue is to organize a searching process on the set of published or advertised functionalities offered by a service provider, and discover them based on their descriptions, terms, and conditions of use.

The objective is that the user of the service does not have to know how the service is implemented or required, and that the service provided satisfies the expectations of the user.
2.4.1.3 Bind

Once the searching process has found some candidate services, the consumer has the possibility to choose the one that it believes is the most suitable for its purposes. In that moment the bind operation allows to pass the messages from the consumer to the service with data about specific details to be checked during the execution of the service; this level corresponds to the “conversation” between the consumer and the chosen service.

2.4.2 Benefits of SOA

SOA gets a set of interesting advantages to the organization, from management levels through technical issues. Some of them are:

- Decreased application development time and costs. The reuse of code is intensive, the code has been extensibility tested, and the only layer expected to be customized is the presentation one.
- Decreased maintenance time and costs.
- Increased corporate agility. The use of “off-the-shelf” solutions speeds up the implementation of flexible solutions, so it is feasible the keep the pace of new corporate strategies.
- Increased reliability on systems that are more resistant to failure and disruption.
- Decreased upgrade cost that can be achieved by components, instead of a total application replacement as used with monolithic applications.

2.5 Discovery of services

Each constituent layer of any implementation of SOA, such as web services, requires and defines a stack of layers that is fundamental to allow the interchange of messages between services. Nevertheless, the process of collecting and storing descriptions of these services has to be translated to a metadata understandable, discoverable and interchangeable by users who have in mind to accomplish a specific business solution.

This metadata becomes a functional repository to be shared between partners to advertise to themselves or to the general public a set of services, their interfaces, and other characteristics of services such as price and reliability.

2.5.1 Metadata

Metadata is commonly defined as information about information, trying to point out the capabilities to label, catalogue and describe the structure of information. The final goal of this capability is to make possible and expedite the searching of information by computers.

OGC (2005) defines metadata as a way to describe information to be understood by machines in the web. In this context the Resource Description Framework (RDF) has been
developed to model and encode metadata and allow the use of the web to document the meaning of the metadata.

Metadata is used by SOA to describe a service with its functional and non-functional characteristics. In this way, interfaces, data types, protocols, and so on, can be described in a way understood by potential requestors on the web.

### 2.5.2 UDDI

UDDI defines a service with metadata aggregation and offers specifications to find and query offered services.

- **Intra Enterprise UDDI:** This level of registry corresponds to the initial sharing of services inside an organization. The effort has a local organizational scope that intends to harmonize internal heterogeneities, and let the organization to test their approach and strengthen their services before an external exposure.

- **Inter Enterprise UDDI:** In this context, groups of organizations or communities define to share their services; knowledge domain specific is the driven reason to define boundaries between these registries.

- **Public UDDI:** Corresponds to the open exposure and announcement of services. It means that anybody, after satisfying a set of requirements such as price, contracts and so on, has the possibility to execute them.

### 2.6 Current drawbacks of service discovery

Current practical alternatives to discover services assist the searching of feasible services for the solution of business requirements. However, they have a set of difficulties that encourage the proposal of new and sometimes challenging alternatives.

At first, the use of UDDI requires that the solution developer has at least a pale idea about the location of the service. The solution developer has to use a well-known directory at design time to verify if the found service is compatible with the requirement to solve.

After the developer finds the directory, a set of queries has to be sent against the registry to get detailed information about the service and bind the implementation. This information feeds the sequence of messages that the developer is arranging for the solution and executed on runtime. The other alternative is to use a well-known partner UDDI that has in advance the services used for a domain-specific area of knowledge or business.

The assurance of the right selection of service shows the second problem: actually these UDDI repositories follow standard metadata definitions, usually ISO specifications. This practice could be a suitable solution for the beginning, but to have a level of flexibility to
be applicable to a broad range of services, some values of the metadata specification accept free values.

Free values can be null, something following a business vocabulary, or anything without predefined structure. Here we are not including problems related with different languages, evolution of languages, or diversity of knowledge areas.

The use and misused of free values metadata generates different types of result; initially a not match situation can be generated when the feasible service is not found due to its poor or inconvenient description. In that case the user has to be using another service or rethink the used keywords. The other possible result, and worse than the first one, is that the selected service is not appropriate for the solution in process; in this case the developer of the solution relies on the service, includes it in the sequence of messages for the solution, and generates an unexpected result.

The main technical problem is generated by the differences in assumptions about the semantics of the service; these differences are generated because the administrative space of the consumer and provider are different, and their effect can be devastating.

From a more managerial point of view, the implementation of a SOA solution requires time and money. In many organizations there is no budget management, so the final user of a system, and sometimes acting as the manager of the project, is responsible for the whole project; in these circumstances, if the decision is to implement a SOA solution, an adjustment to the organizational model is required to facilitate this kind of projects (McGovern et.al., 2004).

2.7 Conclusion

Our discussion has presented the advantages of SOA and their implementations; a special attention has been paid to the paradigm that facilitates SOA. Literature and current projects’ documentation suggest that SOA and its implementation, web services, are the most exciting paradigm and technology available to speed up the implementation of better and faster solutions.

The discovery process has to solve and support the huge task of finding the more suitable services for every specific request. How to deal with the semantic differences? Current solutions say that they rely on metadata and UDDI, but the real status of this process is that they rely on knowledge disseminated between people. The big problem is a matter of understanding: how to be sure that my concept is at least in the same category that my potential provider’s?

This is the type of question that next chapter is intending to tackle with common understanding on domain areas.
3 Taxonomies and Ontologies of Geo-Operations

3.1 Introduction

Human vocabulary is full of vagueness. If we meet a friend downtown and he asks: “Please tell me the shortest route from home to school”. We will be able to answer because we can process the language by “context”. We can relate words with our knowledge about the situation, giving meaning to them. If we give this command to a computer machine, it will certainly find lots of difficulty trying to give us the answer. We should be more specific, and some questions may arise: what city do you live in? What are the routes? What is your home address? Where do you study?

This is well known as the semantic problem. A similar situation happens on the web. Think about a google search using the key word “Java” and do not be surprised to get thousands of hits about the Java Island, the Java Language, Coffee from Java, … and more. The problem is that on the web most of the information is written in HTML and expressed in natural language. The Semantic Web can be seen as a huge engineering solution. It was thought up by Tim Berners-Lee, inventor of the WWW, URIs, HTTP, and HTML: “The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. [Berners-Lee, T et al, 2001]

The semantic web will provide a universal medium for data exchange characterized by a common data representation framework, integration of multiple sources and formats, increase of utility of information (by connecting it to definitions and to its context), more efficient information access and analysis and successful searches through concepts instead of keywords.

In the previous chapter the current shortcomings of service discovery were presented. In this chapter, we start with a compact presentation about a semantic way to continue with an outline of an alternative approach to drive the user in the finding process of the service. Since it uses taxonomies and ontologies of geo-operations, we will first start to develop the foundations on the concepts, and further in the next chapter we will explain the proposed solution.
3.2 Semantic Web

3.2.1. What Is The Problem?
To solve the question proposed in the introduction about the shortest route, a “smart software agent” needs to identify objects in the Universe of Discourse (UoD) of the problem. Uniform Resource Identifiers (URI) are the standard way to identify items if the web is used; they are assigned to anything, and are a requirement to accept that something exists in the web. To find these URIs in the web, Uniform Resource Locators (URL) are used as addresses to visit every webpage; they work as identifiers and locators.

Actually any person can create a URI; there is practically no central control that constrains its creation. (Except for domain names, and some URI schemas such as http). This great flexibility and freedom generates some of the heterogeneity problems presented by some authors such as [Bishr, 1997]; [Bishr, 2002]; and [Ouksel, 1999]; it will be inevitable to end up with multiple URIs that represent the same item, so we are not able to say what a certain URI exactly means. [Swatz, 2002]

At the end of the day the web is full of URIs and URLs that need to be understood by humans because these addresses and links do not have a meaning by themselves. A name such as www.mystuff.com/myfriendquestions.html can be understood only by people who have spent some time reading the web page and checking now and then that the “behavior” of the URL has not changed. This situation can be multiply by thousands when somebody decides to look for information and tools to support a research: that person has to go URL by URL, understand its contents, accept or decline its use based on its contents, and be aware about its changes over the time.

---

2 Universe of Discourse: that part of reality that the Information System will represent.
A task almost impossible to be achieved with the current number of resources available on the web.

### 3.2.2. Semantic Web Proposal

So far it has been identified that the major obstacle for the automation of machines participation in the smart use of data available of internet is the design of this data: it has been set up only for human use. With this constraint it is not possible to think about automated browsers on this huge amount of data.

According to Tim Berners-Lee defines Semantic Web as “a web of data, in some ways like a global database” [Berners, 1998].

![Semantic web diagram](image)

**Figure 3.2.** *Semantic web as a Universal Medium for the exchange of data Service*

By the same author [Berners, 2001], Semantic Web is envisioned as an extension of the current web, with a well defined schema to give meaning to data; in this way computers will have the capability to understand the meaning of data spread all over the web.

Actually, computers are used to display data from the web, but the universality of this data is constraint by the lack of a common understanding about the facts stores in this massive database that the web has become. The issue that the Semantic Web intends to tackle is how to make humans and computers work together, how to create intelligent software agents to “understand” what the user means and figure out a reasoning path to offer solutions, not only one but several with their corresponding rationale that explains the reason for its proposal.

Some interesting characteristics of the current web are its universality and decentralization: HTML has the possibility of linking anything to anything, without
any restrictions (just security issues), crossing cultural, geographical, and technological barriers. And its decentralization allows a permanent service of the web, but does not guarantee the permanent availability of all the data. Quality of Service (QoS) will be a topic to check to guarantee the trusted use of these new smart resources.

### 3.2.3. How should the Semantic Web Work?

The logical question to answer after the presentation of the Semantic Web vision is how all these expectations should work together. Figure 3.3 outlines the process to follow. The initial trigger is a human request, a topic to be investigated, solved, or extended with data from internet. This human defines his request to an agent, a robot in charge of searching on meaningful tags in internet, and generating new facts from data found on the web. This smart agent makes use of annotations and ontologies to get the meaning of the resources available for its quest; this way to assign and find meaning to internet resources is one of the backbone components of semantic web. Knowledge representation, both in facts and inference rules, are expected to conduct the automatic reasoning of these agents. [Berners, 2001]

![Semantic web operational schema](image)

After getting a set of possible facts, relationships, and inferences, the agent receives his first proposals to solve his question. In that moment the human can continue using the agent to improve the search or go directly to the annotated web page to solve and/or extend his question.
3.3 What is a Taxonomy?

A taxonomy describes a classification structure for content or other information according to a pre-determined system. Almost anything, animate objects, inanimate objects, places, and events, may be classified according to some taxonomic scheme [Wikipedia, 2005].

Taxonomies are frequently hierarchical in structure in order to permit us to understand the relationships among entities and between entities and properties which are responsible for their character in the real world. However taxonomy may also refer to relationship schemes other than hierarchies, such as network structures. Other taxonomies are single children with multi-parents, a simple organization of objects into groups, or even an alphabetical list [Wikipedia, 2005].

Taxonomies have been used in many fields for a long time, especially in biology where is used as a methodology of classifying organisms based on physical and other similarities [CoRIS, 2005]. Furthermore we can find Soil Taxonomy that tries to group similar soils into increasingly general categories [Rossiter, 2004]. Benjamin Bloom created this taxonomy for categorizing level of abstraction of questions that commonly occur in educational settings [Bloom, 1984]. And also we can find taxonomies that enable an organization to consolidate its terminology and categories into single enterprise taxonomy [Wordmap, 2005]. With the advent of the internet, there has been increased interest in using taxonomies for structuring information for easier management and retrieval [Hunter, 2001].

3.4 GIS Taxonomies

The use of taxonomies in GIS is not an exception. The widespread application of computers and use of GIS have led to the increased analysis of geographic data within multiple disciplines [Percivall, 2002]. The constant increase of the number of GI services on the web have generate the necessity of compose them to enable more complex processing tasks [Lutz, 2004]. For the purpose of interoperability, standards have been developed for the interfaces of those services. GIS and software developers will use these standards to provide general and specialized services that can be used for all geographic information. [Open Geospatial Consortium, 2001].

An important example of these approaches is the international standard ISO 19119 adopted by the OpenGIS Consortium (OGC) as part of OpenGIS Abstract Specification, Topic 12 "Service Architecture" [Open Geospatial Consortium, 2002]. The ISO19119 International Standard provides a framework for developers to create software that enables users to access and process geographic data from a variety of sources across a generic computing interface within an open information technology environment [Percivall, G, 2002].

The geographic services taxonomy consists of the titles of the categories and the definitions for the categories (see Table 3.1.). Systems shall use the geographic services taxonomy to
organize its services. A specific service shall be categorized in one and only one category, unless it is an aggregate service that may perform services from more than one category.

| - Geographic human interaction services
| - Geographic model/information management services
| - Geographic workflow/task management services
| - Geographic processing services
  |   - Geographic processing services – spatial
  |   - Geographic processing services – thematic
  |   - Geographic processing services – temporal
| - Geographic processing services – metadata
| - Geographic communication services
| - Geographic system management services

Table 3.1 Geographical Services Taxonomy

3.5 Geo-Operations and Geo-Services

The terms “operations” and “services” in the geography domain have a high relevance in this document and need to be defined. We will stick to the international standards definitions: ISO 19119 defines “an operation as a specification of a transformation or query that an object may be called to execute; a service as a distinct part of the functionality that is provided by an entity through interfaces; and an Interface as a named set of operations that characterize the behavior of an entity” [OGC, 2002].

These three concepts are tied together in a model called OpenGIS Web Services (OWS) architecture that works in this manner: Each service is a collection of operations, accessible through an interface, which allows a user to evoke a behavior of value to the user. When provided with certain parameters through such interfaces, the service calculates a result, which is served to an application client [McKee, 2003].

According to the OGC Web Services vision, any internet device potentially has access to an unlimited spatial data resources and spatial processing resources. Web Services are “web accessible applications and application components that exchange data, share tasks, and automate processes over the Internet. Because they are based on simple and non-proprietary standards, Web Services make it possible for computer programs to communicate directly with one another and exchange data regardless of location, processing platforms, operating systems, or languages” [McKee, 2003].

As the development of web services is going on, but still in progress, in our work we do not want to restrict it only to the set of web services that is currently available (very limited compared to the large amount of geoprocessing operations in a proprietary architecture). Consequently, we will build “virtual” web services from “collection of operations”, even
though in the real world they still are not commercially available. This assumption (although is strong) gives us more flexibility at the time to decide about the modeling of the taxonomies and free will in the kind of mappings that we want to do, as we will see later in the next chapter.

3.6 The core element: a taxonomy of Geo-Operations

Lemmens [2006] proposed a taxonomy of geo-operations; his idea was to derive this taxonomy from a feature model that puts together elements found before in ISO geographic standards (19119, 19109, 19123, and 19103), OGC (Topic 5), some new elements, and add value linking them in a single model named Opera-R.

Operations are grouped according to their function. In the upper level we find: human interaction operations; feature data acquisition, storage and exchange; feature data processing and analysis; feature data presentation manipulation; service creation and management; service execution; and some operations for manipulation of Meta Information. Lower levels in the classification helped to find different operations according to specific criteria, such as the kind of data the operation was acting on (feature, raster), their input and output parameters, and other elements that helped to group operations (please see Lemmens [2006] for a full description).

The use of the taxonomy can be understood when doing the search for a web service, since a user may be in one of the following situations:

1) The user knows the name of the operation from one GIS provider, but not from others; for example “find route”.
2) The user does not know the name of the operation in any system, but he may have an idea of what the operation performs; for example key words as “distance” or “cluster” could be used to find specific operations.
3) The user only knows a very broad category of what he/she wants to do; for example, performing “networking” operations.

The taxonomy requires a method to be used to categorize operations and collection of operations. The idea tested in this research was to use this taxonomy for classification of geographical operations and to identify potential similar geo-operations found in different models. The vision is that having a complete taxonomy of geo-operations, and doing the mapping of concepts from different GIS systems to that taxonomy, will help this user to find a broader list of services, since the search can start acting from any point in the hierarchy, and the user could navigate in any direction and abstraction levels (upper or down in the taxonomy).
3.7 What are Ontologies?

An ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, databases, and applications that need to share domain information. Ontologies indicate the hierarchies and relationships that exist between different resources within a specific domain. Ontologies include computer Usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable [W3C, 2004a].

Ontologies, as a form of knowledge representation, classically are defined as "explicit specification of conceptualizations" [Gruber, 1993]. An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. Ontologies are documents that formally define the relations among terms. They are social contracts to agree about an explicit semantics, understandable to outsiders and often derived in a community process.

3.8 From Taxonomies to Ontologies

We have decided to represent our GIS operations with ontologies. An ontology is a schema that formally defines the hierarchies and relationships between different resources. Semantic Web ontologies consist of a taxonomy and a set of inference rules from which machines can make logical conclusions. A taxonomy in this context is a system of classification, such as the scientific kingdom/phylum/class/order/etc. system for classifying plants and animals that groups resources into classes and sub-classes based on their relationships and shared properties.

While taxonomies are sets of hierarchical vocabulary terms, ontologies go further than that, because they use richer semantic relationships among terms and attributes, as well as strict rules about how to specify terms and relationships. Ontologies are suited to express the hierarchical relationships that exist between resources. We can use ontologies to assign properties to classes of resources and allow their subclasses to inherit the same properties.

The fundamental difference between an ontology and a conventional representational vocabulary (such as a taxonomy) is the level of abstraction and relationships among concepts. The value that they add to existing vocabularies such as thesauri or classification schemes is additional information that defines how objects can be classified and related to one another [Quin, J. Paling, S., 2001]. Ontologies also have deeper semantics for class/subclass and cross-class relationships. But the main point is that ontologies are represented with a formal description language based on Description Logic (DL) in order to have the ability to reason about descriptions.

---

3 A domain is just a specific subject area or area of knowledge, like medicine, real estate, geography, financial management, etc.
3.9 Description Logics

Over the past 20 years, the AI community has been studied a set of knowledge representation languages defined as Description Logics (DL)\(^4\). DL has been effectively used as a modeling language by several domains including medicine, digital libraries, software engineering, and database management, among others. In the last few years has gained considerable attention as a language for the development of the Semantic Web [Kemke, Mithun, 2004].

Description Logics uses graphs to represent knowledge in a given application domain. The basic elements used by DL are [Baader, Nutt, 2003]:

- Atomic Concepts: unary predicates, they represent sets of individuals
- Atomic Roles: binary predicates, they represent relations between individuals
- Individuals

Description Logics have the characteristic to have a formal, logic-based semantics, using operators to separate the structure of concepts and roles. Description Logics has improved knowledge representation and reasoning in many ways including formalizing many ideas such as concepts and roles.

DL systems consist of two components. First, the knowledge base (KB), which can further be divided into the TBox and the ABox. Second, the reasoning engine, which implements the various inference services.

The knowledge base contains information expressed in explicit way. Using it, the KB can obtain implicit, new information. The kind of reasoning we can do using Description Logics is [Baader, Nutt, 2003]:

- Reasoning with the terminology (TBox). TBox contains the concepts that conform the vocabulary of the application in a given domain
- Reasoning with the assertions (ABox). ABox contains statements about individuals using the terms declared in the vocabulary. ABox introduces the assertional knowledge (world description).

The idea is to have the ability of reasoning that make possible to infer implicitly represented knowledge from the explicitly knowledge that is contained in the knowledge base.

\(^4\)http://dl.kr.org/
3.10 Resource Description Framework (RDF)

Resource Description Framework (RDF) offers a way to encode knowledge in triple sets; these triples intend to represent the relationship between a subject and an object through the use of a verb. At the beginning, a RDF sentence looks like a common sentence, but the “magic” in its proposal is that the words are URIs and can be written using XML tags. So each RDF instance and RDF graph consist of an assertion about a subject or a resource, with a property or relationship, and certain values such an object or resource. This schema shows a “natural” way to describe the huge amount of data to machines and become their potential an enormous knowledge base available everywhere.

In this context the sentence: “Jim Smith studies in ITC” could be represented (see figure 3.4) as:

\[
<\text{http://jsmith.com/}>
\text{http://swterms/study}
<\text{http://www.itc.nl/student/>}
\]

The schema offered by RDF lets build in information nets with meaningful URIs. The disadvantage of this proposal is the requirement to have an accepted unique meaning for each concept.

3.11 Web Ontology Language (OWL) basis

The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language. An OWL ontology is an RDF graph (RDF Concepts), which is in turn a set of RDF triples. As with any RDF graph, an OWL ontology graph can be written in many different syntactic forms [W3C,2004c].

Some basics of OWL and some important design considerations will be introduced using a small fragment of the geo-operations ontology. In OWL, classes are used to represent...
concepts from the ontology. *CalculateAspect, Resampling, GridOverlay* are all geo-operations, they are the concepts in our ontology (examples from OPERA-R (Lemmens, 2006)).

Classes are organized into a super class-subclass hierarchy. Subclasses specialize (‘are subsumed by’) their super classes. Subsumption in OWL means necessary implication, so in our geo-operations ontology when we say that “MapDisplay operation is a subclass of HumanInteraction operation” means that “All map display operations are human interaction operations”. In OWL, classes are overlapping until disjointness axioms are entered. We used them to express that anything can’t be, for example, Blue and Red at the same time.

The basic object in OWL is an individual. OWL classes are interpreted as sets that contain individuals. In this example, UTM and WGS84 are individuals from the Class CoordinateReferenceSystem. An ontology together with a set of individual instances of classes constitutes a knowledge base.

Another important object is a property. OWL uses properties to link two individuals together, in a similar way as databases use relations and Protégé uses slots. Allowed classes for properties are often called a range of a property. The classes to which a property is attached or classes which property describes, are called the domain of the property. For example the property *hasReferenceSystem* links individuals from the domain CoordinateSet to individuals from the range CoordinateReferenceSystem.

In OWL properties are used to create restrictions that are used to limit the individuals that belong to a class. Properties can have different restrictions describing the value type, allowed values, the number of the values (cardinality), and other features of the values the property can take. The number of values a slot can have for each instance is limited by the slot’s cardinality. If a maximum cardinality is not defined, the slot can have any number of values.

Restrictions are sometimes called constraints or axioms. The main purpose of constraints is to express machine-readable meaning to support accurate automated reasoning.

There are two quantifiers that may be used:

- The existential quantifier (∃), which can be read as at least one, some, or some values from
- The universal quantifier (∀), which can be read as only, or all values from

For our *buffer* operation class, the existential quantifier from figure 3.5, represent all of the individuals that at least have one input symbol that is a *GeometricObject*. In other words, in order for something to be a buffer operation it is necessary for it to have a (at least one) Input Symbol that is a Geometric Object.
Restrictions can be expressed as necessary or necessary and sufficient. Necessary conditions can be read as, “If something is a member of this class then it is necessary to fulfill these conditions”. Necessary and sufficient condition states that “not only are the conditions necessary for membership of the class, they are also sufficient to determine that any (random) individual that satisfies them must be a member of the class”.

OWL calls primitive (partial) classes, the classes having only necessary conditions, while it calls defined (complete) classes, the classes with necessary & sufficient conditions. According to Drummond et al. [2004] the most common accidental error in implementing OWL ontologies is to fail to make a set of restrictions a definition - i.e. to fail to make the definition “complete” rather than “partial”, since nothing will be inferred to be subsumed under a primitive class by the classifier.

3.12 Ontology development environment

We used the Protégé system\(^5\) (version 3.1 build 195) at Stanford's Knowledge Systems Laboratory to create the ontology. Protégé provides a development environment for ontology construction. Protégé is a free, open source ontology editor and knowledge-base framework. In addition to editing tools, it allows the creation of modular, combinable ontologies. Protégé is based on Java, is extensible, and provides a foundation for customized knowledge-based applications. [Protégé, 2005].

Many formal languages to specify ontologies have been proposed for the Semantic Web, such as OIL, DAML+OIL\(^6\), OWL, SHOE, and RDF [Doan et al, 2003]. We used the OWL - Web Ontology Language\(^7\), to define and instantiate our ontology. The Protégé-OWL plug-in used was 2.1 beta, build 275. OWL is a language intended to be used when the information needs to be processed by applications, instead of just being presented to humans.

The advantage for expressing meaning and semantics of OWL is its ability to represent machine interpretable content on the Web [W3C, 2004b]. OWL also utilizes the XML Schema data types and supports class axioms such as subClassOf, disjointWith, etc., and class descriptions such as unionOf, intersectionOf, etc. Many other advanced concepts are included in OWL, making it the richest standard ontology description language available

---

\(^5\) [http://protege.stanford.edu/](http://protege.stanford.edu/)

\(^6\) Predecessors of OWL

\(^7\) [http://protege.stanford.edu/plugins/owl](http://protege.stanford.edu/plugins/owl)
today. OWL has three sub languages, each with increasing complexity: OWL Lite, OWL DL, and OWL Full. We choose OWL DL since it gave us enough complexity and level of detail required by our semantic model and still provides decidability.

The reasoning tool we used was RacerPro -Renamed ABox and Concept Expression Reasoner-, version 1.8.1. It is a knowledge representation system that implements the description logic (DL) ALCQHRI+ (SHIQ) and provides facilities for algebraic reasoning [RacerPro, 2005]. RacerPro accepts OWL DL subsets and reads OWL ontology documents from web servers (more on reasoning in section 4.6). [Haarslev; Möller, 2004].

3.13 Development Methods for Ontology based solutions

Software engineering suggests that the definition and use of a method offers a reliable environment to improve the quality of software and speed up its process of construction. A general framework of a software method includes analysis, design, development, testing, implementation, and maintenance. The question is how this general framework has to be adjusted, modified, or extended to include ontology based solutions.

Some of the reasons to develop an ontology, further explained by Noy and McGuinness, [2000] are:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

According to [Dong, 2005] Software Engineering “Modeling and verification techniques can be useful at many stages during the design, maintenance and deployment of ontologies”. Our previous discussion can demonstrate that the semantic web foundations give a pivotal role to ontology, because it provides the mechanism to share knowledge inside communities and between them. On the other hand the behavior of software agents as explained in section 2.3 demands a consistent ontology to make their decision based on knowledge and its correct application following the rules and constraints defined in the knowledge base.

So far, a lot of effort has been invested in the definition and improvement of languages, frameworks, and tools to support the semantic web, but methods and strategies for its development are still in their very early stages. [Knublauch, 2004].

Some potential issues to be tackled with ontology based applications include an error-handling strategy that relies somewhat on the ontology and the agent, so the error can be controlled and its

---

effect will not affect the later inferences generated by the model. If there is an error in the initial stages of an inference process, the final results will be useless, if they are identified as wrong results. The worst scenario to have in mind is that these wrong results are taken as right and some decisions are made based on them.

The explanation of an inference is another issue to construct for semantic web applications. The development process has to generate reliable, fast, and stable agents, but the product is not complete without a component in charge of explaining the path that the agent followed to get the inferences.

So from the ontology point of view, the development problem can be split in two worlds. The first one corresponds to the construction of the ontology itself, and can be named Semantic Web Layer; this layer is understood as the set of facts, relationships, and rules to generate inferences. The second one, or Internal Layer, corresponds to the construction of agents or engines that will make the search for the human user; this layer is in charge of the reasoning and control of the process [Knublauch, 2004]. The big difference is that the first component is expected to be developed and maintained by humans; meanwhile the second one is expected to be executed by a machine that is the final goal of the semantic web, to make data available to computers.

The Internal Layer can be constructed with the use of black boxes a set of pre-build components that consume ontologies and generate inferences\textsuperscript{9}. The Semantic Layer has to be a translucent component with a very high level of quality [Knublauch, 2004], and it becomes the asset that every stakeholder will want to keep, expand, and protect.

Due to that the Internal Layer can be built based on off-the-shelf components, the most suitable developing method to be applied would be eXtreme Programming (XP): use of generic libraries, high level pre-existent models, and focused on the interface with the user. For the development of the Semantic Layer the set of requirements about its stability, long term projects, size of people involved, and high quality, suggests the use of formal and disciplined methods with high control on schedule and products.

### 3.14 Concluding remarks

From vagueness in our daily language it is possible to define a kind of organization of concepts and restrictions to give structure to knowledge and allow its sharing between communities and across organizations. Taxonomy gives a hierarchy structure and constructs vocabulary, and ontologies offers relationships between concepts and restrictions, giving higher levels of abstraction and including semantics for class/subclass relationships.

Nevertheless when a consumer wants to select a concept, even in the same domain area, the big challenge is to get the set of potential ones from different efforts that can be

\textsuperscript{9} Black boxes software involves the development of programs from the customer’s view point with no knowledge of how the program was built. For example, racer is a black box to us, we know it was build by humans, but we use it without understanding the details of its implementation.
represented as taxonomies and ontologies. Current ontology languages available give us the flexibility to integrate an editing environment to represent knowledge, along with an interface to generate new classification of concepts.

Next chapter will present a case study in location services that involves the necessity to search between different ontologies to find similar services and an ontology mapping between them in order to generate inferred hierarchies. Location services is just one application domain in which mapping could improve the discovery process in web service paradigm, but the concepts and techniques are applicable to other scenarios as well.
4 Case study: Location-Based Services

4.1 Introduction

The integration of GIS with consumer devices, such as cell phones, has spawned a whole new industry—Location-Based Services –LBS– [ESRI, 2005]. They are wireless services dependent on a certain location, for example, the location of the requesting mobile station, or from another positioning service, such as GPS [Hjelm, 2002]. LBS are defined as applications that deliver location-based information when and where it is needed. Users can access these services via the desktop, web browser, mobile phone, personal digital assistant (PDA), pager, or other device. LBS provide users of mobile devices personalized services tailored to their current location.

Location-based services answer three questions: Where am I? What's around me? How do I get there? They determine the location of the services, advice users of current traffic conditions, supply routing information by using one of several technologies for determining position, and then use the location and other information to provide personalized applications and services. Traffic advisories, navigation help including maps and directions, and roadside assistance are natural location-based services. Other services can combine present location with information about personal preferences to help users find food, lodging, and entertainment to fit their tastes and pocketbooks. [Mahmoud, 2004]

In this chapter we will first introduce the location-based services information model used for this research. In a geographical data model, reality (the real world) is represented as a series of geographical features. Creating a data model involves sampling the continuous or analogue space of reality and representing it in a discrete form [Maguire, Dangermond, 1991]. For this research we have chosen two different modeling schemas. The first one follows the specification in location services given by Open Geospatial Consortium and well known as OpenLS. The second one describes part of the geoprocessing commands used by ESRI’s ArcGIS.

Having the two different approaches of modelling, we will introduce the case study used in this research as a proof of concept. It is important to clarify that the models have a large bunch of operations from which we have extracted only a small fragment to create our ontologies required to solve our case study. Let us finish saying that the case study will give us a well known context of application of ontologies and mapping techniques, but does not limit the use the concepts to this domain.
4.2 OpenGIS Location Services (OpenLS)

The Open Geospatial Consortium defined a Specification for the implementation of Location Services (OpenLS). It aims to create an open location service platform defining access to the Core Services and Abstract Data Types (ADT). This platform is referred as the GeoMobility Server (GMS). [Open Geospatial Consortium, 2005].

GMS provides subscribers with location-based application services and content. It uses open interfaces to access the location of the user and access to the OpenLS core services. The Core Services include directory, gateway, geocoder, reverse geocoder, presentation and route services. Content is also provided such as maps, routes, addresses, traffic, points of interest and the possibility to access local content databases. Content is hosted on GMS, but also on other servers and accessed through the Internet. Other supporting functions for personalization, context management, billing, logging, among others, and a set of local applications build upon the Core Services and accessing them through OpenLS interfaces, could also provisionally be provided by GMS.

![The OpenLS Information Model](image)

Figure 4.1 shows the OpenLS Information Model. Content is exchanged using OpenLS standard Abstract Data Types (ADTs) and structures for location information. Since all the...
core services “talk” using the same defined schemas (ADTs), communication between the services is possible.

An example scenario is the request a service to display “where I am” in a map on my mobile device. Solving this request requires the concourse of three services: gateway service finds my current position; location utility service converts coordinates given by gateway into an address; presentation service portraits the map into the mobile device. The model is very flexible, since it allows adding parameters to specify constraints to the request, as needed.

Core Services are the heart of the OpenLS specification, since they provide the functions for performing location operations. However, another very useful concept provided by the specification is the use of Abstract Data Types. Area of Interest, Point of Interest, Address, Position, Map, Location, Route Geometry, are all standard ADT’s, they are defined as application schemas that are encoded in XML for Location Services (XLS). The relationship between different OpenLS services viewed with Altova Schema Agent Software\(^10\) is shown in figure 4.4.

![Figure 4.2. Open LS Schema Documents (XSD)](image)

Requests are expressed through a predefined standard list of XML tags with optional and mandatory parameters. Answer is obtained through response parameters. To understand the way a request is made, a small fragment of code of the XLS encoding for a typical directory query is shown below.

---

\(^{10}\text{Altova Schema Agent enables to manage multiple schemas and build relationships between schemas from within a Graphical User Interface (GUI).} \text{ http://www.altova.com/} \)
**Example Query: Which Restaurants are within 500m of my University? (Directory service)**

```xml
<DirectoryRequest>
  <POILocation>
    <WithinDistance>
      <POI ID="1">  
        <POIAttributeList>
          <POIInfoList>
            <POIInfo name="POI Name" value="My University"/>
          </POIInfoList>
        </POIAttributeList>
      </POI>
      <MaximumDistance value="500"/>
    </WithinDistance>
  </POILocation>
  <POIProperties directoryType="Yellow Pages">
    <POIProperty name="NAICS_type" value="Restaurant"/>
  </POIProperties>
</DirectoryRequest>
```

Figure 4.3. XLS encoding of a typical directory service to find restaurants within 500 m of my University

The example shows a proximity directory service constrained by the position of a university and a given distance that serves as boundary for the query. The response may consist of a text address list, a location map, or a route map and a list of directions from the source place.

A short description of the Core Services follows:

The first service is the *Gateway Service*. This service is employed to obtain the position data of the subscriber’s mobile terminal from the network and forwards to the Location Service Client, which may store it for as long as needed.

The second service is the *Directory Service*. It is like a “yellow pages” catalog to find a specific or nearest place, product or service. It is an online directory that grants access through search criteria to a Point Of Interest (POI) by some property (entering the name, type, category, keyword, phone number, or some other “user-friendly” identifier), or by its location relative to some other POI (nearest to me, within some distance, within some boundary area).
The third service is the Route Service. This service determines a route for a subscriber using a navigation application. It requires route start and end points. Optionally allows specifying waypoints, route preferences (shortest, fastest, least traffic, most scenic, etc.) and the preferred mode of transport. Figure 4.2 presents part of the XML schema request parameters for the route service.

The fourth service is the Location Utility Service. This service performs geocoding by determining a geographic position, given a partial or complete address (place name, street address or postal code). It also performs reverse geocoding by determining a complete, normalized address (place name, street address, postal code), given a geographic position. It answers questions such as: Where I am or where is the location of a service? Once the address is normalized, it can be used for other services, such as routing, presentation, etc.

The fifth service is the Presentation Service. It renders geographic information for display on a Mobile Terminal. Any OpenLS Application may call this service to obtain a map of a desired area of OpenLS ADT.
4.3 ESRI geo-processing commands

ESRI present a classification of geographical operations in ArcGIS according to the functionality provided. Functionality is defined as all the data collection, storage, manipulation, analysis and presentation operations carried out by a Geographical Information System (GIS). [Maguire, Dangermond, 1991].

The scheme is intended as a top-down hierarchical classification of the major types of functions which characterize GIS. The idea behind this is that a satisfactory classification of GIS will enable systems to be organized in a common framework using standard terminology. Our assumption was that each function or a composition of functions can be thought as a potential future web service, so instead of restrict ourselves to only the list of available web services, we decided to abstract the concept and use the full list of geo-processing commands to have a rich set for mappings.

In ESRI, all commands are referred to as tools, scripts, or models and are maintained in toolsets within the ArcGIS toolboxes. A model consists of one or multiple processes strung together; a toolbox contains a list of the toolsets and tools; and a tool is a single geo-processing command.

Toolboxes are the first concept in ESRI hierarchy. They are organized according to the geo-processing commands it contains. In our ESRI taxonomy we find at the top level the following toolboxes: analysis, cartography, conversion, coverage, data management, geocoding, linear referencing, spatial analyst, spatial statistics, and 3D analyst toolbox.

In the second level of the hierarchy we locate the toolsets for each of the toolboxes. And in the third level we finish with the geo-processing commands. A small fragment of the ESRI taxonomy follows to illustrate the concept hierarchy. Although required and optional parameters are described in the model, we have decided not to include them in the ontology to keep it simple.

Analysis toolbox
  Extract toolset
    Clip
    Select
    Split
    Table Select
  Overlay toolset
    Union
    Intersect
    Symmetrical Difference
  Proximity toolset
Cartography toolbox  …
4.4 The TRAVELER System

The case study for this research is the TRAVELER system. TRAVELER is a computerized application used to advise a person to find exciting activities to do during a journey abroad. It is typically used for a tourist traveling to a city. It includes hardware components such as a Personal Digital Assistant (PDA) –a handheld computer or personal organizer device–, an optional Global Positioning System (GPS) –a system of satellites and receiving devices used to compute the position of the tourist on the Earth–, and software to run the system.

4.4.1 Motivating Scenario

Following is the description of a typical scenario for the TRAVELER system:

“Silvhia, as researcher of the Geosemantics group is visiting Enschede to make a presentation as part of the program of the SVG conference, organized by Twente University and ITC in the Netherlands. At the end of the afternoon, Silvhia wants to have dinner at a Dutch restaurant and do some evening activity. She uses her PDA and activates the system to find a nearby restaurant and interesting things to do after. The system uses knowledge of her preferences and location capabilities offered by her PDA.

The system accesses the Internet via the wireless connection available at the conference. It finds a list of nearby restaurants that she might like. Silvhia asks for location, menu specialties and prices. The system displays a map and detailed information of each pointed restaurant and Silvhia selects one. The system calculates the shortest route to the selected restaurant and displays it.

After dinner, she asks the system for local evening activities to do. The system accesses the internet and it finds current shows, concerts, dancing places and cinema films Silvhia might like. Taking into account her preferences and timing, the system displays an ordered list of suitable activities and she selects some. After, Silvhia goes to her hotel”.

This is just an example of the kind of request that the system handles\(^\text{11}\). However, like this, any traveler visiting any city may ask for any tourist-related request, and the system must give an answer to the request. Some examples are: finding hotels in Praga, searching for the nearest metro station to Vatican city, or finding bus routes from the airport to the Van Gogh Museum in Amsterdam.

4.4.2 Using UML to model the system

The Unified Modeling Language (UML) is used to express the requirements of the system. Writing use cases—stories of using a system—is an excellent technique to understand and describe requirements. In this case, one package and two use cases are envisioned, as seen in Fig. 4.5.

\(^{11}\) The example was inspired by the work done for the International Federation for IT and Travel & Tourism (IFFIT) available at http://www.ifitt.org/
Activity diagrams are used to model the procedural flow of actions that are part of a larger activity. In the case study, they were used to model each of the specific use cases at a more detailed level. Fig. 4.6 shows the flow of actions for finding nearby restaurants and Fig. 4.7 shows the flow for finding events.
Figure 4.6. Activity Diagram for the Use Case Find Nearby Restaurants
4.5 Building location-based geo-operation ontologies using OpenLS

OpenLS Ontology was derived from the OpenGIS Location Based Service Implementation Specification. Three main classes comprise the model: service, parameter and ADT. Service has subclasses for each of the core-services. Parameter is a class that group input and output parameters used for the services. ADT contains all the standard abstract data types used by the parameters. We have modelled the specification in this way since it let us reusing classes, multiple inheritance and good flexibility.
We illustrate the model using the gateway service, shown in Figure 4.8. Concepts are indicated by circles, properties indicated by curved lines and individuals indicated by diamonds.

![Ontology model for an example service: the gateway service](image)

In Protégé we have created the hierarchy according to this model (Figure 4.9). Asserted conditions are used to express restrictions in the classes. For example, to say that the output parameter retrieved by the gateway service (OutputMSInformation) is type PositionADT, we use these expressions:

\[
\begin{align*}
OLF_{\text{GatewayService}} & \models OLF_{\text{hasOutputParameter}} \, OLF_{\text{OutputMSID}}^{12} \\
OLF_{\text{OutputMSInformation}} & \sqsubseteq OLF_{\text{OutputMSID}} \\
OLF_{\text{OutputMSInformation}} & \models OLF_{\text{hasADT}} \, \text{PositionADT}
\end{align*}
\]

These expressions give us some flexibility since having independent parameters from data types could be used as restrictions. They offer interesting elements for setting up the mappings explored in next chapter. One assumption is that having similar restrictions (for example, having input and output

\[^{12}\text{User Position}\]
parameters) might give us an “idea” of a kind service that we are trying to find during the discovery process.

![Figure 4.9. OpenLS Gateway service modeled using asserted conditions](image)

### 4.6 Conclusion

Current technological products such as cell phones and GPS required the participation of LBS to satisfy market expectations. “W&H” questions such as Where, What and How are expected to be answered in the least possible response time regardless of the service provider. The searching of feasible services available involves the participation of different proposals; for this example OpenLS, derived from the OpenGIS Location Based Service Implementation Specification, has been analyzed and chosen to solve the case study.

There could be many different ways of modelling the ontology given the specification. We have analyzed and decided for one using three main classes: service with subclasses for each of the core-services; parameter to group input and output parameters used for the services; and ADT for containing all the standard abstract data types used by the parameters.

The next chapter will discuss the required foundations in ontology mapping, some basis in description logics and a summary of reasoning mechanisms available to give a solution to the case study.
5 Mapping and reasoning with ontologies

5.1 Introduction

Ontologies are critical for applications that want to search across or merge information from diverse communities. Although XML is sufficient for exchanging data between parties who have agreed to definitions beforehand, their lack of semantics prevent machines from reliably performing this task given new XML vocabularies. The same term may be used with (sometimes subtle) different meaning in different contexts, and different terms may be used for items that have the same meaning. Interoperability requires agreements on the definitions of concepts. Ontologies help to standardize sets of concepts and formal descriptions of those concepts. Concepts can be shared for a community who explicitly agree in using the same meanings. [W3C, 2004a].

Organizations sharing the same universe of discourse may define their set of ontologies with their own semantics. Many ontologies will describe similar domains, but using different terminologies, and others will have overlapping domains. In order to integrate them we must know the semantic correspondences between their elements. [Doan et al, 2003]. However, there are terms which are already embedded in a particular community. For example, ESRI users know what a clip operation performs. Sharing a single ontology between communities of GIS users who are familiar with those existing terms, would mean to find similar terms to the concepts the users are already familiar with, and change them accordingly. This will not always likely happen.

Communities who use different terms for items that have the same meaning, may agree to map such concepts between them. In this chapter we will explore how the mapping between concepts can be done, explain the techniques available for doing it and describe through our case study, how this can be implemented. It also explains how reasoning on ontologies can be done. Given two or more ontologies and having already defined the mappings between concepts, this chapter explores how reasoning on the mappings is achieved.

5.2 Ontology integration

Ontologies have gained popularity in the Artificial Intelligence (AI) community as means for establishing explicit formal vocabulary to share between applications. In fact, the goal of ontologies is to facilitate knowledge sharing.

In order to have two different ontologies together, one can introduce an importing mechanism. This allows using parts (or all) of previously created domain
conceptualizations. When integrating several ontologies, one can perform a *match* operation between concepts from the ontology. The *match* could be interpreted as:

a. To be exactly like; correspond exactly
b. To be like with respect to specified qualities

In our experiment, we identify similar concepts for all the ontologies and try to identify how close they are in their semantics, so that we can establish a known relationship. Results of the matching operations are mappings between terms that belong to different ontologies.

Having two ontologies that need to be integrated, the easiest way to integrate them is to have the designers of the two original ontologies get together and compare the semantics of the concepts. To determine correlation between two ontologies, one domain expert must find all the concepts in the source ontologies that are similar to one another, determine what the similarities are, and either change the source ontologies to remove the overlaps or record a mapping between the sources for future reference. But this is not always feasible, there are simply too many ontologies available and they are too large to have manual definition of correspondences as the primary source of mapping.

The heuristics for the integration include analysis and comparison of concept names, properties that are attached to concepts, domains and ranges of properties, restrictions on properties (such as cardinality restrictions), subclasses and super classes, and so on.

Noy and Musen [2003] argue that a similarity metric between concepts in OWL ontologies can be developed as a weighted combination of similarities of various features in OWL concept definitions. According to Noy [2003], the tools for automatic and semi-automatic ontology alignment use the following features in ontology definitions (to various extents):

- Concept names and natural-language descriptions
- Class hierarchy (subclass–superclass relationships)
- Property definitions (domains, ranges, restrictions)
- Instances of classes
- Class descriptions (as in DL-based tools).

### 5.3 Mapping between concepts

The wide-spread use of ontologies by diverse communities and in a variety of applications in an effort to share knowledge has led the need of integrate them [Kalfoglou Y.; Schorlemmer M., 2003b]. Information processing across these ontologies is not possible without knowing the semantic mapping between them. As a manual process is impossible at the web scale, the development of tools to assist in the ontology mapping process is crucial to the success of the Semantic Web. [Doan et al, 2003]
Sometimes users prefer to keep the source ontologies separate for several reasons: compatibility with data and tools using the source ontologies, maintenance of relations between ontologies when the source ontologies evolve, and others. In that case, all the user may want to do is finding and save correspondences between different ontologies [Noy and Musen, 2003].

Ontology mapping could be defined as the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected [Kalfoglou; Schorlemmer, 2003a].

OntoWeb [2002] developed a survey of the most relevant ontology tools and semantic web technology available in the community, including ontology merge and integration tools. Based in this survey it is possible to identify two approaches to mapping ontologies: merge ontologies making lattices from their instances, or make the merging process search for the concepts that have some similarity.

### 5.4 OWL constructs for mapping

All OWL constructs can be used for mapping, since mapping is in fact nothing else than defining a concept and relate it with others [Lemmens, 2006]. Following we find a list of some primitives [W3C, 2004b] used for mapping more than others:

**Owl: equivalentClass.** Two classes may be stated to be equivalent. Equivalent classes have the same instances. Equality can be used to create synonymous classes.
Owl:equivalentClass construct is even more powerful: one can use it to create a Boolean combination of the classes in a source ontology and equate that to a class (or even another Boolean combination) in a target ontology.

For example, RouteStop can be stated to be equivalentClass to WayPoint in a Route Service. From this a reasoner can deduce that any individual that is an instance of RouteStop is also an instance of WayPoint and vice versa.

**Owl: equivalentProperty.** Declare that two edge labels denote the same relationship. Equivalent properties relate one individual to the same set of other individuals. Equality may be used to create synonymous properties.

**Owl: same as.** Two individuals may be stated to be the same. These constructs may be used to create a number of different names that refer to the same individual. Owl does not adopt the unique-names assumption; just because two instances have a different name does not imply that they are different individuals.

**Owl: differentFrom.** An individual may be stated to be different from other individuals. Explicitly stating that individuals are different is important in Owl since it does not assume that individuals have only one name.

**Owl: allDifferent.** A number of individuals may be stated to be mutually distinct in one statement.

### 5.5 Mapping method

Noy and Musen [2003] distinguished in their work between the notions of merging and alignment, where “merging is defined as the creation of a single coherent ontology” and “alignment as establishing links between ontologies and allowing the aligned ontologies to reuse information from one another” [Kalfoglou; Schorlemmer, 2003a]. According to Noy and Musen’s definition, in our work we are doing alignment of the ontologies, we establish some links between the ontologies and we leave the original ontologies intact.

The bridge ontology is used to specify the mappings between the operation ontologies. The correspondence between two ontologies is expressed as a set of axioms relating classes and properties of the two source ontologies to the bridge ontology. Since the two ontologies are stored in (possibly) two different namespaces, the bridging axioms are basically translation rules referring to concepts from the source ontologies to the bridge ontology. Then, by transitivity if two different concepts from the source ontologies refer to the same concept in the bridge ontology, the concepts are similar in the source ontologies (see figure 5.1, bridge ontology would be the one in the lower part).

This method (as well as the merging mechanism) provides different ways of linking concepts from the source ontologies. It provides means for defining many-to-one or many-to-one mappings.
to-many aggregation relationships between concepts in the source ontologies, as well as one-to-many concept-decomposition relations. It allows specification of recursive mappings, complex mappings between that collect information from several related concepts.

5.6 Ways of Mapping

The way to specify the mappings is through similarity axioms that link concepts from two ontologies. Our concepts are GIS operations, so what we do is try to compare the functionality of the operations and make the relations. A prefix is used to identify each ontology: ESRI (as described in section 4.3), Opera (as described in section 3.6) and OLS (OpenLS, as described in section 4.2). The ways to specify the mappings are:

- **Technique 1:** The simplest way is a one-to-one (1-1) mapping between the elements, using mapping constructs. This technique is possible only when a single operation matches directly another single operation in a different model, having both of them the same functionality.

  For example, we can do direct mapping between concepts in the following way:

  \[
  \text{ESRI:Buffer} \equiv \text{OLS:MaximumDistance}
  \]

  This expression could be paraphrased as “Buffer operations in ESRI are exactly like (or correspond exactly) maximum distance operations in OpenLS” according to the perception of similarity described in section 5.2.

- **Technique 2:** The second way is a one-to-many (1-n) mapping between the elements, by using subclasses, super-classes and properties to describe the concepts and using the reasoner to find the similarities.

  For example:

  \[
  \text{ESRI:Clip} \subseteq \text{Opera:Within}
  \]

  This expression maps the clip operation in ESRI ontology to the within operation defined in the Opera ontology by using a subsumption relationship.

  Another example,

  \[
  \text{OLS: PortraitMapRequest} \subseteq \\
  \text{Opera: ShowMap} \cup \text{Opera: AttributeDisplay}
  \]

  This expression maps the Portrait map request operation in OpenLS, with show map and attribute display operations in Opera.
• **Technique 3: The third way** is by using asserted conditions.

This technique is illustrated with an example presented by Lemmens [2006]:

\[
\text{riskmap:ProvincialRoad} \equiv \exists \text{symbol:hasThematicAttributeType top10nl:RegionalRoad}
\]

This expression maps the concept of provincial road in a hazard mapping ontology to the concept of regional road defined in the top10nl data model of the Dutch topographic service.

A second example inspired from the previous technique is:

\[
\text{Opera:GetPosition} \equiv \exists \text{OLS:hasInputParameter OLS:InputMSID}
\]

This expression maps the operation to find the user position in Opera ontology to the concept of position retrieving given by the gateway service in OpenLS (OLS), helped by an asserted condition specified by MSID (the identification of the mobile subscriber).

We presented three ways of mapping here, but there are some variations possible. These techniques will be illustrated using some examples from our ontologies in chapter 6.

### 5.7 What is the Role of a reasoning engine?

One of the main services offered by a reasoner is to test whether or not one class is a subclass of another class. By performing such tests on all of the classes in an ontology it is possible for a reasoner to compute the inferred ontology class hierarchy.

The basic inference on concept expressions in Description Logics is *subsumption*, typically written as \( \text{C} \sqsubseteq \text{D} \). Determining subsumption is the problem of checking whether the concept denoted by \( \text{D} \) (the subsumer) is considered more general than the one denoted by \( \text{C} \) (the subsume). In other words, subsumption checks whether the first concept always denotes a subset of the set denoted by the second one [Baader et al, 2003].

Another typical inference on concept expressions is concept satisfiability, which is the problem of checking whether a concept expression does not necessarily denote the empty concept. In fact, concept satisfiability is a special case of subsumption, with the subsumer being the empty concept, meaning that a concept is not satisfiable [Baader et al, 2003].

Some of the services offered by the reasoner are:
• To derive new information via automated inference (fundamental role)
• To ensure integrity of instances in a knowledge base
• To ensure the logical consistency of the ontology itself
• To provide reasoning across ontologies, and hence integrate data from multiple nodes [Halevy et al, 2003]
• To build and maintain sharable ontologies by revealing inconsistencies, hidden dependencies, redundancies, and misclassifications [Rector, 2003]
• To reason over ontologies, done by general logic-based theorem provers [Bergamaschi, S et al, 2005]

The OWL Plugin provides direct access to reasoners such as Racer. Racer supports two types of DL reasoning: Consistency checking and classification also called subsumption. [Knublauch, 2004]. Consistency checking tests whether a class could have instances. In Racer words: “Is the set of objects described by a concept empty?” Classification infers a new subsumption tree from the asserted definitions. In racer words: “Is there a subset relationship between the set of objects described by two concepts?” [Haarslev, Moeller, 2003].

5.8 Conclusion

Two ontologies like OpenLS and ESRI define a different approach of modeling part of some GIS functionality. The name and the description of a concept are too general to communicate the meaning. The model may be somewhat different for example in the vocabulary used, or in the assumptions made, or in the task performed, while sharing the same conceptualization.

Given the need to give access to the user to “as many services” as possible during the discovery process, we have discussed in this chapter the idea that building mappings between “similar” operations in different schemas, can give us an added value of finding new and undefined services. The findings are possible through the use of a reasoner that finds new subsumption classes (inferred hierarchy) given a set of asserted definitions established by a domain expert.

Next chapter summarizes the results of our experiments with the ontologies. We present details of the mappings required in our three ontologies and finish with some methodology issues and guidelines for the elaboration of the mappings.
6 Results and Discussion

6.1 Introduction

In this chapter a summary of our experiences with web ontology language OWL, description logics, and a reasoner such as RacerPro are presented. An evaluation of results obtained in our experiments is presented as well.

Initially there is an overview over the mappings used for the case study, and the description step by step of the solution provided by each ontology to solve the two use cases included in the case study. From that description, the corresponding Opera-R operations [Lemmens, 2006] are identified, when they are available. A deep understanding about the different operations provided by each model and their corresponding attributes was required to identify the set of operations to be used in the solution.

How to capitalize this experience to generate future mappings of ontologies? At the end of the chapter it is included a proposal, which is a modified version from Kalfoglou [2003b], with the intention of give some discipline to the mapping process.

6.2 Mappings between the ontologies

In our case study we started with three different ontologies:

- OpenLS, was derived from the Open Geospatial Location Services Implementation specification [OpenGIS, 2004]
- ESRI geo-operations, was derived from the ESRI geo-processing commands [ESRI, 2004]
Opera-R, was adapted from Lemmens [2006]. It contains a taxonomy for classification of
generic geo-operations

We will now list the geo-operations required in each of the ontologies for solving the use case
described in section 4.3.1. step by step. After the independent geo-operations are identified, we find
the similarities in the three models and establish the semantic links that tie them together. Since the
expressions are quite long, containing the full paths (hierarchy in the taxonomy), for the mappings
they will written in a simplified manner trying to make them more readable.

USE CASE 1: SELECT RESTAURANTS NEARBY MY LOCATION

“At the end of the afternoon, Silvhia wants to have dinner at a Dutch restaurant and do some evening
activity. She uses her PDA and activates the system to find a nearby restaurant that she might like.
Silvhia asks for location, menu specialties and prices. The system displays a map and detailed
information of each pointed restaurant and Silvhia selects one. The system calculates the shortest
route to the selected restaurant and displays it.”

STEP 1. Get the user position

The system needs to get the position where Silvhia is located.

We will describe the operations required in each model to solve this step. Dots are used in the
notation to express the hierarchy in the ontologies. First, Opera uses a GetPosition geo-operation,
added to the original geo-ontology from Lemmens [2006]:

\[
\text{OPERA: OperationModelElement.} \\
\quad \text{FeatureDataOperationElement.} \\
\quad \text{FeatureDataOperation.} \\
\quad \text{FeatureProcessingOperation.} \\
\quad \text{AcrossAttributeTypes.GetPosition} \\
\exists \text{OPERA:hasOutputSymbol symbol:CoordinateSet}
\]

Gets the user location. Figure 6.2 shows how the asserted conditions are expressed in Protégé. The
condition in this case asserts that in OPERA the Get Position operation has as output parameter a
coordinate set.

Second, OpenLS (OLS) uses the gateway service for acquiring the user position.

\[
\text{OLS: Service.GatewayService.SLIR} \\
\exists \text{OLS:hasInputParameter OLS:InputMSID}
\]

\[
\text{OLS: Service.GatewayService.GatewayResponse} \\
\exists \text{OLS:hasOutputParameter OLS:OutputMSID}
\]

Obtains position data for mobile terminals. SLIR stands for Standard Location Immediate
Request. As input parameter the operation requires the mobile subscriber identification
(InputMSID) and as output it retrieves the position of the subscriber (OutputMSID).
Third and last, ESRI uses the Add XY Coordinates

\[ ESRI: \text{DataManagementToolbox.FeatureToolset.AddXYCoordinates} \]
Calculates and adds \( xy \) coordinates of points to the layer

Mapping between the concepts can be expressed as:

\[ OLS: \text{GatewayResponse} \equiv \exists \text{Opera:hasOutputSymbol symbol:CoordinateSet} \]

\[ ESRI: \text{AddXYCoordinates} \equiv \exists \text{OLS:hasOutputParameter OLS:OutputMSID} \]

First expression maps the concept Gateway response from OpenLS ontology to the concept Get position from OPERA ontology using its asserted conditions (technique 3 from section 5.6). Second expression maps the concept Add XY coordinates from ESRI ontology to the concept Gateway response from OpenLS ontology using its asserted conditions. Some other ways to express the mappings can be found, using similar restrictions. We selected one example of such set of possible mappings to illustrate the technique.
Figure 6.3 shows two columns: on the left is the asserted hierarchy and in the right the inferred hierarchy, showing the concepts derived for the reasoner in blue color. What we want to obtain is that given the mappings between OLS to Opera and between ESRI to OpenLS, the reasoner can infer the relationship between ESRI and Opera. Figure 6.4 displays basically the same information but in a graphical form, using OWLViz plugin. Notice that the inferred hierarchy shows the new links between the ESRI and OpenLS.
**STEP 2. Find nearby restaurants**

*Silvhsia requested a list of nearby restaurants from her current position.*

ESRI needs a sequence of four geo-processing commands for finding nearby restaurants, as follows:

- **ESRI: AnalysisOperation.ProximityToolset.Buffer**  
  Calculate buffer around x,y coordinates (the user position). In the exercise a fixed a priori chosen distance value was used to quantify the “nearby” concept.

- **ESRI: GUI.LayersAndTablesToolset.SelectLayerByAttribute**  
  Creates, updates or removes the selection on a layer or table view using an attribute query. Here we select from all the POIs (museums, sport centers, parks, etc) only for the typical Dutch restaurants.

- **ESRI: AnalysisOperation.ProximityToolset.PointDistance**  
  Computes the distance between each point in a feature class or layer to all points in a different feature class or layer. Here we measure the distance from the user location to the selected restaurants from the previous step.

- **ESRI: AnalysisOperation.ExtractToolset.Clip**  
  Extracts those features from an input feature class that overlap with features from a clip feature class. We select the restaurants inside the buffer layer.

In contrast, OpenLS uses the directory service and specifies some parameters for finding the restaurants, as follows:

  - `hasInputParameter (POILocation □ WithinDistance □)
    `PropertyName □ PropertyValue`

  - `hasOutputParameter ListOfPOI`

This service provides subscribers with access to an online directory to find a near place, product or service. As input parameter it requires the start location of the subscriber, maximum distance from start point, property name and value (restaurant, Dutch). As output parameter retrieves the list of Point of Interest (POI) satisfying the request and the distance from the source location.
Opera has defined a similar group of operations using feature selection geo-operations. Sequences of operations can be visualized as a “bag” of operations that together meet a specified purpose, in this case the specified search. In OWL it has been expressed using the Union operator (\( \cup \)).

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.FeatureDataOperation.}
\text{FeatureProcessingOperation.FeatureSelection.SelectByThemAttrib}
\]
Selects features based on a query on thematic attribute types (using as criteria “Dutch restaurants”).

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.FeatureDataOperation.}
\text{FeatureProcessingOperation.DistanceBased.FixedDistance.Buffer}
\]
Creates geometric objects that have a fixed distance to input geometric objects

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.FeatureDataOperation.}
\text{FeatureProcessingOperation.SpacialMeasurement.DistanceMeasure}
\]
Creates geometric objects that have a fixed distance to input geometric objects

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.FeatureDataOperation.}
\text{FeatureProcessingOperation.FeatureSelection.SelectByTopo.SelectByWithin}
\]
Selects features based on a topological relationship with a geometric object. We need to select the restaurants within the buffer area

Using all this knowledge a set of mapping can be defined between the ontologies. Starting from very simple 1-1 mappings:

\[
\text{ESRI:Buffer} \equiv \text{OLS:MaximumDistance}
\]
This expression maps the concept of buffer in ESRI geo-processing commands to the concept of maximum distance defined in the OpenLS data model.

\[
\text{OPERA:Buffer} \equiv \text{ESRI:Buffer}
\]
This expression maps the concept of Buffer defined in the OPERA ontology to the concept of buffer in ESRI geo-processing commands. The obvious inference given the two previous expressions is to find the equivalence between the OpenLS maximum distance and the Opera Buffer.

Such inference in fact is done by the reasoner as shown in Figure 6.5. See the asserted model to the left of the graph, it shows only the knowledge given (the expressions just described). To the right we see the inferred model displaying the new knowledge found by the reasoner, new concepts are linked to the existing concepts.

Another simple mapping is the following:

\[
\text{ESRI:PointDistance} \equiv \text{OPERA:DistanceMeasure}
\]
This expression maps the concept of Point distance defined in ESRI geo-processing commands to the concept of distance measure defined in OPERA ontology.

![Image](image_url)

**Figure 6.5.** OPERA buffer Asserted (left) and Inferred (right) models displayed with OWLViz

More complex mappings could also be defined, as follows:

First, let’s group some OPERA operations into one new concept, find nearby restaurant, to make the expressions easier to read and the mappings neat:

\[
\text{OPERA:FindNearbyRest} \equiv \\
\text{OPERA:SelectByThemAttrib} \downarrow \\
\text{OPERA:Buffer} \downarrow \\
\text{OPERA:DistanceMeasure} \downarrow \\
\text{OPERA:SelectByWithin} \\
\]

Second, let’s express the mapping of concepts using asserted conditions from OpenLS ontology:

\[
\text{OPERA:FindNearbyRest} \equiv \\
\exists \text{OLS:hasInputParameter} \quad (\text{OLS:POILocation} \downarrow \text{OLS:MaximumDistance} \downarrow \\
\text{OLS:PropertyName} \downarrow \text{OLS:PropertyValue})
\]
This expression maps the operation find nearby restaurants defined in the OPERA ontology to the operation directory request in OpenLS using assertions. In this example we have used the input parameters from OpenLS to do the mappings. Similarly, various concepts that perform the restaurant search in ESRI may be grouped as a new concept in ESRI ontology:

\[
\text{ESRI:DiscoverNearbyRest} \equiv \\
\text{ESRI:Buffer} \downarrow \\
\text{ESRI:SelectLayerByAttribute} \downarrow \\
\text{ESRI:PointDistance} \downarrow \\
\text{ESRI:Clip}
\]

Second, let’s express the mapping of concepts using asserted conditions from OpenLS ontology:

\[
\text{ESRI:DiscoverNearbyRest} \equiv \exists \ OLS:\text{hasInputParameter} \ (OLS:\text{POILocation} \downarrow OLS:\text{WithinDistance} \downarrow \\
(OLS:\text{PropertyName} \downarrow OLS:\text{PropertyValue}))
\]

This expression maps the operation discover nearby restaurants defined in ESRI ontology to the operation directory request in OpenLS using asserted conditions. Since now we have two complex operations from both ontologies ESRI and Opera mapped to the same set of asserted conditions expressed in OpenLS, we can expect that an inference can be generated finding the mapping from ESRI to OPERA automatically.

Figure 6.6 shows the inferred hierarchy to the right. Note that the link between ESRI Discover Nearby Restaurant and OPERA Find Nearby Restaurant is there in the graph.
displayed with OWLViz plugin. This is the most complex mapping operation that we implemented in our ontologies.

**STEP 3. Display map and show details of restaurant**

*Sylvia wants to see the restaurants displayed in a map, she pointed to different restaurants and the system shows detailed information of each restaurant, such as menu specialities and prices.*

Opera-R has a group of human interaction operations, among all the display operations. For this step, we can use these two operations as follows:

\[
\text{OPERA: OperationModelElement.} \\
\quad \text{HumanInteractionOperationElement.} \\
\quad \text{MapDisplay.ShowMap}
\]

Based on a feature selection, this operation renders a map for displaying in the client.

\[
\text{OPERA: OperationModelElement.} \\
\quad \text{HumanInteractionOperationElement.} \\
\quad \text{AttributeDisplay}
\]

Based on a feature selection, this operation renders feature attributes for displaying in the client, e.g. a table display. Here we display detailed information about the pointed restaurant (menu specialities, prices, etc.)

Let’s group some OPERA operations into one new concept, display and show details, to make mappings easier to read:

\[
\text{OPERA:DisplayAndShowDetails} \equiv \\
\quad (\text{OPERA: ShowMap} \uparrow \text{OPERA: AttributeDisplay})
\]

OpenLS has a service specifically for presenting results in the screen, it is called: presentation service.

\[
\text{OLS: Service.PresentationService.PortraitMapRequest} \\
\exists \quad \text{OLS:hasInputParameter (OLS:CenterPoint} \uparrow \text{OLS: DisplayScale})
\]

\[
\uparrow \\
\text{OLS: Service.PresentationService.PortraitResponse} \\
\exists \quad \text{OLS:hasOutputParameter} \quad \text{OLS: Map}
\]

Renders geographic information for display on a mobile terminal. Any OpenLS service call upon this service to obtain a map of a desired area, with or without map overlays required. More parameters can be specified to define the size and encoding of the portrayed map, such as width, height, format, etc. Here we include just few to give an idea of the request. No detailed information is available in OpenLS.
ESRI uses two operations for displaying the map and detailed information of restaurants:

\[
ESRI: GUI.DrawMap \\
\text{Renders a map in the screen}
\]

\[
ESRI: GUI.LayersAndTablesViewsToolset.SelectLayerByAttribute.SelectFeatureByQuery \\
\text{Get detailed information of pointed restaurants}
\]

Let’s group two ESRI operations into one new concept, draw and query, to make mappings easier to read:

\[
ESRI: DrawAndQuery \equiv (ESRI: DrawMap \sqcup ESRI: SelectFeatureByQuery)
\]

Some mapping can be expressed as follows:

\[
ESRI: DrawAndQuery \equiv OPERA: DisplayAndShowDetails
\]

This expression maps the complex concept of draw and query in ESRI to the concept of display and show details defined in OPERA ontology.

\[
OLS: PortraitMapRequest \sqsubseteq OPERA: DisplayAndShowDetails
\]

This expression maps the concept portrait map request in OpenLS to the concept of display and show details defined in OPERA.

With these definitions we can check whether the concept denoted by Opera (the subsumer) is considered more general than the one denoted by OpenLS (the subsume). One possible inference could be that OLS is subsumed by ESRI.

**STEP 4. Show shortest route to selected restaurant**

_Silvha wants to obtain the shortest route from her current location to the selected restaurant._

Opera-R has a category of geo-operations for managing linear features:

\[
OPERA: OperationModelElement.FeatureDataOperationElement. \\
FeatureDataOperation.FeatureProcessingOperation. \\
Acumulation.Network.RouteOptimization
\]

Finds the optimal route between two points. Optimal may be “shortest”, “fastest”, or constrained by any other criterion.
ESRI uses the linear referencing toolbox for manipulating routes:

\[ \text{ESRI: } \text{FindRoute.RouteStop} \]
Add route stops

\[ \text{ESRI: } \text{FindRoute.RouteFinderOptions.RouteOptions.RouteType} \]
Calculate shortest route to selected restaurant (shortest, fastest)

\[ \text{ESRI: } \text{FindRoute.RouteFinderOptions.MapImageOptions} \]
Display map with route

\[ \text{ESRI: } \text{FindRoute.RouteFinderOptions.RouteDisplayOptions} \]
Display route instructions

Let’s group some ESRI operations into one new concept, Find Shortest Route, to make the expressions easier to read when doing the mappings:

\[ \text{ESRI: } \text{FindShortestRoute} \equiv (\text{ESRI: } \text{RouteStop} \sqcup \text{ESRI: } \text{RouteType} \sqcup \text{ESRI: } \text{MapImageOptions} \sqcup \text{ESRI: } \text{RouteDisplayOptions}) \]

OpenLS has a service specifically for manipulating routes, it is called: route service. We defined expressions using input and output parameters as follows:

\[ \text{OLS: Service.RouteService.RouteRequest} \]
\[ \exists \quad \text{OLS: hasInputParameter} \quad (\text{OLS: RoutePreference} \sqcup \text{OLS: WayPointList} \sqcup \text{OLS: RouteMapRequest} \sqcup \text{OLS: RouteInstructionsRequest} \sqcup \text{OLS: RouteGeometryRequest}) \]

\[ \text{OLS: Service.RouteService.RouteResponse} \]
\[ \exists \quad \text{OLS: hasOutputParameter} \quad \text{OLS: RouteMapOutput} \]

Determines a route for a subscriber. Many input and output parameters can be specified, we have chosen only a few just to illustrate the case study, but more can be defined according to what is provided by each model.

Some mappings can be expressed using concepts defined above, as follows:

\[ \text{ESRI: } \text{FindShortestRoute} \equiv \]
\[ \exists \quad \text{OLS: hasOutputParameter} \quad \text{OLS: RouteMapOutput} \]
This expression maps the operation find route defined in ESRI ontology to the operation route response in OpenLS using assertions. Note that a similar expression could be defined for input parameters, we are just illustrating one of the many possible ways of doing mappings.

\[ ESRI: \text{FindShortestRoute} \sqsubseteq \text{OPERA: RouteOptimization} \]

This expression maps the operation find shortest route defined as a complex concept in ESRI to the operation route optimization defined in the OPERA ontology. Note that here we are using subset relation instead of an equivalence relation. The parent concept, in this case OPERA:RouteOptimization is a more general concept than the child concept, the ESRI:FindShortestRoute concept.

Having the two previous mappings, one can expect that the reasoner checks whether the concept denoted by Opera (the subsumer) is considered more general than the one denoted by ESRI (the subsume).

**USE CASE 2: FIND LOCAL EVENTS**

“... After dinner, Silvha asks the system for local evening activities to do. The system accesses the internet and it finds current shows, concerts, dancing places and cinema films Silvha might like. Taking into account her preferences and timing, the system displays an ordered list of suitable activities and a map and she selects some”.

**STEP 5. Find local current evening activities**

Silvha requested a list of local evening activities to do.

ESRI needs a sequence of two geo-processing commands for finding the activities, as follows:

\[ ESRI: \text{DataManagementToolBox.LayersAndTableViewsToolset.SelectLayerByLocation} \]

Creates updates or removes the selection of features in a layer based on a spatial relationship. Using this command we restrict our search of activities to the geographic location (Enschede in our case study) where the user is located

\[ ESRI: \text{DataManagementToolBox.LayersAndTableViewsToolset.SelectLayerByAttribute} \]

Creates updates or removes the selection of features on a layer or table view using an attribute query. We used this command to select activities by attribute date (only for tonight).

Let’s group some ESRI operations into one new concept, Find Nice Activities, to make the expressions easier to read when doing the mappings:

\[ ESRI: \text{FindNiceActivities} \equiv (ESRI: \text{SelectLayerByLocation} \sqcup ESRI: \text{SelectLayerByAttribute}) \]
In contrast, OpenLS uses the directory service and specifies some input and output parameters for finding the local activities, as follows:

\[
\text{OLS: Service.DirectoryService.DirectoryRequest} \\
\exists (\text{hasInputParameter POILocation} \sqsubseteq \text{WithinBoundary} \\
\sqsubseteq (\text{PropertyName} \sqsubseteq \text{PropertyValue}) \sqsubseteq \text{SortCriteria} \sqsubseteq \text{SortDirection})
\]

\[
\exists \text{hasOutputParameter ListOfPOI}
\]

We can use this service in a similar way as we described in step 2, the only difference would be that we can specify as input parameter the polygon to search within (AreaOfInterest ADT), property name and value and also determine an order in the list. As output parameter retrieves the list of Point of Interest satisfying the request and the distance from the source location.

Opera has defined a similar group of operations using feature selection geo-operations:

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.} \\
\quad \text{FeatureDataOperation.FeatureProcessingOperation.} \\
\quad \text{FeatureSelection.SelectByThemAttrib}
\]

Selects features based on a query on thematic attribute types (using as criteria “local activities”).

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.} \\
\quad \text{FeatureDataOperation.FeatureProcessingOperation.} \\
\quad \text{FeatureSelection.SelectByThemAttrib} \\
\quad \text{FeatureSelectionBasedOnGeometricObjects}
\]

This operation selects features based on a topological relationship with a geometric object. We have used it to select features inside the polygon having Enschede as boundary.

\[
\text{OPERA:OperationModelElement.FeatureDataOperationElement.} \\
\quad \text{FeatureDataOperation.FeatureProcessingOperation.} \\
\quad \text{FeatureSelection.SelectByTopo.SelectByWithin}
\]

Selects features based on a topological relationship with a geometric object. We need to select the activities within the city of Enschede (city boundary).

Let’s group some OPERA operations into one new concept, Get Interesting Activities, to make the expressions easier to read when doing the mappings:

\[
\text{OPERA:GetInterestingActivities} \equiv \\
\text{OPERA: SelectByThemAttrib} \sqsubseteq \\
\text{OPERA: FeatureSelectionBasedOnGeometricObjects} \sqsubseteq \\
\text{OPERA: SelectByWithin}
\]
Mapping between some concepts could be expressed as follows:

\[
\text{OPERA: GetInterestingActivities} \subseteq \text{OLS: DirectoryRequest} \\
\text{ESRI: FindNiceActivities} \subseteq \text{OLS: DirectoryRequest}
\]

Considering the explanation given in section 5.7 the basic inference that we could test on this concept expressions is subsumption, typically written as \( C \subseteq D \). Here one can check whether the concept denoted by \( D \) (the subsumer) is considered more general than the one denoted by \( C \) (the subsume). Our intuition tells us that given the very complete definitions given by OpenLS, the set of operations defined by OPERA and the ones defined by ESRI are both subsets from OpenLS. Nothing can be derived from OPERA to ESRI, since the only given fact is that each of them is subset of OpenLS.

### STEP 6. Select activities based on user preferences

*The system has to take into account Silvia’s preferences to display an ordered list of suitable activities.*

For example, in the kind of activity to do the user may have defined that she prefers going dancing than going to a concert, and that she does not like going to museums. Then the system has to take into account these preferences (stored locally) to discard museums in the list and to give an order to the list shown.

For that, ESRI needs a sequence of three geo-processing commands for finding user preferences and matching them with activities, as follows:

\[
\text{ESRI: AnalysisToolbox.ExtractToolset.TableSelect} \\
\text{ESRI: DataManagementToolBox.LayersAndTableViewsToolset.SelectLayerByAttribute} \\
\text{ESRI: DataManagementToolBox.LayersAndTableViewsToolset.Sort}
\]

Extracts selected attributes from an input table or table view and stores them in an output table. We used this command to get the user preferences (stored in a table).

Creates updates or removes the selection of features on a layer or table view using an attribute query. We used this command to select activities according to user preferences.

Orders a dataset on ascending or descending way.

Let’s group some ESRI operations into one new concept, Match User Preferences, to make the expressions easier to read when doing the mappings:

\[
\text{ESRI:MatchUserPreferences} \Rightarrow
\]
In contrast, OpenLS does not give us the functionality to query the user preferences and match them against activities.

Opera has defined a similar group of operations using feature selection geo-operations:

\[ \text{OPERA: } \text{OperationModelElement.FeatureDataOperationElement.} \]
\[ \text{FeatureDataOperation.FeatureProcessingOperation.} \]
\[ \text{FeatureSelectionBasedOnThematicAttributes.SelectByQuery} \]

The SelectByQuery operation selects features based on a query on thematic attribute types. It does not involve spatial selections. We need to find the user preferences related to evening activities preferred by the user.

\[ \text{OPERA: } \text{OperationModelElement.FeatureDataOperationElement.} \]
\[ \text{FeatureDataOperation.FeatureProcessingOperation.} \]
\[ \text{FeatureSelection.SelectByThemAttrib} \]

Selects features based on a query on thematic attribute types. We use this operation to do the match between the user preferences and the activities given in previous step.

\[ \text{OPERA: } \text{OperationModelElement.FeatureDataOperation.} \]
\[ \text{FeatureDataPresentation..Sort} \]

Sorts results. This concept was added to the original Opera ontology from Lemmens [2006].

Let’s group some OPERA operations into one new concept, Compare Preferences, to make the expressions easier to read when doing the mappings:

\[ \text{OPERA:ComparePreferences } \equiv \]
\[ \text{OPERA:SelectByQuery} \sqcup \]
\[ \text{OPERA:SelectByThemAttrib} \sqcup \]
\[ \text{OPERA:Sort} \]

Mapping between some concepts could be expressed as follows:

\[ \text{OPERA:ComparePreferences } \equiv \]
\[ \text{ESRI:MatchUserPreferences} \]

**STEP 7. Display map**

This step is similar to the one we described in step 3. Please refer to it to see the kind of operations required in each model.
6.3 Guidelines for ontology mapping

Ontology Mapping offers the potential to establish a common layer from which several ontologies could be accessed and hence could exchange information. Starting from any of the participating models in the mapping process a consumer could access the other’s operations to have a bigger set of possibilities to choose from.

Figure 6.7 outlines the interaction between the participating ontologies and the virtual one generated from the mapping process. Opera-R ontology proposed by Lemmens [2006] is taken as the initial reference ontology, and from that the corresponding operations in OpenLS and ESRI are mapped in the new Virtual Ontology. This means that the reasoning process should infer the equivalent operations between OpenLS and ESRI ontologies based on Opera-R ontology through the matching pairs identified in the virtual ontology.

![Figure 6.7. Ontology mapping strategy (Adapted from Kalfoglou[2003b])](image)

Figure 6.8 presents the main stages of the proposed method which has some adjustments from the original model presented by Kalfoglou [2003b]; these stages are: ontology harvesting, translation of acquired ontologies to RDF, matching process between concepts, and display of results. Other methods presented by authors are summarized in section 6.4.

The acquisition of ontologies is done during the ontology harvesting; they can be downloaded from different sites, received from several communities, or from libraries of ontologies. After getting the ontologies, they are translated, if it is necessary, to RDF to use a common language for all of the acquired ontologies. The case study analyzed and implemented in this exploration did not require this step; both ontologies were obtained in RDF.
Chapter 6. Results and Discussion.

The third step is related with the matching of concepts; it constitutes the core activity of the process because its product is the virtual ontology which will be used to share and exchange knowledge between ontologies. For the analyzed and implemented use cases, this step was done in a manual way, which means that the identification of operations and constraints followed an approach based on the specific knowledge of the author; nevertheless automated proposals are under research, and are based on levels of similarity between concepts.

Finally the resulted matches are stored in the virtual ontology in RDF. From this point a reasoning process can be executed to get the inferred hierarchy and identified the possible choices of a required service, which is the final expected result of this whole process.

Kalfoglou [2003b] states that the process of ontology mapping has demonstrated a set of drawbacks that prevents the definition and use of a formal method. Some of these difficulties include:

- Lack of exposure of assumptions made for the mapping and combination of ontologies.
- Lack of clear distinction between ontology mapping tools in current integrated environment for ontology editing.
- Lack of semantics in the matching process; it is more based on heuristics with syntactic clues.
- Difference in scope about the concept of mapping itself.

6.4 Ontology based discovery of geo services

Current supporting OGC compliant catalogues are enabling syntactic discoveries based on matches between data types and name. Nevertheless this approach does not provide any choice to get a semantic enrichment that allows a better selection of available services. This means that current
mechanisms to discover services are still insufficient to respond to consumer’s expectations in terms of accuracy and speed.

How can an ontology be involved in the discovery of geo services? The application of ontologies in the discovery stage of the process is expected to overcome this shortcoming through the identification and association of corresponding concepts with a semantic validation. In this scenario the user would be able to choose a concept from an ontology or to define a concept based on a shared vocabulary [Klien et al, 2004]; both concepts, selected and defined by user, would be searched in different application ontologies with the use of a reasoner to identify inferred hierarchies from mapping definitions.

The reasoner is in charge of the identification of transitive relationships between concepts and their corresponding mapping; in this sense, if a service X in local ontology A is matched with a service Y in local ontology B, and service Z in local ontology C is matched with a service Y in local ontology B, the reasoner will infer that service X in local ontology A matches with service Z in ontology C. This of course depends on which characteristics of the services are targeted.

In this way the search of services is not constraint by the search of just names or key words. A more “intelligent” search is done. Additional to the use of ontologies, their mapping provides with a broader universe of available services.

6.5 Conclusion

In this chapter results have been presented and the analysis of the followed process offers the “real taste” of ontology mapping; it is a time demanding activity that relies on knowledge of the domain and experience. This approach let a knowledge gaining process which is strengthen with the practice.

Our method can be summarized as finding correspondences among the ontologies of three different ontologies: for each concept node in one ontology, we found the most similar concept node in the other two ontologies. Although our initial idea was to find the equivalences using always the classification of geo-operations as the bridge ontology (i.e. from ESRI to Opera-R and from OLS to Opera-R), we showed that the mappings could also be expressed bridging the two other ontologies (from ESRI to OLS, from Opera-R to OLS) and (from OLS to ESRI, from Opera-R to ESRI). Any kind of mapping establishing semantic correspondences between elements and having the reference the geo-operations ontology in the expressions was rich and could give us the inference to the other two similar geo-operations in the other models.

After doing the mapping process, the OpenLS, derived from the Open Geospatial Location Services Implementation specification, could have been used as reference ontology. This finding is generated due to the selection of domain in Location Web Services for the case study. In other circumstances, where no such specification is available, the reference ontology of Geo-Operations will be the best choice.
7 Conclusions and recommendations

7.1 Conclusions

Our research explored a method for mapping similarities between concepts between source geo-ontologies using a reference ontology of geo-operations as a bridge ontology. The method proved to be far better than finding mappings between all the source ontologies, since in the real world there are simply too many ontologies available and they are too large to have manual definition of correspondences between all of them.

Mappings were defined in three levels of similarities starting from full equivalence of concepts, moving to concepts having some “kind” of similarity in the functionality (being a subset of a more general operation) and finishing with a set of interrelated concepts with some “commons rules” or sharing similar attributes/input or output parameters. For the 3 level of similarity selected for the study, we succeed in the assertions stated, since the reasoning process found new mappings derived from the given constraints. In fact the main achievement from the research was to get inferences generated by the inference engine from a set of asserted conditions.

Our study worked on the definition of correspondences among three different ontologies; however, this process could be generalized and extended to more than three ontologies. Our experiments showed that the mappings could be expressed using as bridge the two other ontologies. We concluded that in general, any kind of mapping establishing semantic correspondences between elements and having reference to the geo-operations ontology in the expressions was rich and could give us the inference to the other two similar geo-operations in the other models.

The mapping of ontologies done in chapter 6 showed that the Open Geospatial Location Services Implementation specification – OpenLS- could be used as reference ontology for problems in the domain of Location Based Services. In other circumstances, where no such specification is available, the reference ontology of Geo-Operations will be the best choice.

Finding proper expressions for the mappings was an interesting experience. We realized that having a unique set of ontologies, and giving the same task to two different experts, they may find different set of matches. This happens since there are multiple ways to model the same matching expressions and the problem of anomalies in interpreting similar models leads to a greater complexity of the semantic interoperability problem.
We faced the challenge of selecting a rich set of mappings. Definition of matching set of concepts can be done in many different ways. Thus, one is faced with the question of what interesting matches should be presented to the user. How to define which one is more appropriate? Some possible criteria for this selection could be based on measures of the expressiveness of the axioms, on the computational complexity of mappings, among others. Studying these methods and how they can be applied is an area that needs further research.

We also realize that two matching problems may be equivalent in case they have the same set of solutions. One question to solve is how to detect redundancy in a set of matches. One approach could be to generate candidate matchers and check if these candidates really solve the matching problem. Consequently, new methods for detecting similarities in the matching expressions, is a topic that needs further investigation.

From our experience we realized that the mapping process was incremental and time demanding activity that relies on a heuristic strategy; we started with a set of single assertions and went through more complex expressions. This means that the mapping activity is a “discovery” of matching operations that gain experience through a continuous interactive learning process.

Unfortunately, up to now there is a lack of formal methods to build ontologies and define mappings between them accepted by the knowledge engineering community; the use of the approach presented in chapter 6 can be considered a starting point for these activities. Noy and Kalfoglou are proposing methods and tools to speed up the process of ontology mapping, but actually their researches are independent efforts without a common acceptance between them.

We realize that some considerable work is required to understand the findings of the reasoner, especially the analyzing inferences that were not expected to be derived. Inferences join all the expressions and derive new knowledge. But when learning (step by step process) it is not desired to see all the inferences at once, it is better to see only the ones the user anticipates (the ones he/she expects to find), and only the ones which are related to the concepts just defined and being analyzed at the moment of running the reasoner. Graphic tools such as OntoViz were very useful in such process, since they show the set of asserted and inferred hierarchies, which helped a lot in the discovery of new knowledge.

Navigation in large ontologies can be very hard, when large set of concepts are defined. For example, OPERA-R contained more than 100 concepts and four to six levels of deep in the hierarchies. Establishing a clear notation that shows the path to get to the concept is key in the documentation process, as well as a neat document with the underlined semantics of the concepts defined in the ontologies.

---

13 Heuristic is the art and science of discovery and invention. (Source: www.wikipedia.com, 2006)
Ontology proposals and solutions do not just depend on technology; they also depend on people and organizations making their data and knowledge freely available by the web in such a way it can be reused by others. The initial idea of this project was to reuse an existent set of ontologies in the tourism domain, but the lack of a rich set and the difficulty to obtain them prevented us to include them in this research. The decision was to create a new ontology based on a specification. This issue addresses the necessity of reviewing current policies that restrict the sharing of data between members of a community.

The integration of ontologies from different sources presented some difficulty, especially differences in format required additional effort to generate a unique platform with diverse set of concepts. This issue was surpassed with the decision of building the set of ontologies from scratch for the research in a unique standard software platform, such as protégé.

Using any logic-based ontology language presents new users with significant problems, often made worse by details of the language and user interface. Our experiences having used Protégé as editing tool and Racer as inference engine were positive. The open world assumption and the representation of domain and range constraints as logical axioms run counter to most new users’ experience, including us, being exposed for the first time to such tools.

7.2 Recommendations and future work

The mappings we have explored so far are somehow manual, since it is required that an expert knows about the application domain in the three models to define the initial links between the concepts. Some researches [Noy, 2004] have explored semi-automatic approaches for creating the semantic mappings taking advantage of the levels of similarity between the constraints and cardinality of the concepts. This strategy is expected to speed up the mapping process and raise the level of accuracy of results.

One of the main issues that need to be addressed in later explorations of this topic is to find the way of establishing a similarity measure between two concepts. Doan et al. [2004] have proposed the use of the joint probability distribution of the concepts involved. This and other approaches must be revised and analyzed.

How to identify services with equivalent behaviour but different names? [Klien et al, 2004] exposed the problem of heterogeneities in the ontology-based discovery of services. This issue has been analyzed by [Lutz, 2004] and corresponds to a hot topic on research. Nowadays no OGC compliant is tackling these problems.

Among these future work and questions to be explored are the relationships, effects, competition, or support between ontologies, semantic web and internet 2. Both of them are promising a better, more reliable, and knowledge oriented applications, but so far, they are
considered distinct research fields. What is the best decision for a project? Or how to get
the best of both of them to a business?. Zambonini [2005] offers a small discussion about
these two worlds, and his conclusion is that we need to be aware about both of them.

A mechanism to control changes on ontologies is essential to have a stable and valid
set of mappings. Ontologies are accepted to be extended with new knowledge; however it is
not expected to have significant changes in the design and structure of them, due to the high
impact on generated mappings. An strategy to detect and come across these changes is an
issue to be explored and integrate in mapping methods.

The size of the current web is measured in terms of Peta-bytes, so people interested in
the semantic web proposal are asking how to translate this huge amount of data to a
knowledge representation with the least required effort. RDF triples seems too easy, but to
define these simple set times the current size of the web results in an effort almost
impossible to fulfil. A topic of research is about mapping knowledge tools from the current
web contents in an automatic way.
References


Bergamaschi, S, Guerra, F and Vincini, M. Critical Analysis of the emerging ontology languages and standards. 22 pag. June 9,2005


Bishr, Y, 2005. OGC’s Open Location Services Initiative & Location Interoperability Forum: Putting them together. Image Matters LLC.

Bloom, B., 1984. Taxonomy of educational objectives. Published by Allyn and Bacon, Boston, MA. Copyright (c) 1984 by Pearson Education. Available in: http://www.coun.uvic.ca/learn/program/hndouts/bloom.html


Drummond, R; Rector, A; Horridge, M; Rogers, J; Stevens, R; Wang, H; Wroe, C. 2004. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns, University of Manchester, Stanford University.


Haarslev, V; Moller, R; Wessel, M., 2005. Description Logic Inference Technology: Lessons Learned in the Trenches. Available in:


Knublauch, H.; Musen, M.; Rector, A., 2004. Editing Description Logic Ontologies with the Protégé OWL Plug-in Stanford University, CA, University of Manchester, UK


Lemmens, R.; De Vries, M., 2004. Semantic description of location based web services using an extensible location ontology. In IfGI prints, editor, Proceedings of the Munster GI-days, Geoinformation and Mobility, Germany, number 22, July 1-2 2004


