Semantic Annotation of RESTful and WFS OGC services

Author: Víctor Saquicela Galarza
Supervisor: Oscar Corcho and Luis Vilches

Facultad de Informática
Universidad Politécnica de Madrid

A thesis submitted for the degree of

PhD

2015
Tribunal nombrado por el Sr. Rector Magnífico de la Universidad Politécnica de Madrid, el día ........ de .......... de 20..

1. Presidente:

2. Vocal:

3. Vocal:

4. Vocal:

5. Secretario:

6. Suplente:

7. Suplente:

Realizado el acto de defensa y lectura de la Tesis el día........ de........ de 20.. en la E.T.S.I/Facultad..................

Calificaciom:
EL PRESIDENTE
LOS VOCALES
EL SECRETARIO
Abstract

The Web contains an immense collection of Web services (WS-*, RESTful, OGC WFS), normally exposed through standards that tell us how to locate and invoke them. These services are usually described using mostly textual information and without proper formal descriptions, that is, existing service descriptions mostly stay on a syntactic level. If we want to make such services potentially easier to understand and use, we may want to annotate them formally, by means of descriptive metadata.

The objective of this thesis is to propose an approach for the semantic annotation of services in the geospatial domain. Our approach automates some stages of the annotation process, by using a combination of third-party resources and services. It has been successfully evaluated with a set of geospatial services.

The main contribution of this work is the partial automation of the process of RESTful and WFS semantic annotation services, what improves the current state of the art in this area. The more detailed list of contributions are:

- A model for representing Web services.
- A method for annotating Web services using ontological and external resources.
- A system that implements the proposed annotation process.
- A gold standard for the semantic annotation of RESTful and OGC WFS services, and algorithms for evaluating the annotations.
Abstract

Actualmente, la Web provee un inmenso conjunto de servicios (WS-*, RESTful, OGC WFS), los cuales están normalmente expuestos a través de diferentes estándares que permiten localizar e invocar a estos servicios. Estos servicios están, generalmente, descritos utilizando información textual, sin una descripción formal, es decir, la descripción de los servicios es únicamente sintáctica. Para facilitar el uso y entendimiento de estos servicios, es necesario anotarlos de manera formal a través de la descripción de los metadatos.

El objetivo de esta tesis es proponer un enfoque para la anotación semántica de servicios Web en el dominio geoespacial. Este enfoque permite automatizar algunas de las etapas del proceso de anotación, mediante el uso combinado de recursos ontológicos y servicios externos. Este proceso ha sido evaluado satisfactoriamente con un conjunto de servicios en el dominio geoespacial.

La contribución principal de este trabajo es la automatización parcial del proceso de anotación semántica de los servicios RESTful y WFS, lo cual mejora el estado del arte en esta área. Una lista detallada de las contribuciones son:

- Un modelo para representar servicios Web desde el punto de vista sintáctico y semántico, teniendo en cuenta el esquema y las instancias.

- Un método para anotar servicios Web utilizando ontologías y recursos externos.

- Un sistema que implementa el proceso de anotación propuesto.

- Un banco de pruebas para la anotación semántica de servicios RESTful y OGC WFS.
To my family, especially to my daughter Vicky who will take care of my family forever.
Acknowledgements

Quiero empezar agradeciendo a toda la gente que ayudó todo este proceso de formación y finalización de esta tesis doctoral. Expreso mi más sincera gratitud a mi director Oscar Corcho por su: infinita paciencia, amenazas, estímulos, etc. a través de mis estudios de PhD en el grupo de ingeniería ontológica en la Universidad Politécnica de Madrid.

También quiero agradecer a Luis Vilches por su incondicional soporte en los momentos más difíciles en el trayecto de esta tesis doctoral.

Mi agradecimiento a todos los compañeros de trabajo de OEG quienes me proveyeron de un ambiente óptimo de trabajo.

Un agradecimiento especial a Boris Villazón por esas largas conversaciones acerca de este trabajo.

Para mi familia, un agradecimiento especial por acompañarme en todo este viaje, realmente un viaje complejo desde la salida de Ecuador a España, así como el regreso y la feliz culminación de este trabajo. Largo viaje que me ha dejado los más valioso de mi vida, mis dos hijos Sebastián y Camilo. Aunque la inspiración más grande es mi hija que desde algún lado siempre cuida de mi familia.

A mi mujer por aguantar todo este tiempo, apoyar y recordar que lo que se empieza hay que terminar.

No puedo dejar un agradecimiento especial a la gente del fútbol, esos minutos que hacían que la tesis pase a segundo plano.

Finalmente, gracias Asun por esa confianza al aceptar que me integre al OEG y por esos días del jamón que están marcados en este trabajo.
# Contents

List of Figures

List of Tables

Glossary

1 Introduction

1.1 Thesis Structure ......................................... 4
1.2 Publications and Reports ............................... 5

2 Background and State of the Art .......................... 9

2.1 Annotations .............................................. 9
2.2 Web Services and their Descriptions .................. 12

2.2.1 Web Service(WS-*) .................................. 14
2.2.1.1 Finding WS-* .................................... 16
2.2.1.2 Limitations of WS-* ............................. 17

2.2.2 RESTful Services (Web APIs) ....................... 18
2.2.2.1 Finding RESTful services ....................... 20
2.2.2.2 Types of RESTful services ..................... 22
2.2.2.3 Syntactic Web Service Descriptions Languages for RESTful Services ................................. 24

2.3 Semantic Annotations of Services on the Web ......... 28

2.3.1 Semantic WS-* ....................................... 29
2.3.1.1 OWL-S ............................................ 30
2.3.1.2 WSMO ............................................. 30
2.3.1.3 SAWSDL .......................................... 30
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1.4 SWS WS-* Limitations</td>
<td>31</td>
</tr>
<tr>
<td>2.3.2 Semantic RESTful Services</td>
<td>31</td>
</tr>
<tr>
<td>2.3.2.1 Semantic Annotation Approaches</td>
<td>32</td>
</tr>
<tr>
<td>2.3.2.2 RESTful Semantic Annotations Tools</td>
<td>37</td>
</tr>
<tr>
<td>2.3.2.3 Limitations of Semantic RESTful Annotation</td>
<td>37</td>
</tr>
<tr>
<td>2.4 Geospatial Web Services</td>
<td>39</td>
</tr>
<tr>
<td>2.4.1 OGC and ISO/TC211</td>
<td>40</td>
</tr>
<tr>
<td>2.4.1.1 Web Features Services</td>
<td>42</td>
</tr>
<tr>
<td>2.4.1.2 Syntactic Descriptions</td>
<td>43</td>
</tr>
<tr>
<td>2.4.1.3 Semantic Annotations</td>
<td>43</td>
</tr>
<tr>
<td>2.4.2 Semantic Resources: Ontologies and SPARQL endpoints</td>
<td>47</td>
</tr>
<tr>
<td>2.4.2.1 Scheme Annotation</td>
<td>48</td>
</tr>
<tr>
<td>2.4.2.2 General Resources Annotation</td>
<td>50</td>
</tr>
<tr>
<td>2.4.2.3 Geospatial Domain Ontologies</td>
<td>51</td>
</tr>
<tr>
<td>2.4.3 Tools</td>
<td>52</td>
</tr>
<tr>
<td>2.4.4 Limitations of the Current State of the Art on Syntactic Descrip-</td>
<td>52</td>
</tr>
<tr>
<td>tions and Semantic Annotations</td>
<td></td>
</tr>
<tr>
<td>3 Objectives and Contributions</td>
<td>55</td>
</tr>
<tr>
<td>3.1 Thesis Objectives</td>
<td>56</td>
</tr>
<tr>
<td>3.2 Contributions to the State of the Art</td>
<td>57</td>
</tr>
<tr>
<td>3.3 Assumptions</td>
<td>58</td>
</tr>
<tr>
<td>3.4 Hypotheses</td>
<td>58</td>
</tr>
<tr>
<td>3.5 Restrictions</td>
<td>59</td>
</tr>
<tr>
<td>4 A General Model and Method for the Semantic Annotation of Web Servic-</td>
<td>61</td>
</tr>
<tr>
<td>es</td>
<td></td>
</tr>
<tr>
<td>4.1 Preliminaries</td>
<td>62</td>
</tr>
<tr>
<td>4.1.1 Basic definitions</td>
<td>62</td>
</tr>
<tr>
<td>4.2 Definitions - Problem formulation</td>
<td>62</td>
</tr>
<tr>
<td>4.2.1 Service URLs</td>
<td>62</td>
</tr>
<tr>
<td>4.2.2 Service Description</td>
<td>63</td>
</tr>
<tr>
<td>4.2.3 Ontology and SPARQL</td>
<td>64</td>
</tr>
<tr>
<td>4.2.4 External Terminological Resources</td>
<td>65</td>
</tr>
</tbody>
</table>
### CONTENTS

5.3 Semantic Annotation Process ........................................... 97
  5.3.1 A model for describing RESTful service parameters semantically 98
    5.3.1.1 Ontological Resources ...................................... 100
    5.3.1.2 Using ontological sources in the annotation process .... 103
    5.3.1.3 Use of External Resources for enriching the semantic annotations ...................................... 105
    5.3.1.4 Using external resources in the annotation process .... 107
  5.3.2 Definition of Semantic RESTful Annotation ....................... 107
    5.3.2.1 Possible semantic definition of RESTful services .... 109
  5.3.3 Checking the semantic annotation of RESTful services .......... 112
    5.3.3.1 SA-REST .................................................. 115
  5.4 Conclusions .......................................................... 115

6 OGC Web Feature Service Annotation Process ........................ 117
  6.1 An automated approach for the semantic annotation of WFS ....... 118
  6.2 Syntactic description of WFS services: Registering invocation details in a repository ............................................. 119
    6.2.1 GetCapabilities .............................................. 119
    6.2.2 DescribeFeatureType ......................................... 124
    6.2.3 GetFeature .................................................. 126
    6.2.4 Implementation of invocation and syntactic registration .... 129
  6.3 Semantic Annotation Process ........................................ 133
    6.3.1 A model for describing semantically WFS service parameters .. 134
    6.3.2 Annotation of WFS service capabilities ...................... 135
    6.3.3 DescribeFeatureType Annotations ................................ 138
    6.3.4 Enriching with External Resources and annotating .......... 139
    6.3.5 Using GetFeature to validate the DescribeFeatureType Annotations .................................................. 139
      6.3.5.1 Definition of Semantic WFS Annotation ................... 139
      6.3.5.2 Possible semantic definition of WFS services .......... 141
    6.3.6 Checking the semantic annotation of WFS services .......... 141
  6.4 Conclusions .......................................................... 141
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D  Procedure-Oriented Service Model (POSM)</td>
<td>207</td>
</tr>
<tr>
<td>E  Glossary of Terms</td>
<td>209</td>
</tr>
<tr>
<td>E.1 Preliminaries Background Technologies</td>
<td>209</td>
</tr>
<tr>
<td>E.1.1 XML</td>
<td>209</td>
</tr>
<tr>
<td>E.1.2 RDF, RDF schema, OWL</td>
<td>209</td>
</tr>
<tr>
<td>E.1.3 SPARQL</td>
<td>209</td>
</tr>
<tr>
<td>E.1.4 Ontologies</td>
<td>210</td>
</tr>
<tr>
<td>E.1.5 SDO</td>
<td>210</td>
</tr>
<tr>
<td>E.1.6 REST APIs</td>
<td>210</td>
</tr>
<tr>
<td>F  RESTful and WFS Services</td>
<td>211</td>
</tr>
<tr>
<td>References</td>
<td>217</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Generic Annotation Model ............................................. 11
2.2 Web of Services .......................................................... 12
2.3 Service Oriented Architecture ........................................ 13
2.4 RESTful Services ......................................................... 19
2.5 Web service ............................................................... 20
2.6 Documentation ............................................................ 21
2.7 hREST ................................................................. 31
2.8 Relative positioning of WSMO-Lite and MicroWSMO ............. 35
2.9 Levels of annotations, copied from (MSD09) ....................... 37
3.1 Thesis ................................................................. 55
4.1 Model ................................................................. 67
4.2 Service ................................................................. 70
4.3 External Resources ..................................................... 73
4.4 MSM extended .......................................................... 82
5.1 RESTful services ......................................................... 84
5.2 RESTful ................................................................. 94
5.3 HTML ............................................................... 96
5.4 RESTful parameters .................................................... 98
5.5 Model Parameter ....................................................... 99
5.6 Cache model ............................................................ 107
6.2 WFS ............................................................... 131
6.3 WFS ............................................................... 133
LIST OF FIGURES

6.5  Model Parameter ......................................................... 135
6.6  Representation .......................................................... 136
List of Tables

3.1 Mapping between objectives and contributions with associated assumptions, hypotheses and restrictions ............................................. 60
4.1 Web service characteristics ................................................. 72
4.2 External Resources ....................................................... 73
4.3 Models comparison ....................................................... 81
5.1 SPARQL query result ..................................................... 104
5.2 SPARQL query results .................................................... 104
5.3 SPARQL query result ..................................................... 105
7.1 Equality procedures used in the experiments ......................... 145
7.2 Equality procedures used in the experiments ......................... 146
7.3 Enriching initial parameters with external resources ............... 147
7.4 Results of the input and output parameters .......................... 147
7.5 Output parameters metrics ............................................. 149
7.6 Enriching initial parameters with external resources ............... 151
7.7 Output parameters metrics ............................................. 153
Glossary

BPEL  Business Process Execution Language
CSW   Catalogue Service for the Web
GIS   Geographic Information Systems
GML   Geography Markup Language
HTTP  Hyper Text Transport Protocol
ISO   International Organization for Standardization
JSON  JavaScript Object Notation
N3    Notation3
OGC   Open Geospatial Consortium
OWL   Web Ontology Language
OWL-S Semantic Markup for Web Services
OWS   Open Geospatial Consortium Web Services
POSM  Procedure-Oriented Service Model
RDF   Resource Description Framework
RDFs  RFD schema
REST  REpresentational State Transfer
SAWSDL Semantic annotation for WSDL and XML Schema
SDI   Spatial Data Infrastructure
SOA   Service-Oriented Architecture
SOAP  Simple Object Access Protocol
SPARQL SPARQL Protocol and RDF Query Language
SWS   semantic Web Services
URI   Uniform Resource Identifier
URL   Uniform Resource Locator
W3C   World Wide Web Consortium
WCS   Web Convering Service
WFS   Web Feature Service
WGS 84 world Geodetic System 1984
WSDL  Web Service Description Language
WSMO  Web Service Modeling Ontology
WSMO-Lite Lightweight semantic Description for Services on the Web
WWW   World Wide Web
XML   Extensible Markup Language
1

Introduction

The Web contains an immense collection of resources, normally exposed through standards that tell us how to locate and transport data. Such resources might be textual information on a Web site but can also be images, videos, Web services, etc. Without proper descriptions, the use of such resources may be limited to small user groups. Hence, before publishing a resource on the Web, it should be annotated with as much descriptive metadata as possible to make it easier to use by a wider audience. Otherwise people will neither be able to find the resource easily using search engines nor will they be able to evaluate easily if the discovered resource satisfies their current information needs.

Our work focuses on a specific type of resources: Web services. Web services are considered as self-contained, self-describing, modular applications that can be published, located and invoked across the Web (RS05). Two of the main tasks are search and composition. One of the main reasons why service search and composition is complex is that existing service descriptions mostly stay on a syntactic level. This limits Web service usage to manual discovery (SF10). Clients can figure out which operations are provided by a service and which kinds of data types are used, but determining the meaning of an operation is much more difficult. The functionality of a service or operation has to be discovered by manual intervention due to the fact that existing service descriptions do no encompass information about the provided features in a machine-processable manner. This is often provided in an informal and unstructured manner and thus, not machine-processable. Therefore, only insufficient tool support is available due to the fact that only low-level information retrieval mechanisms can
be applied. For instance, a keyword-based search within an informal and text-based service description often results in a low accuracy and hence, requires a lot of manual inspection to determine the desired service.

Ideally, a service discovery mechanism should enable the location of services by both humans and machines. Humans can be either end-users looking for a service to use it as it is or developers who want to find a service at design-time and incorporate it in their program. However, discovery is still one of the major challenges of Web service technology ([TPBC02]). An effective and automated search and selection of relevant services is essential both for human users (developers or non-technical persons) and programs (such as software agents).

In the early 2000s, a prominent research field addressing this challenge was that of Semantic Web Services (SWS) ([MSZ01]). These try to overcome the deficits of a purely syntactical description by creating semantically-enriched service descriptions. Some of the obtained results were languages for describing services and techniques for discovery and composition. However, these languages and techniques have had limited impact so far.

In recent years, since the advent of Web 2.0 applications and given some of the limitations of classical Web services based on SOAP and WSDL, Representational State Transfer (REST) services have increased their presence on the Web ([PZL08]). Machine-oriented Web applications and APIs that are conformant to the REST architectural style ([Fie00]), normally referred to as RESTful Web services, have started appearing, mainly due to their relative simplicity and their natural suitability for the Web. Many of these services are generated by users and not by service engineers. For example, mashups are introduced as a new single simple way of composing services and combining content from different sources.

However, using RESTful services still requires much human intervention, since they usually lack standardized descriptions from providers, as there are neither standardized ways to describe services nor public repositories to store those descriptions. As a result, the majority of their descriptions are given in the form of unstructured text in Web pages (HTML), which contain lists of available operations, their URIs and parameters (also called attributes), expected output, error messages, and some examples of their execution. This hampers the automatic discovery, interpretation and invocation of
these services, what may be helpful to improve the efficiency in the development of applications that are based on their use.

Likewise, the Geospatial Web is a rich source of geospatial data which remains hidden for users not aware of the geospatial services defined by the Open Geospatial Consortium (OGC). The OGC standards baseline provides accepted and well thought-out methods to make spatial resources (data and processes) served via Web services (MSD09). Service capabilities describe, besides contextual information like contact information, how to access and invoke the service to retrieve the required geospatial data. The individual name and location of the operations are also listed in the Capabilities document of each OGC-conformant Web Service (as defined in OGC WS-Common). Since such operations and the format to encode the data are predefined in OGC Implementation Standards and OGC Encoding Standards, generic clients can, without knowledge about the nature of the data, display the resulting data on a map.

Despite the benefits that the Geospatial Web provides, several open issues have to be discussed. It is possible to request geospatial data served by an OGC Web Service (OWS) via unique URLs, for example vector data offered by a Web Feature Service (WFS). WFSs model vector data as a feature collection. Features, such as as buildings, streets, or rivers, are entities with a spatial location. Individual features included in this data set cannot be accessed by people or clients via URIs. There is also a lack of links to related data entities. Our claim is that attaching semantic descriptions to the data helps to infer what they represent in reality and facilitates the task of finding them.

In summary, semantic annotation approaches for services (WSMO, OWL-S, SAWSDL, etc.) have focused on defining service description formalisms, and have been normally applied to WS-* services and their corresponding middleware. More recently, these (usually heavyweight) formalisms have started to be adapted into more lightweight approaches for the semantic description of RESTful services (MKP09) [KGV08]. One of the reasons for making these annotations more lightweight is to increase their uptake, promoting the creation of such annotations by service developers and publishers, while still providing some added value to the syntactic descriptions currently available for both WS-* and REST approaches. Nevertheless, most of the processes related to the annotation of such services still require a large amount of human intervention. In this thesis we mainly focus on RESTful service annotation, for which some approaches
already exist (e.g., [MGPD09 AMT09]) and in OGC WFS services for which there is little work done in the state of the art. In these approaches, humans firstly need to understand the informal descriptions provided in the RESTful service description pages, and then the semantic annotation of these services is done mostly manually, with or without tool assistance. In this thesis, we describe our approach to address the challenge of automating the semantic annotation of these services by: (1) obtaining and formalising their syntactic descriptions, what allows their registration and invocation, and (2) interpreting and semantically enriching their parameters, what allows generating semantically-enriched service descriptions in several of the available formalisms for semantic service description.

The main contribution of our work is the partial automation of the process of annotating semantically RESTful and WFS services, what improves the current state of the art in this area. We have defined a process to perform this type of annotation, and we propose the usage of diverse types of external resources and services to help during this process: a cross-domain knowledge-base like DBpedia, two domain ontologies like GeoNames and the WGS84 vocabulary (since we are focused on RESTful services on the geospatial domain), and suggestion and synonym services.

1.1 Thesis Structure

The rest of the thesis is organized as follows:

• Chapter 2 - Background and State of the Art, provides a survey of the background, context and relevant work for this research. It describes basic concepts, theory and technology for the syntactic description and semantic annotation of Web services. Furthermore, we discuss a number of existing syntactic modeling languages and semantic annotation processes.

• Chapter 3 - Objectives and Contributions presents the objectives and contributions of the thesis. We also present the contributions to the state of the art, assumption and hypotheses.

• Chapter 4 - A generalised Method for the Semantic Annotation of Services on the Web introduces our basic semantic annotation process consisting of finding services, automatic invocation, external resource usage, and model annotation.
1.2 Publications and Reports

- Chapter 5 - RESTful Annotation Process, we discuss the problem of syntactic registration and semantic annotation of RESTful services using the following resources: a cross-domain ontology like DBpedia, two domain ontologies like GeoNames and the WGS84 vocabulary, and additional external resources like suggestion and synonym services.

- Chapter 6 - Web Feature Service Annotation Process, we present the outline of a novel semantic annotation process for OGC Web Services, concretely Web Feature Services using the following resources: cross-domain ontologies like DBpedia and three domain ontologies like GeoNames ontology, WGS84 vocabulary and WFS ontology.

- Chapter 7 - Evaluation, we describe experiments related to the semantic annotation of RESTful and WFS services. We also include experiments covering all the process proposed in Chapter 4. Furthermore, we present a gold standard for validating the experiments, which can be reused by other researchers working on this area.

- Chapter 8 - Conclusions and Future Work summarizes the work, and outlines the future work to point out the possible improvements and the interesting directions of further research.

1.2 Publications and Reports

The contributions produced in this thesis have been published in international peer-reviewed workshops, conferences and journals.


1. INTRODUCTION


- Saquicela, Victor; Espinoza-Mejia, Mauricio; Palacio, Kenneth; Alban, Humberto; Enriching Electronic Program Guides using semantic technologies and external resources,Computing Conference (CLEI), 2014 XL Latin American, 1-8, 2014, IEEE

In addition to the development of this thesis work, the author has produced some publications, which even though not being strictly related to the contributions of this thesis.

- Saquicela, Victor; Espinoza, Mauricio; Piedra, Nelson; Terrazas, Boris Villazón; Ecuadorian Geospatial Linked Data, w3c, 2014

- Vilches-Blázquez, Luis M; Villazón-Terrazas, Boris; Saquicela, Victor; de León, Alexander; Corcho, Oscar; Gómez-Pérez, Asunción; GeoLinked data and INSPIRE through an application case Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, 446-449, 2010, ACM

- de León, Alexander; Saquicela, Victor; Vilches, Luis M; Villazón-Terrazas, Boris; Priyatna, Freddy; Corcho, Oscar; Geographical linked data: a Spanish use case, Proceedings of the 6th International Conference on Semantic Systems, 36, 2010, ACM
1.2 Publications and Reports

- Vilches-Blázquez, Luis M; Saquicela, Víctor; Corcho, Oscar; Interlinking Geospatial Information in the Web of Data, Bridging the Geographic Information Sciences, 119-139, 2012, Springer Berlin Heidelberg

- Saquicela, Víctor; Espinoza, Mauricio; Piedra, Nelson; Terrazas, Boris Villazón; Ecuadorian Geospatial Linked Data, W3C, 2014

- Siguenza Guzman, Lorena; Saquicela, Víctor; Cattrysse, Dirk; Design of an integrated Decision Support System for library holistic evaluation, Proceedings of IATUL Conferences, 1-12, 2014

- Saquicela, Víctor; Bermeo, Jorge; Espinoza, Mauricio; Villazón, Boris; Identifying Common Research Areas: A Study Case, 2014
1. INTRODUCTION
2

Background and State of the Art

In this chapter, we present some background information about the syntactic description and semantic annotation of services on the Web, with the purpose of introducing this topic, describing the current state of the art, and identifying limitations to it.

2.1 Annotations

Annotation is usually defined as data about data, which aims at expressing the semantics of information, hence improving information seeking, retrieval, understanding and use (Cor06). Annotation can be attached to a wide range of entities, including not only documents, but also services available in the web.

In (OMS+06), the author distinguishes between three types of annotations: informal (e.g. Web 2.0, tags, comments in natural language, etc.), formal, and ontological. Informal annotations lack machine-readability because they do not use a formal language. Formal annotations are machine-understandable, but do not necessarily use ontological terms. In ontological annotations the terminology has a commonly understood meaning that corresponds to a shared conceptualization, that is an ontology. In this work we use ontological annotations for semantic annotations of Web services.

The term Semantic Annotation is described as "the action and result of describing (part of) an electronic resource by means of metadata whose meaning is formally specified in a ontology" (electronic resource can be text contents, images, video, services, etc) by (LLPB11). Talantikite, et al. (TAB09) introduce it as "An annotation
2. BACKGROUND AND STATE OF THE ART 

assigns to an entity, which is in the text, a link to its semantic description. A semantic annotation is referent to an ontology”.

Ontologies appeared first as the backbone of document metadata annotation in preSemantic Web applications. With the emergence of the Semantic Web, ontology based document and service annotation has been the focus of many projects and applications, since the availability of annotated content is one of the key challenges to overcome in order to make the semantic Web a reality \cite{BCCGp02}. We will now describe different dimensions applicable to ontology-based annotations, with special focus on the annotation of services.

If we consider the characteristics of the annotation process we can distinguish the following types of annotations:

- **Manual Annotation.** In a manual annotation process, users are responsible of analyzing Web services, trying to identify their main characteristics and relating them with some vocabulary (already existing or generated in an ad-hoc manner). Manual annotation is a completely human-oriented task. Manual annotation is tedious and quite often error prone given large ontologies with thousands of concepts and millions of instances involved. The advantage of manual annotation is normally the high accuracy in extracting semantic information at several levels. It is the most precise way of annotation and for now, the only way of exploiting the full value of adding semantics to Web services. The main disadvantage of this annotation process is the cost of producing annotations.

Manual annotation research focused more on annotation representation, sharing and storage mechanisms as well as friendly user-interfaces to help people write down their notes.

- **Semi-automatic Annotation.** Semi-automatic annotation is a type of manual annotation aided by automatic extraction of some information and automatic suggestions. Likewise, users can be requested to add additional information, that is, there is little human intervention.

- **Automatic Annotation.** The automatic way does not require human intervention and the users only verify resulting descriptions. Research about automated
2.1 Annotations

![Figure 2.1: Generic Annotation Model](http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html)

annotation tools focuses more on the ways of creating annotations according to specified domain ontologies.

There are many types of annotation models available in the state of the art. In the semantic web community, these models have been abstracted by the Subject-Predicate-Object triple that can be used for most of the annotation kinds discussed here (Figure 2.1). This model is based on the notion of semantic annotation ([KPT+04]). It describes both the process and the resulting annotation or metadata consisting of aligning a resource or a part of it with a description of some of its properties and characteristics with respect to a formal conceptual model or ontology. Ontology based annotations describe a resource with respect to a formal conceptual model, allowing links between structured and unstructured data. This allows a whole new range of retrieval techniques, which can be based on the knowledge schema expressed in the ontology.

([AZP10]) describes the issue of semantic annotation and reviews a number of research and end-users tools in the field. This work provides a clear classification scheme of the features of annotation systems based on three criteria: i) the structural complexity; ii) the vocabulary type which describe the vocabulary used for annotation; and iii) the user collaboration which users contribute to create different types of annotations.

---

2. BACKGROUND AND STATE OF THE ART

2.2 Web Services and their Descriptions

In this section, we discuss some related efforts on how to add semantics to web services. We also look into some semantic Web service description languages.

A Web service is a method of communication between two electronic devices over the web (Internet) (Fig. 2.2). The W3C defines a Web service as a software system designed to support interoperability in machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using messages, typically transmitted using HTTP with an XML serialization in conjunction with other Web-related standards.

In essence, a Web service is a modular, self-describing, and self-contained software application which is discoverable and accessible through standard interfaces over the network. Web service technology allows for uniform access via Web standards to software components residing on various platforms and written in different programming languages.

At a technology level, the community distinguishes two types of Web services: classical Web services based on WSDL/SOAP (big Web services, WS-*), and Web APIs.

---

3http://www.w3.org/TR/ws-arch/
4http://en.wikipedia.org/wiki/Web_service
2.2 Web Services and their Descriptions

RESTful. The first have defined a stack of standards and the second are characterized for the simplicity.

- SOAP relies on a comprehensive stack of technology standards. SOAP plays a major role in the interoperability within and among enterprises and serves as the basic construct for the rapid development of low-cost and easy-to-compose distributed applications in heterogeneous environments (PTDL08).

- APIs are characterized by their relative simplicity and their suitability for the Web. Web APIs, which conform to the REST paradigm (Fie00), are commonly referred to as RESTful services. RESTful services are centered around resources, which are interconnected by hyperlinks and grouped into collections, whose retrieval and manipulation is enabled through a fixed set of operations commonly implemented by using HTTP. RESTful are more lightweight in their technological stack.

A Service Oriented Architecture (SOA) involves three types of actors: service providers, service consumers and service registries also called discovery agencies. The interaction within a SOA are illustrated in Figure 2.3.

- Service Provider. Creates service descriptions which are published in the registry. The provider also receives request messages from the consumer.
2. BACKGROUND AND STATE OF THE ART

- Service Consumer. Request for available services as given by the service description in the registry. Once the consumer finds the required services, interactions with the provider are established in order to start using the services.

- Discovery Agency. Provides the service descriptions published by the service provider. The agency can be localized or distributed. After the services have been published, service requestors can search for services within the registry. The discovery agency plays the role of a mediator between service providers and requestors.

A prerequisite to reusing and composing Web services is the ability to find the right service for a given task. However, Web services discovery is problematic with the increased number of both WSDL and Web APIs (Fig. 2.4).

In the following subsections, we describe in detail the fundamental concepts of WS-* and RESTful services.

2.2.1 Web Service(WS-*)

WS-* service (also know as Big Web services ([PZL08])) are based on a stack of WS-* standards. One of their most important characteristics is that they use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Describing WS-* services has been an issue of research for almost ten years. Web services are normally designed for machine consumption, and someone has to tell machines which particular services to use for a specific purpose. Web services are normally perceived as too complex and heavyweight. Such services often contain a machine-readable description of the operations offered by the service, written in the Web Service Description Language (WSDL), an XML language for defining interfaces syntactically.

The Web service architecture is composed of three functional components, which are transport, description and discovery. These are implemented using SOAP, WSDL, and UDDI respectively.

- SOAP. The Simple Object Access Protocol (SOAP) is an XML-based protocol for transfer of structured data and type information across a network in a stateless manner. Web services use SOAP for communication between WS registries,
2.2 Web Services and their Descriptions

remote WSs and client applications. A SOAP based design must include the following elements:

– A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract providing a machine-processable description of the structure of the service, which may include request and response messages, operations, bindings, and the location of the Web service.

– The architecture must address complex non functional requirements. Many Web services specifications address such requirements and establish a common vocabulary for them. Examples include, transaction, security, addressing, trust, coordination and so on.

– The architecture needs to handle asynchronous processing and invocation.

• WSDL. The Web Service Definition Language (WSDL) is an XML based language used to describe WSs and how to locate them. WSDL is a widely accepted standard for describing Web service interfaces. WSDL gives details of how communication with a remote WS is done. Using a standard XML schema, it describes how to interpret the messages, how to contact the WS and the protocols to use.

• UDDI. The Universal Data Description and Integration (UDDI) is the global look up for locating services. The standard provides an information repository and query service for WSs. UDDI is a domain-independent standard method allowing publishing and discovering information about WSs.

Applications take advantage of Web services through the publication of service descriptions, finding, retrieval and composition of service descriptions, and binding or invoking of services based on the service description. Three operations involved in SOA are:

• Publish. The publish operation is used by service providers to register data and services to a discovery agency. A service provider contacts the service directory

---

6www.w3c.org
7http://www.w3.org/TR/wsdl20/
8http://www.w3.org/TR/ws-arch/
to publish or unpublish a service. A service provider typically publishes service metadata describing its capabilities and network address called, the URL.

- **Find.** The find operation is used by service consumers to discover specific service types or instances. Service consumers describe requests for services from the registry and the registry responds by delivering the result that matches the request. Service consumers typically use metadata published to find services of interest.

- **Bind/Interact.** Used when a service consumer invokes a service. A service consumer uses service metadata provided by the registry to bind to a service provider.

### 2.2.1.1 Finding WS-*

The landscape of this type of Web service repositories has changed substantially in the last years. Many repositories have been discontinued or stag for example: UDDI, .NET XML Web service repository, SalCentral, WebserviceX.NET, Xmethods, etc.

We show here a list of repositories, technologies and its evolution.

- **UDDI,** Universal Descriptions, discovery and Integration is a platform-independent, Extensible Markup Language-base registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.

- **WebserviceX.NET** provides “programmable business logic components and standing data that serve as black boxes to provide access to functionality and data via web services”.

- **Xmethods** is one of the largest Web service repositories, containing several hundred services. However, this site only provides a list of services and no support for browsing or search facilities of any kind.

- **BindingPoint** is an independent resource for locating and consuming XML web services that allows for keyword search on web services as well as category browsing. It provides business and technical information on all web services published

---

9 www.webservicex.net
10 www.xmethods.com
11 www.bindingpoint.com
in the public UDDI Business Registry, as well as ratings, reviews, and general support for these web services.

- SalCentral\footnote{www.salcentral.com} is a centralised brokerage that can help customers find web services and help web services providers advertise their functionality using specialist searching facilities and quality assurance checking.

- Seekda Services provides a common interface for registering, browsing and annotating Web services. It lists more than 27000 services

- Service-Finder provides an efficient access to publicly available services. Service technology can be used to expose functionalities in many domains. It is a platform for service discovery where service related information from heterogeneous sources is automatically integrated in a coherent semantic model to allow effective discovery.

- http://webservicelist.com/, a catalogue of around a thousand web services classified in about 20 categories.

- wsfinder.jot.com, a Wiki for Finding Web Services and Open APIs (233 APIs organized in a 47 categories).

### 2.2.1.2 Limitations of WS-*

UDDI is an industry effort to provide directory services for Web services offered by businesses. It allows businesses to publish their services in a directory and enable other business entities to locate partners and to form business relationships based on the Web services they provide. UDDI provides a set of search facilities for finding businesses, and their services. Services can be searched by specifying business name, service name, service category or models. However, UDDI in its current form is limited in its search services by its inability to extend beyond the keyword-based matches. Additionally, WSDL still require substantial manual effort related to locating, interpreting and integrating existing services. The main reason for this is that the semantics of these services and of the data they manipulate remain implicit. WSDL is an XML grammar for specifying properties of a Web service such as what it does, where it is located and
2. BACKGROUND AND STATE OF THE ART

how it is invoked, i.e., it describes only the functional and syntactic aspects of a service. WSDL does not support non-functional information of services.

2.2.2 RESTful Services (Web APIs)

The Representational State Transfer (REST) is an architecture style for applications within distributed environments and especially applies to Web Services. The creation of the adjective RESTful accomplished a simple ability to express if something works accordingly to the principles of REST [Fie00]. REST is an architecture style rather than a concrete architecture. RESTful APIs are characterized by resource-representation decoupling, so that resource content can be accessed via different formats.

The REST architectural style proposes a uniform interface, that if applied to a Web service induce desirable properties, such as performance, scalability, and modifiability, what enables Web services to work better on the Web. In REST architectural style, data and functionality are considered resources. These resources are accessed using Uniform Resource Identifiers (URI), and are acted upon by using a set of simple, well-defined operations.

The following encourage RESTful application to be simple, lightweight and fast:

- Resource identification through URI. A RESTful Web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource discovery.

- Uniform interface. Resources are manipulated using a fixed set of four operations (create, read, update, delete): PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.

- Self-descriptive messages. Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, RDF, JSON, and others. Metadata about the resource is available and used, for example to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.
2.2 Web Services and their Descriptions

- Stateful interaction through links. Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interaction are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be also embedded in response messages to point to valid future states of the interaction.

The RESTful service implementation approach has gained increased attention over the recent years. Figure 2.4 depicts the result of a usage study that shows that in 2010 74% of all Web services were RESTful. Despite this huge success, RESTful does not provide a standardized and accepted description language, which hampers its application within business related scenarios.

![Figure 2.4: RESTful Services - usage study from programmableweb.com](image)

Therefore, it can be considered that REST is better suited for basic, ad hoc integration scenarios. RESTful are based on standard Web technologies, such as HTTP, URI, and XML. RESTful services have been adopted by major companies, such as Google, Yahoo, Facebook, etc. Unlike SOAP-based web services, which have a standard vocabulary to describe the web service interface through WSDL, RESTful web services do not have currently such a grammar or standard (PZL08). However, RESTful services are often described with HTML pages that describe for users what the service does and how to invoke it. This pages have all the necessary details for a human to be able to create an application that uses the service. This HTML is in a way the equivalent to a WSDL for RESTful services. The problem, however, with treating an HTML as a service description is that HTML is meant to be human readable whereas languages like WSDL were designed to be machine readable. HTML pages are intended for humans and provide no means for supporting their automated discovery, invocation, and
The interaction between RESTful services and their clients is normally done with structured data such as XML or JSON\(^4\) different to the standard Web document HTML, which is a human oriented presentation language.

Most of applications provided by the WEB 2.0 community are RESTful services or have APIs to access the application, which can be invoked via RESTful Web services. The liberation of the APIs allow generating new types of applications, which are commonly called mashups. However, to use this type of API it is important to know that the API provides the functionality required and that it is correct.

![Web service functional model](image)

**Figure 2.5: Web service - functional model**

The service model shown in Figure 2.5 indicates that a Web service has a number of operations, each one with inputs and outputs. Input and output messages are not standardized unlike SOAP. This model presents the requirements for a machine-readable description of RESTful Web services. Unsurprisingly, the model is very similar to the structure of WSDL.

### 2.2.2.1 Finding RESTful services

The task of discovering RESTful services is more challenging than the search for WSDL-bases Web services, as there is no widely accepted structural language for describing Web APIs and RESTful services. As a consequence, search engines and crawlers, cannot easily differentiate between common Web sites and Web APIs descriptions.

Finding services is generally a manual and tedious task. Services are located at different servers of different organizations, and are not commonly registered in service

\(^4\)www.json.org
repositories. There are different ways to search for these services: by word of mouth or with specialised search engine and wrappers. This is a typical situation in the search of web services, due to the fact that the information is heterogeneous and located at various servers.

For a service consumer to understand the content of the data that must be sent to and received from the service, both the service consumer and service producer must have an agreement. In RESTful services, this normally takes the form of documentation, sample code, and an API that the service provider publishes for developers to use, as described above. For example, the many web services available from GeoNames, Google, Yahoo, Flickr, and so on have accompanying large pages in HTML format describing how to consume them. This page has all the details for a human to be able to create an application that uses the service. In figure 2.6 we can see a typical documentation page about some RESTful services.

![Documentation](image)

**Figure 2.6: Documentation - Geonames RESTful**

More end-users based annotations are provided on platforms such as Strikeiron.

---

15 www.strikeiron.com

---
2. BACKGROUND AND STATE OF THE ART

ProgrammableWeb\footnote{http://www.programmableweb.com/}, Seekda\footnote{webservices.seekda.com}, BioCatalogue\footnote{http://www.biocatalogue.org/}, iServe\footnote{http://iserve.kmi.open.ac.uk/} APIFinder\footnote{http://www.webapifinder.com}. Strikeiron hosts a Web service marketplace which groups services in pre-defined categories. While there are several features for the Web services, such as pricing, sample code and other, these features are maintained by internal experts and there is no option for end users to provide more annotations. ProgrammableWeb hosts a directory of Web services offering detailed descriptions of APIs which can be annotated using natural language and also some controlled vocabularies. Seekda provides an attribute based description of Web services. It provides means to add Wiki based comments, ratings, or free tags to Web services, and as such it realizes a collaborative approach to Web service annotation. BioCatalogue provides a curated catalog of Life Science Web Services. iServe is a “service warehouse which unifies service publication, analysis, and discovery through the use of lightweight semantics as well as advanced discovery and analytic capabilities”. iServe provides the typical features of service registries and additional functionality that exploits service descriptions, service annotations and further data gathered and derived from the analysis of these descriptions, data crawled from the Web, periodic monitoring and user activities”. APIFinder is a categorized repository about different domains. Furthermore, Searchers (Google, Yahoo, etc.) are system that search information stored in someone Web server.

2.2.2.2 Types of RESTful services

RESTful services commonly allow obtaining lists of entities or details of a specific entity. When a list is returned, each entity contains compact, summary information for each resource. This summary information will typically contain one or two displayable fields, as well as the identifier and URL for retrieving the full record for that entity.

We can distinguish three types of RESTful services with which we deal in this work.

- **With parameters.** http://example.com/resources/operation?param=value. Retrieve a list of entities that match certain conditions.
2.2 Web Services and their Descriptions

- **Specific Entity**: Get information on a specific element URI, such as http://example.com/resources/ef7d-xj36p. Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type. These services belong to a specific RESTful entity and they are always invoked with its associated parameters.

- **Without parameters**: Get all entities Collection URI, such as http://example.com/resources/. List the URIs and perhaps other details of the collection’s members.

REST services are invoked with different types of HTTP requests:

- **HTTP GET requests** are used to read resources. They are also used in list operations that may return multiple items, such as list all, keyword search and partial term matching (term suggestion / term completion). This is the type of request that we focus on in our work.

- **HTTP POST requests** are used to create resources.

- **HTTP PUT requests** are used to update (change) resources.

- **HTTP DELETE requests** are used to delete resources.

The invocation of a specific RESTful service may return diverse formats (XML\(^{22}\), HTML\(^{23}\)). Besides, other formats are also commonly supported:

- **JSON**\(^{24}\) (JavaScript Object Notation), which is a lightweight data-interchange format. It is easy for humans to read and write, and for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999

- **RSS**\(^{25}\) (Rich Site Summary), which is a “format for delivering regularly changing web content. Many news-related sites, weblogs and other on-line publishers syndicate their content as an RSS Feed to whoever wants it”.

\(^{22}\)http://www.w3.org/XML/

\(^{23}\)http://www.w3.org/html/

\(^{24}\)http://www.json.org/

\(^{25}\)http://www.whatisrss.com/
2. BACKGROUND AND STATE OF THE ART

- XHTML\textsuperscript{26}, which is a “family of current and future document types and modules that reproduce, subset, and extend HTML 4. XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents”.

- CSV\textsuperscript{27}, which is tabular data that has been saved as plaintext data separated by commas.

- RDF\textsuperscript{28}, which is a family of World Wide Web Consortium specifications originally designed as a metadata data model.

- JSON-LD\textsuperscript{29}, which is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale.

This allows the service to be used by a variety of clients written in different languages running on different platforms and devices.

2.2.2.3 Syntactic Web Service Descriptions Languages for RESTful Services

Web services require models that capture in sufficient detail the main characteristics of the services, such as the operations offered and the data manipulated, as well as the semantics of both the services and the data that they handle.

RESTful services are almost exclusively described by human-readable documentation describing the URLs and the data expected as input and as output. The community has proposed different approaches to describe RESTful services, such as WRDL\textsuperscript{30}, NSDL\textsuperscript{31}, SMEX-D\textsuperscript{32}, Resedel\textsuperscript{33}, RSWS\textsuperscript{34}, USDL\textsuperscript{35} and WDL\textsuperscript{36}. Most of them were

\textsuperscript{26}http://www.w3.org/TR/xhtml1/
\textsuperscript{27}http://en.wikipedia.org/wiki/Comma-separated_values
\textsuperscript{28}http://en.wikipedia.org/wiki/Resource_Description_Framework
\textsuperscript{29}json-ld.org
\textsuperscript{30}http://www.prescod.net/rest/wrdl/wrdl.html
\textsuperscript{31}http://norman.walsh.name/2005/03/12/nsdl
\textsuperscript{32}http://www.tbray.org/ongoing/When/200x/2005/05/03/SMEX-D
\textsuperscript{33}http://recycledknowledge.blogspot.com/2005/05/resedel.html
\textsuperscript{34}http://www.xml.com/pub/a/ws/2003/10/14/salz.html
\textsuperscript{35}http://www.w3.org/2005/Incubator/usdl/
\textsuperscript{36}http://www.pacificspirit.com/Authoring/WDL/
2.2 Web Services and their Descriptions

more or less ad-hoc inventions designed to solve particular problems and have not been updated for many years. The most recent and alive are hRESTS and WADL. The following sections provide a short overview of both of them.

Web Application Description Language

WADL is as well as WSDL an XML-based format. It is targeting on providing a machine-readable description of HTTP-based web application, mainly RESTful services. WADL can be considered as the most prominent technical description model for RESTful services. Other approaches, like the Web Description Language (WDL)\(^\text{37}\) or the Web Resource Description Language (WRDL)\(^\text{38}\) are nearly without any practical relevance.

WADL is a W3C submission\(^\text{39}\) that provides a means to describe services in terms of schemas (set of resources), relationships between resources, methods that can be applied to each resource, resource representation formats, HTTP methods, and the request or response structures exchanged.

In contrast to WSDL, WADL did not gain so much interest and acceptance so that nowadays the standard is only rarely used.

WADL constitutes an approach that adapts some of the ideas of WSDL for the description of input and output messages. However, WADL only provides syntactic service descriptions and does not define any semantic annotations mechanism(Linting\(^\text{2.1}\)). The listing shows: the format is self-explanatory, and enriches REST with XML schema types (including XML namespace specifications, importing schema grammars, etc.).

Listing 2.1: WADL description of service 1

\[
\begin{verbatim}
<application xmlns="http://wadl.dev.java.net/2009/02">
  <doc xml:lang="en" title="1"/>
  <resources base="http://ws.geonames.org">
    <resource path="countryInfo" id="countryInfo">
      <doc xml:lang="en" title="countryInfo"/>
      <param name="country" default="ES" type="x:s:string" required="false" style="query"
        xmlns:s="http://www.w3.org/2001/XMLSchema"/>
    </resource>
  </resources>
</application>
\end{verbatim}
\]

\(^{37}\)http://www.pacificspirit.com/Authoring/WDL/

\(^{38}\)http://www.prescod.net/rest/wrdl/wrdl.html

\(^{39}\)http://www.w3.org/Submission/2009/SUBM-wadl-20090831/
2. BACKGROUND AND STATE OF THE ART

In contrast to WADL, hREST \cite{KGV08} provides a more lightweight mechanism to service description. Service descriptions in hREST are based on the textual descriptions of the RESTful Web service in HTML. With hREST, we can make the HTML documentation of a Web API machine-processable, with the information about the structure of the API akin to what WSDL can express about WS-* services. hREST is a simple approach compared to WADL or WSDL 2.0, due to its complexity. hREST only carries syntactic information.

The relationship between the HTML page and the hREST description is the following:

- The **service class** indicates that it contains a Web service or an API description.
- The **operation class** indicates that it contains a description of an operation.
- The **address class** indicates the URI of the operation.
- The **method class** specifies the HTTP method (GET, POST, PUT, DELETE) used by the operation.
- The **input and output classes** indicate the description of the input and output parameters of the operation. hRESTS does not provide more machine-readable description about the inputs and outputs.
- The **label class** is used to specify a human-readable label for a service, an operation or a message.
2.2 Web Services and their Descriptions

Listing 2.2: hRESTS format example

```html
...<div class="operation" id="op1">
<a href="#http://ws.geonames.org#countryInfo">
  http://ws.geonames.org/countryInfo
</a>
</div>
<code class="address">
 http://ws.geonames.org/countryInfo
</code>
...</div>
<h4 id="http://ws.geonames.org#Method1">GET</h4>
<h6>request query parameters</h6>
<table>
  <tr>
    <td><span class="input">country</span></td>
    <td><span class="output">countryCode</span></td>
  </tr>
</table>
```

Unified Service Description Language (USDL)

The Unified Service Description Language (USDL)\(^{40}\) aims at bridging the gap between technical descriptions and models required for business related issues. In doing so, it mainly focuses on the description of business aspects (mostly non-functional) orthogonal to technical service implementation. Existing service description models can be integrated into USDL to address technical descriptions. Hence, it is not aiming on the replacement of other specifications, but on adding necessary business information to them.

Linked USDL\(^{41}\) aims at better promoting and supporting the use of the Unified Service Description Language (USDL) on the Web. USDL is a “platform-neutral language for describing services consolidated from SAP Research projects. The kinds of services targeted for coverage by USDL include human services (e.g., consultancy), business services (e.g., purchase order requisition), software services (e.g., WSDL and RESTful services), infrastructure services (e.g., CPU and storage services), etc”.

\(^{40}\)http://www.w3.org/2005/Incubator/usdl/
\(^{41}\)http://linked-usdl.org/
Linked USDL is a “remodelled version of USDL that builds upon the Linked Data principles and the Web of Data”. This effort is therefore most concerned with remodelling the existing USDL specification as an RDF(S) vocabulary that could better support machines in trading services on the Web. To maximise the potential interoperability Linked USDL adopts, where possible, existing RDF(S) vocabularies such as GoodRelations, the Minimal Service Model and FOAF to name a few. Linked USDL is inline with other Linked Data centric initiatives around services such as the work on Linked Services.

2.3 Semantic Annotations of Services on the Web

Current Web services standards have focused on operational and syntactic details for the implementation and execution of Web services. For example, XML-based standards like WSDL, SOAP, and UDDI allow interoperability of applications. However they are more focused on operational and syntactic details in order to implement and execute Web services.

This limits the search mechanism for Web services to keyword based searches. Research in the Semantic Web area has shown that annotation with metadata can help us solve the problem of inefficient keyword based searches. Adding semantic information to syntactical Web service definitions can help interpret their purpose and usage. This is on the premise that a Web service references to a proper ontology which provides a computer- interpretable description of the service.

Most of the time, the syntactic description of a service is not enough. Indeed, two services can have the same syntactic definition but perform significantly different functions. Thus, also the semantics of the data and the behaviour of the service have to be documented and understood. This is normally done in the form of a textual description which is, hopefully, easy understandable by a human being. Machines, on the other hand, have huge problems to understand such documents and cannot extract enough information to use. To address this problem the service can be annotated semantically.

Semantic Web Services, the combination of the Semantic Web and Web Services, aim to provide a mechanism for organizing information and services so that the correct
2.3 Semantic Annotations of Services on the Web

relationships between available data and services can be determined automatically, thus
helping to build workflows for specific problems.

The research area of Semantic Web service aims at creating semantic and formal
service descriptions (MSZ01). This should be achieved by enhancing existing service
description models with semantic to support automated service discovery, consump-
tion and composition. Influenced by the Semantic Web movement, which envisioned
descriptions for Web resources on the basis of ontologies, SWS are also focused on the
use of ontologies as foundation for the service descriptions.

To address the semantics of Web services, the Semantic Web community has de-
veloped an OWL ontology for Web services known as OWL-S. There are also other
Semantic Web Service technologies available such as Web Service Modeling Ontol-
ogy (WSMO), Web Service Semantics (WSDL-S), Semantic Annotation for WSDL
(SAWSDL), Semantic Web Services Framework (SWSF). WSMO and SWSF do not
limit their knowledge representation to description logic. WSDL-S and SAWSDL aim
to extend existing WSDL elements with semantic annotations; thus, they are not defin-
ing a complete ontology framework for Web services as OWL-S does. Most previous
work use OWL-S, and many tools are available.

The following subsections provide a short overview of semantic WS-* and semantic
RESTful approaches.

2.3.1 Semantic WS-*

Service description standards involve towards the use of general Web service standards,
such as WSDL for syntactic service description, whereas OWL and WSMO for semantic
service descriptions. The annotation approach has emerged as a way to bridge the gap
between the syntactic and semantic worlds.

Semantic level is built up through semantic technologies and concepts expected to
enable dynamic and scalable cooperation between different system and organizations
(GABP09). Semantic Web Services are generally defined as the augmentation of Web
Service descriptions through Semantic Web annotations, in order to facilitate the higher
automation of service discovery, composition, invocation, and monitoring in an open,
unregulated, and often chaotic environment (that is, the Web)(PL04).
2. BACKGROUND AND STATE OF THE ART

2.3.1.1 OWL-S

OWL-S provides abstract constructs for input and output process parameters. OWL-S consists of three parts, service profile, process model and grounding. Service profile includes general information and is used to describe what the service will do; process model describes how the service will perform its functionally while grounding describe links with industry standards. Its main goal is to enable users to automatically and semanticallyally discover, invoke, compose and execute web services under certain constrains.

2.3.1.2 WSMO

WSMO provides a rich description of all the related aspects of Web services and allows the annotation of functionality and behavior of Web services. Moreover it allows the attachment of non-functional properties whose intention is to describe non-functional aspects of Web services such as the creator, rights, or quality of service.

2.3.1.3 SAWSDL

Approaches which directly integrate semantic annotations into WSDL files are WSDL-S and its successor SAWSDL. Both provide means to point to semantic descriptions outside of the WSDL.

SAWSDL is a w3c recommendation. SAWSDL suggest adding semantics to WSDL by using extensibility in the elements and attributes supported by the WSDL specification as well permitting the relation between existing WSDL constructs and ontology concepts.

SAWSDL constitutes a lightweight framework aimed to add semantic annotations to syntactical WSDL files. It allows add annotations types through a Model Reference:

- port/types interfaces
- inputs or outputs types
- faults
- operations

\[\text{http://www.wsmo.org/2004/d2/}\]
\[\text{http://www.w3.org/TR/sawsdl/}\]
2.3 Semantic Annotations of Services on the Web

2.3.1.4 SWS WS-* Limitations

All the semantic service frameworks proposed so far, e.g. WSMO, SAWSDL, OWL-S share a common principle: they assume that the (semantic) description of services comes solely from the service providers. A critic of them (Klu08): SAWSDL originates as a mere syntactic extension of WSDL and lacks of formal semantics; OWL-S is limited expressiveness of service descriptions, only static and deterministic aspects of the world can be described;

Semantic service efforts have still a low adoption rate. To come up with elaborated semantic service descriptions, the service providers have to be convinced about the additional value of these semantics in order to spend resources to annotate their services. The challenge of our research is to give a novel approach to automatically annotate web services by finding precise semantic link of service.

2.3.2 Semantic RESTful Services

Generally, the high complexity of SWS approach discourages both technical and business people from adopting such solutions. These problems have created pressure to the SWS research community to come up with lightweight approaches, which may lack in expressivity but win in simplicity (e.g. SAWSDL, SA-REST, MicroWSMO, WSMO-Lite). However, it is still early to evaluate the applicability and adoption of such lightweight semantic service models.

Adding semantics to RESTful Web service is more challenging than adding semantics to WSDL. Enrichment of RESTful services with semantic metadata facilitates RESTful services organization, indexing and retrieval. However, most RESTful services have not semantic due to the difficulty and the cost of manually annotating them.

<table>
<thead>
<tr>
<th>SA-REST</th>
<th>MicroWSMO</th>
<th>- advanced tool support</th>
</tr>
</thead>
<tbody>
<tr>
<td>hRESTS microformat</td>
<td>- machine-readable information</td>
<td></td>
</tr>
<tr>
<td>HTML (text, link, forms)</td>
<td>- existing service descriptions</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.7: hREST - layer cake
2. BACKGROUND AND STATE OF THE ART

In comparison to WSDL-based services, approaches for the creation of semantic descriptions of RESTful services are hampered because the majority of the Web APIs are described in textual form in Web pages and do not have WSDL-like machine-processable documentations. The lack of a common structured language for describing Web APIs is addressed by some initial work such as WADL and WSDL 2.0 as described in section 2.2.2 however still the majority of the APIs are documented directly in HTML.

MicroWSMO and SA-REST (Fig. 2.7) are two formalisms for the semantic descriptions of RESTful services, which are based on adapting the SAWSDL approach.

2.3.2.1 Semantic Annotation Approaches

Here, we provide a brief introduction to semantic annotatios of RESTful services, and describe existing approaches related to the semantic annotation of such types of services.

SA-REST

SA-REST [SGL07] is an RDFa-based annotation mechanisms that extends hREST with support for the description of these different service properties (domain-rel, sem-rel and sem-class).

Listing 2.3: SA-REST description of our sample RESTful services

...<div class="section domain-rel" lang="en" title="sarest:Service" >
...<span class="domain-rel" title="sarest:Operation" >
<a href="#http://ws.geonames.org#countryInfo">http://ws.geonames.org/countryInfo</a>
</span>
...<code class="address">http://ws.geonames.org/countryInfo</code>
...<span class="domain-rel" title="sarest:inputMessage" >
<strong>country</strong>
</span>
...<td>
<a href="http://xsd" class="sem-rel" title="http://dbpedia.org/ontology/country" >
This is the input service</a>
</td>
...
}<span class="domain-rel" title="sarest:outputMessage" >
SA-REST is a Plain Old Semantic HTML (POSH) format to add additional meta-data to (but not limited) REST API description in HTML or XHTML. Being POSH means that the embedded annotations are similar in nature to Microformats, but may not necessarily have gone through a rigorous open community process. SA-REST is flexible enough to use meta-data from different models such an ontology, taxonomy, or a tag cloud. Annotating means adding pointers (Model Reference) to ontology. This embedded meta-data permits various enhancements, such as improve search, facilitate data mediation, and provide easier integration of services.

Basic properties are specified using the class and the title attributes defined by the HTML specification. SA-REST has three properties (types of annotations) that can be applied to an XHTML document. It defines classes for describing the data format and programming language binding properties of Web APIs, thereby allowing developers to search in homogeneous groups.

- **domain-rel.** This property allows a domain information description of a resource. Enumeration of URIs (optional). Allows a domain information description for a resource.

- **sem-rel.** Enumeration of URIs (optional). Captures the semantics of a link and envolves from the popular rel tag.

- **sem-class.** Enumeration of URIs (optional). This property can be used to markup a single entity within a resource.

SA-REST is designed for humans first and machines later. SA-REST is based on SAWSDL, extend REST while keeping simplicity in mind and allow annotates inputs, outputs, operations, and faults, along with the type of request that it needed to invoke the service.

In SA-REST the subject of the triple should be the URI at which you would invoke the service; the predicate of the triple should be either sareset:input, sareset:output,
2. BACKGROUND AND STATE OF THE ART

sarest:operation, sarest:lifting, sarest:lowering or sarest:fault, where sarest is the alias to SAREST namespace. The object of the triple should be either a URI or a URL to a resource depending of the predicate of the triple.

SA-REST embed semantic annotations in RDFa into the HTML page that describes the service, thus making the page both a human readable and machine-readable and at the same time creating a single place to do an update if the service ever changes.

WSMO-Lite

The main classes of WSMO-Lite are Condition, Effect and FunctionalClassificationParameter, used for capturing functional and behavioral semantics, and NonFunctional-Parameter for nonFunctional semantics. Listing 2.4 shows the WSMO-Lite service ontology in RDFS, serialized in Notation 3. Below we explain the semantics of the WSMO-Lite elements.

Listing 2.4: WSMO-Lite Service Ontology

```xml
@prefix rdf: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#> .

wsl:Ontology a rdfs:Class;
  rdfs:subClassOf owl:Ontology.
wsl:FunctionalClassificationRoot rdfs:subClassOf rdfs:Class.
wsl:NonFunctionalParameter a rdfs:Class.
wsl:Condition a rdfs:Class.
wsl:Effect a rdfs:Class.
```

wsl:Ontology defines a container for a collection of assertions about the information model of a service. wsl:Ontology is a subclass of owl:Ontology limited to such ontologies that may serve as information models. OWL ontology meta-data such as comments, version control and inclusion of other ontologies, are also allowed on wsl:Ontology.

wsl:FunctionalClassificationRoot marks a class that is a root of a classification which also includes all the RDFS subclasses on the root class. A classification of service functionalities can be used for the functional description of a service.

wsl:NonFunctionalParameter specifies a placeholder for concrete domain specific nonfunctional property.

---

44http://www.w3.org/Submission/2010/SUBM-WSMO-Lite-20100823/
2.3 Semantic Annotations of Services on the Web

Figure 2.8: Relative positioning of WSMO-Lite and MicroWSMO

ws:l:Condition and ws:l:Effect together form a capability in a functional service description. Both are expected to use a concrete logical language to describe the logical expressions for conditions and effects.

**MicroWSMO**

MicroWSMO \([KVFG09]\) is an extension of hREST that adds semantic annotations. It captures four aspects of service semantics: information model (a domain ontology), functional semantics, behavioral semantics, and non functional descriptions as in WSMO-Lite. Figure 2.8 illustrate de relations of MicroWSMO to SAWSDL, along with their positioning amount various service description specification. MicroWSMO is a SAWSDL like layer on top of hRESTS.

MicroWSMO \([KVFG09]\) defines the following three XML attributes, along with RDF properties with the same names:

- `modelReference` is used on any component in the service model to point to appropriate semantic concepts identified by URIs.

- `liftingSchemaMapping` and `loweringSchemaMapping` are used to associate messages with appropriate transformations, also identified by URIs, between the underlying technical format such as XML and a semantic knowledge representation format such as RDF.
2. BACKGROUND AND STATE OF THE ART

MicroWSMO and hRESTS, together with the WSMO-Lite ontology for service semantics, support automation of the use of RESTful Web Services.

Minimal Service Model

The service ontology model, referred to as Minimal Service Model (MSM) \( \text{PLM}^{+10} \), is a simple RDFS ontology that was designed to capture the core of both classical Web services and Web APIs in a common model.

MSM is a simple model for capturing service description, which: covers WSDL and Web APIs homogeneously, captures the core semantics of services and data employed by the main semantic Web service models, minimizes modeling and processing overhead, closer aligns services with linked data. MSM is a simple RDFS ontology based on the principle of minimal ontological commitment. It captures the common structures of existing conceptual models for services. This model is able to capture the core semantics of both Web services and Web APIs in a way that enables the homogeneous publication and discovery of both kinds of services.

MSM characterizes Services as being composed of a number of operations which in turn have input, output and fault MessageContent descriptions. This model is complemented by the WSMO-Lite vocabulary \( \text{45} \) which define classes for describing the four core aspects of service semantics identified by previews research on service semantics, namely, functional semantics, nonFunctional semantics, behavioral semantics and an information model. These types of services semantics are relevant for advanced discovery, selection and composition, among other tasks.

Other approaches

RESTfulGrounding \( \text{FF09} \) does for REST and WADL what OWL-S does for WSDL. It gets Semantic descriptions into the syntactic descriptions that automated services may use to interact with a web service, and facilitates discovery, composition, monitoring and execution.

EXPRESS \( \text{AMT09} \) takes a different approach and based on a existing RESTful web service allows us to create an OWL description that can also RESTfully accessed to describe the services, resources, relations and parameters. They describe an adaptation

\[^{45}\text{http://www.w3.org/Submission/WSMO-Lite/}\]
of Amazon S3 buckets and docs with EXPRESS and compare with SA-REST and OWL-S approaches.

2.3.2.2 RESTful Semantic Annotations Tools

The most representative tool for the semantic annotations of RESTful services is SWEET (Semantic Web Services Editing), “developed for supporting users in creating semantic RESTful services by structuring service descriptions and associating semantic annotations”. SWEET uses an ontology annotation schema to describe annotation (Minimal Service Model). Besides SWEET, there are some other manual annotation tools that could potentially be used for semantically annotating Web Services, for example Kino. Kino is a faceted search/indexing engine set up for services, it uses SA-REST as the annotations mechanism and uses Apache SOLR as the faceting indexing engine.

Most automatic semantic annotations tools assume a given pre-constructed domain ontology to avoid the difficult automatic ontology generation problem.

2.3.2.3 Limitations of Semantic RESTful Annotation

The attempt to standarize Web services has taken years. While a number of different approaches, such as OWL-S, WSMO, WSMO-Lite, MicroWSMO, SAWSDL and SAREST as described in the previous sections, have been proposed, none so far has managed to break out of its academic confines. Looking at RESTful-based services the situation looks even worse; there does not even exists a widely accepted standard for the syntactic and semantic description of a RESTful services. No services are semantically described because there are not semantically annotated services.

The semantic annotation of RESTful services presents some difficulties. According to [MKP09, PLM+10, AMT09], these difficulties are related to lack of machine-processable descriptions, which create more work for humans to find a service. Furthermore, there are some problems with documentation, due to the fact that users have to read the service documentation for understanding and using them. With respect to technological issues, WS-* and RESTful approaches are manuals and complex. They only offer syntactic description and are insufficient to semantically annotate a service.

46http://wiki.knoesis.org/index.php/Kino
2. BACKGROUND AND STATE OF THE ART

Web service discovery, composition, and invocation need to do manually; manual labor reduces scalability; the situation is considerably worse for WS APIs.

Some advantage of add semantic annotation to RESTful services

- It allows working with high level concepts
- It allows creating the possibility for homogeneity
- It allows passing of concepts not low level data structure
- It allows adding possibility for advanced search, allows filter search using specific parameters in order to receive results that are more accurate.
- It allows adding possibility for discovery, provisioning of services across multiple or heterogeneous registries, differentiating between services that share similar functionalities.

An important challenge is the lack of tools that generate (semi) automatically semantic annotations of RESTful services.

RESTful services still requires much human intervention since the majority of their descriptions are given in the form of unstructured text in Web pages (HTML), which contain lists of available operations, their URIs and parameters (also called attributes), expected output, error messages, and some examples of their execution. This hampers the automatic discovery, interpretation and invocation of these services, what may be helpful to improve the efficiency in the development of applications that are based on their use.

In this thesis, we describe our approach to address the challenge of automating the semantic annotation of services by: (1) obtaining and formalising their syntactic descriptions, what allows their registration and invocation, and (2) interpreting and semantically enriching their parameters, what allows generating semantically-enriched service descriptions in any of the available formalisms for RESTful semantic service description.
2.4 Geospatial Web Services

OGC Web Services are commonly used in the Geospatial Web. Web Service Description Language (WSDL) documents are required to build applications based on a SOA approach, but current OGC standards for Geospatial web Services do not yet support this standard (Neb04).

According to (SS09), geospatial data correspond approximately to 80% of all available data, thus, it contributes to human knowledge. However, in order to be used, these geospatial data have to be published, analyzed and interpreted.

Nowadays, there are many Spatial Data Infrastructures (SDI), which allow sharing of geospatial data over the Internet. SDIs use the basic principles of SOA. An SDI provides basic technologies, policies and standards to access, share or process geospatial data. However, the search, retrieval and composition of geospatial data are difficult, because the vocabulary used in different SDI generate semantic heterogeneity problems when only simple keyword-based search is employed (KLK06). The geospatial data is available by different SDI through catalogue services as data, maps, and satellite images allowing users to discover them, however not much meaning is attached to them.

Geospatial Web services (GWS) are changing the way in which spatial information systems and applications are designed, developed, and deployed. GWS is a set of services and data that support the use of geospatial data in many domain application. Furthermore, GWS are growing increasingly being used to access, integrate and reuse geospatial data. However, the search and retrieval of useful spatial data and services generate a challenge to the user community due to the variety in meaning of data and Web services (Ege02a). It is not easy for a user to find the right service with a concrete functionality for a specific purpose. Existing catalogues provide searchable repositories of information and services, but mechanisms for supporting discovery and retrieval are insufficient (LEKH04). Because of the continuous increase of GWS, it is very important to find the proper data and service matching specific requirements. Metadata are always used in describing and discovering geospatial Web services. Metadata have been used in several context (AF07). However, mismatching may arise caused by semantic ambiguity, or when the same term is used in different domains may lead to a different meaning.
2. BACKGROUND AND STATE OF THE ART

Generally, a geospatial Web service can be viewed as a modular Web application that provides services on geospatial data, information, or knowledge (DZY+05). It refers to the use of Web services technologies to manage, analyze, and distribute spatial information. Furthermore, a geospatial Web service can be searched and sorted by using its geospatial characteristics, such as location, area, neighborhood, and other spatial features.

Due to the influential specification from OGC, geospatial Web services have as an add-on unique geospatial characteristics on the message-communicating other than those popular W3C Web services standards such as WSDL, SOAP and UDDI. In order to serve the user communities with GIS datasets through catalogue services, SOA principles are used (AAF+05). Moreover, GWS focus directly on standard interface through HTTP protocol, which is now recognized as more related to REST (Boo04, Fie00)

As a Web service, a geospatial Web service involves three actors: user entity (consumer), provider entity (provider), and registry entity (broker) (Boo04) as described in section 2.2. These actors can execute basic operations during the life cycle of a geospatial Web service include publication, discovery, binding, invoking, and execution. The operations of some geospatial Web service have been standardized.

The most important institutions for such standardization efforts are ISO/TC21 and OGC.

2.4.1 OGC and ISO/TC211

In the geospatial Web service area, OGC and ISO/TC211 are the main actors in standardizing geospatial Web services. Next, we describe the main characteristics of them:

- OGC specification focuses on developing implementation standards by adapting or extending the common Web service standards.

- ISO/TC211 concentrates on developing theoretical/abstract standards. ISO/TC211 standards specify methods, tools, and services for acquiring, processing, analyzing, accessing, presenting, and transferring spatial information between different users, systems, and locations. OGC specifications support the full integration of geo-enabled Web services into mainstream computing in order to make complex...
2.4 Geospatial Web Services

Geospatial information and services accessible and useful for all kind of applications.

OGC has been dedicated to the standardization of interfaces of geospatial Web services to enable the interoperability. All OGC Web services are based on a common model and share some unified characteristics: 1) some common operation request and response contents (getCapabilities), 2) some unified parameters for operations request and response, and 3) exclusive adoption of XML and key value-pair in encoding. Under such a common specification, series of specifications (WMS, WFS, WMC, etc.) were developed and adopted for delivering and analyzing geospatial data over the Web.

The OGC has successfully executed efforts for GIS interoperability. For example, OGC Web Service (OWS) initiative has subjected to multiple phases, including WMS, WFS, WCS and OGC Web Service Architecture[47] which support application development by integrating a variety of online geoprocesing and location services.

Conceptually, the OWS stack is a service-oriented architecture (SOA) that includes service discovery, description, and binding layers corresponding to UDDI, WSDL, and SOAP in the W3C standards. Currently, OWS are not equivalent to the W3C SOAP-based Web services, although the OGC is attempting to integrate the Web services standards into the OWS framework, including specifying changes to the common OWS architecture and providing WSDL description in WMS, WFS and WCS.

OGC geospatial Web services are different from Web services in the used domain. OGC geospatial Web service has been developed in parallel with the evolution of W3C and OASIS Web services. Standards from W3C and OASIS, such as WSDL, SOAP and UDDI, are the standards for Web services in the e-business domain. The OGC geospatial Web services do not comply with these Web services standards. Actually, OGC explores the possibilities in bridging the gaps and implementing OGC Web services using W3C/OASIS Web services[48]. Another step in standardization of geospatial Web services in the exploration of Semantic Web interoperability experiment[49].

Interoperability of platforms for discovering and accessing data/services has been created and supported by the OGC through established protocols and interface specifications such as Web Catalogue Services (CSW) thereby offering support for the impor-

---

[47] portal.opengeospatial.org/files/?artifact_id=1320
2. BACKGROUND AND STATE OF THE ART

tant task of discovery and retrieval of information that meets the user’s needs. Retrieval methods that are currently in use are typically limited to keyword search or sub-string matching only. The search methods only account for the syntax of the search terms without taking into account the underlying conceptualizations. Using these methods, the information required is poorly defined in the search and results often do not satisfy the user’s needs. As a result users may often miss critical information when searching for spatial Web services.

Data and services provided by different SDIs can be shared and exchanged over the web and semantic annotations play an important role in the discovery of geospatial Web services.

The introduced OGC standards ensure syntactic operability (e.g. data format, communication protocol, message format). We describe two Web services as interoperable if then can communicate and exchange data that is based on a shared data model. In other words, two Web services are interoperable if they are interoperable on the syntactic, structural and semantic level. OGC standards ensuring structural interoperability.

2.4.1.1 Web Features Services

The OpenGis Web Feature Services (WFS) allow a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. WFS define interfaces for data access and manipulation operation on geographic features, using HTTP as the distributed computing platform. The following operation are available to manage and query geographic features: create a new feature instance, delete a feature instance, update a feature instance, lock a feature instance, and get or query features based on spatial and no spatial restrictions. It is possible to request geospatial data served by OGC Web services via unique URL for example vector data offered by a WFS. In OGC catalogues this WFS is registered as metadata.

Currently, each SDI creates WFS services and publication in their web site or catalogue services. Ofthen, the descriptions to these services are offered in a HTML page. These pages contain information about how access to the services, description of features, restrictions and so on. The lack of a repository of WFS services makes to find services difficult, forcing the manual search of services through the common search or browse through websites of SDI. All these syntactic descriptions suggest that the presence of a semantic description of the service is necessary.
2.4 Geospatial Web Services

So far, no automated methods for the Semantic Annotations of WFS exist. Manual annotation is difficult, time consuming and expensive. The annotation process should be as automatic as possible, since manual process can be slow and subject to errors.

We proposed a method for automating the annotation process, both syntactically and semantically based on the specific characteristics of geographic information.

2.4.1.2 Syntactic Descriptions

Standards for description of geospatial data exist as well as catalogue of services to search them. In current standard-basis catalogues, the users can formulate queries using keywords and/or spatial filters. The descriptions of these resources are only syntactic. OGC standard concentrate on syntactic interoperability, OGC define a series of syntactic interface specification, establishing protocols for components exchanging geospatial information. If additional description are contained, the mostly consist of plain-text. However, these do not account for the fact that the conceptualization of features are necessary, causing semantic heterogeneity during discovery and retrieval (LK06).

A first step towards semantic enablement is to annotate these resources of the General Feature Model (ISO05). For these resources to be more useful to users, they have to be annotated with descriptive metadata. Annotations formally identify resources through the use of concepts and the relationships among them (YDZ+06). Semantic annotations links them to the according classes specified within ontologies. The semantic web ideas have brought the semantic issues of information processing into perspective (BLHL01).

Semantic interoperability is a precondition for effectively finding and accessing relevant data in different application context. An important means for achieving semantic interoperability are the ontologies, which capture consensual knowledge and formalize this knowledge in a machine interpretable way.

2.4.1.3 Semantic Annotations

The Semantic Web for geographic information, called Geospatial Semantic Web by Egenhofer (Ege02b), is the way to process request involving different kinds of geospatial data.
2. BACKGROUND AND STATE OF THE ART

The goal of the annotation process is to semantically annotate different kinds of geospatial data, such as satellite images, maps, etc. The work of (AF07) proposed a formal model for annotating different kinds of digital content, such as textual documents, images, services, and multimedia documents in general. According to them, an annotation model should be as uniform as possible, considering all kind of contents, but also flexible, enabling their semantic exploitation.

However, current standards focus on syntactic specification and do not address the semantic geospatial Web services. Semantic Web technologies (BLHL01) provide promises for achieving semantic operability of geospatial Web services. Semantic Web services have been widely used to enabled dynamic service composition (SK03, RS04), and are being introduced into geospatial domain for creating geo-semantic web services (LWdB+06, YDY+07, Lut07) and ensuring better precision of search results (Lut07).

Semantic annotations are increasingly playing an important role in the world of metadata dealing with semantic heterogeneities between information systems. The process of semantic annotation is based on geospatial evidence as it considers the spatial component. The annotation process takes advantage of standards provided by OGC such as GML, although the annotation schema itself uses geospatial metadata standards.

Ontologies has been used in the geospatial domain for information integration and semantic interoperability (FEAC02, LK06). Through mapping concepts in a geospatial web resource to ontological concepts in the geospatial domain, the semantics of those geospatial resources can be explicitly defined.

Realizing the standards established by OGC and how they can be augmented with semantic web technologies for better service description and discovery, W3C has come up with recommendations for Semantic Annotations for WSDL and XML Schema (SAWSDL51). The specification defines “how to annotate WSDL interfaces and operations with categorization information that can be used to publish a Web Service in a registry”.

In the following, we summarize the different researches with respect to semantic annotations of WFS services:

50http://www.opengeospatial.org/standards/as
51http://www.w3.org/TR/sawSDL/
2.4 Geospatial Web Services

- In the geospatial domain, most efforts focus on ontology-based descriptions and semantic matching for discovery of geospatial data and services, with little consideration of registration of semantics in the catalogue service [LK06, LEKH04, LK07].

- KlienEinspanier proposed an architecture for ontology based discovery and retrieval geographic information. The user is allowed to formulate a query for metadata and geodata.

- In [KL05], Klien presented an approach for automating the Semantic Annotations of geodata. This work explains the role of spatial relations for extracting implicit instance level information. Furthermore, suggest to apply spatial analysis methods in order to extract information on spatial relations useful for annotation. Compared to knowledge extraction techniques like string-based attribute analysis, the calculation of spatial relations remains independent from the textual description of geographic features as their properties. This has advantage that semantic heterogeneity problems inherent in the processing of natural language description are avoided. The authors concentrated on the role of spatial relations. Ontologies can be applied for making the semantics of information content of geospatial Web services explicit in order to enhance geographic information discovery and retrieval in geospatial web services environments.

- In [MdSM09] is presented a proposal of a framework for semantic annotations of geospatial data available on the Web, oriented towards agricultural planning and monitoring. The generation is manually for different kind of content.

- In [YDY+09], the author proposes the semantic descriptions for geospatial data, services, and geoprocessing service chains based on the ontologies. This purpose is not to propose new ontologies for semantic description of data, services and geoprocessing chains. Rather, they use the existing set of example ontologies and show how they can be exploited in a catalogue service.

- In [YDZ+06] described briefly the semantic-enabled discovery of geospatial data and services.
2. BACKGROUND AND STATE OF THE ART

- Maue (Mau08) proposes a semantic geospatial service catalogue to support the discovery of geospatial services. The service platform proposed by the Open Service Gateway Initiative (OSGI) is used for catalogue implementation. Thus, his implementation does not follow a standard like UDDI or erRIM. OGC data services, such as those following WFS or WMS specifications, are registered in the catalogue. WSMO is used for a semantic description of geospatial services.

There have been studies focused on Semantic Annotations of WFS. However, as pointed out in our work, an automatic approach is proposed. In other words, this approach takes into account the instance of dataset to connect syntactic world to the semantic world.

(MSD09) in the discussion paper OGC 08-167r1 Semantic Annotation in OGC standards define the best practice solutions to add references to existing standards such as GML, KML, GML schema, OWS, etc. focused on compliance of re-using existing features in standards. This paper already analysis the suitability of existing GIS data representation and query languages for the expression of semantic annotations. In particular, it covers the case of ebXML, ISO19139, OGC filters, to name a few.

This discussion paper proposed a methodology for the annotation of OGC services. The authors suggest three different levels of semantic annotations, distinguish between the annotation of resource metadata, e.g., an OWS Capability document, of a data model e.g. a GML schema, and of data entities, such as GML file. Figure 2.9 shows the three levels at which semantic annotations can be performed. Level 1 (Service Metadata) link the keywords of capabilities document to concepts from domain ontology. The connection between the application schema to application ontology is shown in level 2 (Data Model). Linking features to concepts from domain ontology is illustrated in level 3 (Data Entities).

- Service Metadata. Metadata section such as keywords are semantically enriched so as to make the meaning reality. Keywords that are registered in the metadata offer limited information about the service. To give them more meaning, the keywords are linked directly to the concept in the domain ontology by adding pointers to the URIs.
2.4 Geospatial Web Services

Data Model. Within this level, semantic annotations focus on associating feature type and feature attributes to concepts in the resource ontology. An additional attribute such as an identifier (URI) established the association between the concepts which are in the XML schema and the resource ontology.

Data Entities. Semantic annotation can also be done at attribute level. In OWS, GML features and feature attributes are entities which are annotated at this level. Individual entities are annotated either with domain concepts or with individuals on the domain level. The individual entities are tightly coupled to domain concepts.

Data model and data entities semantic annotations are of more used where spatial reasoning is required.

2.4.2 Semantic Resources: Ontologies and SPARQL endpoints

In our approach we use different kind of ontologies and vocabularies, such as: DBpedia as a general ontology, GeoNames and WGS84 as domain ontologies and WSMO-lite and POSM as representation ontologies.
2. BACKGROUND AND STATE OF THE ART

2.4.2.1 Scheme Annotation

Next, we summarize different research related to schema annotations, these works have been already introduced in previous sections:

- **WSMO-lite** is a lightweight approach to the semantic annotations of Web services descriptions. This ontology represents Information Model Descriptions defining the elements and attributes of the underlying data model.

- **Procedure-Oriented Service Model (POSM)** is a “lightweight approach to the structural description of procedure-oriented Web services, compatible with WSMO-Lite annotation”. It is developed by the Linked Open Service initiative. POSM builds on Web service descriptions in RDF, which in addition is suitable for the growing Linked Open Data community. POSM contains generic statements of Web services, e.g. a service has an operation, which has input and output messages. POSM, similar to the WSDL description of a Web service, includes elements describing the operations, which are provided by the service. It replaces the former role of the Minimal Service Model and is complemented by the Resource-Oriented Service Model (ROSM).

The source of the POSM ontology is illustrated in the listing 2.5. Our approach describes WFS following the model proposed by POSM in order to achieve represents the WFS service structural.

**Listing 2.5: POSM (n3 format)**

```n3
# namespace declarations (this line is a comment)
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix part: <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/part.owl#> .
@prefix posm: <http://www.wsmo.org/ns/posm/0.1#> .

# classes for use in structural definition of procedure-oriented services
posm:Service a rdfs:Class .
```

http://www.wsmo.org/ns/wsmo-lite/
http://www.wsmo.org/ns/posm/0.1/
http://www.linkedopenservices.org
http://linkeddata.org
http://www.wsmo.org/ns/rosm/0.1/
2.4 Geospatial Web Services

- WFS ontology captures the service implementation rules for WFS as specified in the OGC WFS implementation specification. This ontology describes two specific operation, such as: DescribeFeatureType and GetFeature, and as well as their corresponding input and output message.

---

https://code.google.com/p/wsmls/source/browse/trunk/application/OGC/WFS/0.2/WFS.rdf?r=289
2. BACKGROUND AND STATE OF THE ART

2.4.2.2 General Resources Annotation

In the following, we summarize the different general resources annotations:

- As we said before, DBpedia [ABK+07] is a community effort to extract structured information from Wikipedia and to make this information available on the Web, hence allowing sophisticated queries against Wikipedia, and linking other data sets on the Web to Wikipedia data.

With respect to its ontology, it is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia. The ontology (2014) currently covers over 685 classes which form a subsumption hierarchy and are described by 2795 different properties. Moreover, this ontology currently contains about 4,233,000 instances (735,000 instances belong to Place class).

- As we said before, Geonames [Geo10] is integrating geographical data such as names of places in various languages, elevation, population and others from various sources. All lat/long coordinates are in WGS84 (World Geodetic System 1984). Users may manually edit, correct and add new names using a user-friendly wiki interface. Likewise, the GeoNames geographical database is available for download free of charge under a creative commons attribution license.

Regarding its ontology, it makes it possible to add geospatial semantic information to the World Wide Web. All over 6.2 million GeoNames toponyms have a unique URL with a corresponding RDF web service.

- Dolce[^58] is an Upper-level Ontology. DOLCE has a clear cognitive bias, in the sense that it aims at capturing the ontological categories underlying natural language and human commonsense. According to DOLCE, different entities can be co-located in the same space-time. DOLCE is described by its authors as an ontology of particulars, which the authors explain as meaning an ontology of instances, rather than an ontology of universals or properties. The taxonomy of the most basic categories of particulars assumed in DOLCE includes, for example, abstract quality, abstract region, agentive physical object, amount of matter,

[^58]: http://www.loa.istc.cnr.it/DOLCE.html
2.4 Geospatial Web Services

non-agentive physical object, physical quality, physical region, process, temporal quality, temporal region.

- **SPARQL Query Language for RDF** SPARQL Query Language for RDF (SPARQL) \(^{(PS08)}\) is a query language for RDF. This language can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. More details about SPARQL can be found in the W3C Specification.

A SPARQL endpoint is a conformance-mant SPARQL protocol service as defined in the SPROT specification. A SPARQL endpoint enables users (human or other) to query a knowledge base via the SPARQL language. Results are typically returned in one or more machine-processable formats. Therefore, a SPARQL endpoint is mostly conceived as a machine-friendly interface towards a knowledge base. Both the formulation of the queries and the human-readable presentation of the results should typically be implemented by the calling software, and not be done manually by human users.

2.4.2.3 Geospatial Domain Ontologies

- **WGS84** \(^{(59)}\) is a “basic RDF vocabulary that provides the Semantic Web community with a namespace for representing lat(itude), long(itude) and other information about spatially-located things, using WGS84 as a reference datum”.

- **HydrOntology** \(^{(60)}\) is an ontology in OWL that follows a top-down development approach. Its main goal is to harmonize heterogeneous information sources coming from several cartographic agencies and other international resources.

- **GeoSPARQL** \(^{(61)}\) defines “a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data”.

\(^{(59)}\)http://www.w3.org/2003/01/geo/
\(^{(61)}\)http://www.opengeospatial.org/standards/geosparql
2. BACKGROUND AND STATE OF THE ART

2.4.3 Tools

Much work has been done in the state of the art of Geospatial Web Services semantic annotation. Moreover, several tools have been made available for WFS services semantic annotations. Many of these tools constitute editors that help services developers to manually create descriptions according to vocabularies. Without the aim of being exhaustive, we describe in the following paragraphs some relevant work related to our proposal.

First, we mention the SWING (RK07) project, which has a visual interface tool that allows users to semantically annotate WFS services (GKN09).

Second, the Semantic Annotations Proxy (SAPR) (MMR12) project, that has been developed as SaaS (Software as a Service) by making use of the google app engine. It can be seen as a light-weight RESTful API that allows the semantic annotations injection into the Web services holding the standards.

Finally, the ENVISION project, which has made available several functionalities listed as follows: i) Ontologies that aims to support non ICT-skilled users in the process of semantic discovery and adaptive chaining and composition of environmental services; ii) ENVISION supports the creation of web-based applications enabled for dynamic discovery and visual service chaining; iii) The ENVISION Ontology Infrastructure provides support for visual semantic annotation tools and multilingual ontology management; iv) The ENVISION Execution Infrastructure comprises a semantic discovery catalogue and a semantic service mediator based on a generic semantic framework and adaptive service chaining with data-driven adaptability.

2.4.4 Limitations of the Current State of the Art on Syntactic Descriptions and Semantic Annotations

Geospatial data are made available to users through Web services. In order to discover and retrieve information about services, users use metadata registered in catalogue services. Discovery of services registered in catalogues strongly depends on the expressiveness of the metadata and the users’ ability to formulate good queries. The lack of proper descriptions of the resources results in limited use, usually only those who have expert knowledge about the resources. Currently there is no prescribed way to search

---

http://www.envision-project.eu/resources/annotated-services/
for information in catalogues though often searching requires skills in formulation of a query, which the user may not have. Search for services is done through the traditional means of using keywords. The catalogue analyzes the query, compares the given search criteria to the registered service description and returns all matching records.

Considering the vast amount of information available in terms of web technologies, the data and services searching process is marked by a number of challenges. Current search mechanisms are based on keywords only, without taking advantage of underlying hierarchical structures that can improve the search mechanisms. The limitations on the search functionality are:

- The XML based standards like UDDI, WSDL, and SOAP allow syntactic interoperability of applications. However, they are more focused on operational and syntactic details in order to implement and execute Web services. UDDI introduce keyword-based search and retrieval mechanism without any attempt to use underlying hierarchical relationship\cite{YDZ06}.

- UDDI is domain-independent, and so it does not support domain-specific query capabilities particularly for domains such as GIS requiring spatial queries.

- The lack of machine-understandable semantics in the technical specification and classification schema used for retrieving services, prevent UDDI registries from supporting truly effective service discovery.

- Whilst WFS standards cater for technical aspects that allow integration of information; they ignore the semantic aspect, thus limiting scope of data sharing between providers and users.

- Formulation of queries is often left to the user who may have little or no knowledge and experience about the service or data they are looking for, so poor query formulation also results in poor results.

The use of semantic technologies is seen as a part of solution to overcoming limitations of current search mechanism in geospatial data.
2. BACKGROUND AND STATE OF THE ART
3

Objectives and Contributions

This chapter summarizes the research questions, objectives, hypotheses and assumptions associated with this work. Figure 3.1 depicts a general overview of our contributions and the relation between them. Next, we present the main contributions of this thesis:

Figure 3.1: Thesis - main contributions
3. OBJECTIVES AND CONTRIBUTIONS

i) The definition of a semantic model for service annotation. This model provides support for the annotation of RESTful and WFS services.

ii) The development of a process for the semantic annotation of services by using external resources. This contribution is composed of: (a) A set of algorithms and their corresponding implementation in the annotation process. (b) The definition of a gold standard for RESTful and WFS services annotating.

We have defined a semantic model, algorithms and methods concerned with several existing service definition approaches. Our approach provides a unified framework for the automatic semantic annotation of Web services, contrary to manual annotation. It provides a semantic model as well as a method and a library for annotating Web services semantically. Specifically, we have worked on semantic annotations of RESTful services and WFS services.

3.1 Thesis Objectives

The general objective of the thesis is to propose a semantic annotation process using external resources (such as ontologies, datasets, etc.). To fulfill this overall goal, we have decomposed it into the following conceptual and technological objectives:

Conceptual Objectives

O1. The definition of a conceptual model for annotating Web services. We propose a model that contains a set of classes and properties which are used to describe RESTful or OGC WFS services.

O2. The definition of a process for the semantic annotation of services. We develop a novel method that encloses a set of steps for annotating services in an automatic manner.

Technological Objectives

O3. The creation of a system that implements the proposed annotation process.

O4. The definition of a gold standard for evaluating annotation processes for RESTful and OGC WFS services and the development of a system to evaluate the quality of annotation process.
3.2 Contributions to the State of the Art

We rely on the use of a set of external semantic resources like DBpedia and GeoNames, and of external third-party services like suggestion and synonym services. As a result of this process we are able to produce lightweight semantic annotations of service inputs and outputs. These annotations can be later serialised into any of the frameworks described in the state of the art chapter (OWL-S, WSMO, WSMO-lite, SAWSDL, hREST, SA-REST, microWSMO, etc). We have tried to provide solutions to some of the open research and technological problems (see Chapter 2) identified in the scope of this thesis.

C1. **A model for representing a Web service.** This model allows 1) describing semantically RESTful and WFS services and 2) including data (instances) related to the execution of the Web service. The model is described in Chapter 4.

C2. **A method for annotating services using ontological and external resources.** A general method is the result of our investigation and analysis of the semantic annotation process. The description of this method is included in Chapters 4, 5, and 6.

C3. **A system that implements the annotation process.** In this way, the software library covers the annotation of Web services. This library can be applied to RESTful and OGC WFS services.

C4. **A gold standard for evaluating the semantic annotation of RESTful and OGC WFS services.** A gold standard generated with semantic annotations of a representative set of RESTful and OGC WFS services and algorithms for evaluating the annotations.

The contributions are presented in the document as follows: Chapter 4 presents the following contributions: (C1) a model for representing a Web service; (C2) a method for annotating a Web service. Then chapters 5 y 6 present (C3) our library for annotating Web services in the context of RESTful and OGC WFS services; Finally, Chapter 7 presents (C4) the gold standard and algorithms for evaluating the annotations.
3. OBJECTIVES AND CONTRIBUTIONS

3.3 Assumptions

The work described in this thesis is based on a set of assumptions described next:

A1. The external resources to be used are freely available and with no restriction of use.

A2. The quality of the annotation generated can be measured as the similarity value of the annotation generated against gold standard annotations. This gold standard has been created by domain experts.

A3. Web services are freely available and can be invoked.

A4. We only consider RESTful and OGC WFS services for the annotation process.

A6. There is a system capable of discovering new services, or users are able to find these services manually, or by means of search engines.

A7. We only deal with services that provide structured data in XML, even though, it may not always be obvious how best to flatten that data into a set of relational tuples.

A8. We assume that the set of domain ontologies currently available on the Web suffices for annotating the service to be modeled. This is conditioned to the existing knowledge in the ontologies used. For instance, consider the case where the ontology model only contains relations that are useful for describing geospatial data, and the new service provides biological data. Obviously, it will not be possible to annotate a service that cannot be described using the ontologies available, unless new ontologies are created for this purpose and domain. In summary, in our work we do not deal with the creation of vocabularies or ontologies to be used for annotation.

3.4 Hypotheses

Once the assumptions have been identified, the hypotheses of our work are described. These hypotheses cover the main features of the solutions proposed.
3.5 Restrictions

H1. The reuse of external resources, which have reached some degree of consensus in the community, will allow the development of semantic annotations of Web services in an easier and faster way.

H2. It is possible to define a unified method for annotating both RESTful and OGC WFS services.

H3. The method for annotating Web services is extensible and applicable to several types of Web services, independently of its data model or implementation. It can be applied to widely used Web services, such as WS-*, RESTful and OGC WFS.

H4. The method proposed can be implemented in a system that facilitates the automatic creation of semantic annotations when the Web service is available.

3.5 Restrictions

Finally, there is a set of restrictions that defines the limits of our contributions and established future research objectives. These restrictions are the following:

R1. Our analysis of Web services focuses on RESTful and OGC WFS services, not covering WS-*.

R2. The method for annotating Web services covers the annotations, but it does not consider the definition of a methodology.

Table 3.1 summarizes the mapping between the objectives identified in section 3.1 and the specific contributions. This table also summarizes for each contribution, the associated assumptions, hypotheses and restrictions of the thesis.
### 3. OBJECTIVES AND CONTRIBUTIONS

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Contributions</th>
<th>Assumptions and Hypotheses</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>O3. Creation of a system that implements the annotations process.</strong></td>
<td>C3. A system that implements the annotation process</td>
<td>A1, A3, A4, A5, A6</td>
</tr>
<tr>
<td><strong>O4. Definition of a gold standard for evaluation annotations and development of system to evaluate the annotations.</strong></td>
<td>C4. A gold standard for RESTful and OGC WFS services and algorithms for evaluating the annotations.</td>
<td>A1, A2, A3, A5, A6</td>
</tr>
</tbody>
</table>

**Table 3.1:** Mapping between objectives and contributions with associated assumptions, hypotheses and restrictions
A General Model and Method for the Semantic Annotation of Web Services

As described in Chapter 2, syntactic descriptions of services can be enriched with ontology-based annotations that may follow different semantic annotation frameworks, such as OWL-S, WSMO, WSMO-lite, SAWSDL, hREST, SA-REST, microWSMO, etc. These frameworks specify how these annotations have to be structured and encoded, and make no assumptions about whether the annotation process is manual, automatic or semi-automatic.

In this chapter we describe our proposal to automate the semantic annotation of Web services. We rely on the use of a set of external semantic resources like DBpedia and GeoNames, and of external third-party services like suggestion and synonym services. As a result of this process we are able to produce lightweight semantic annotations of service inputs and outputs. These annotations can be later serialised into any of the aforementioned frameworks (OWL-S, WSMO, WSMO-lite, SAWSDL, hREST, SA-REST, microWSMO, etc).

First of all, we will describe the theoretical foundations underpinning the semantic annotation of services. Then, we will have a section that formalises the main definitions used in this chapter. The following section describes the proposed service description model. The last section explains the process of service annotation. Prototypical implementations of this process for services of type RESTful and WFS will be presented in
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

chapters 5 and 6 respectively.

4.1 Preliminaries

In this section we define the basic concepts that we will be using throughout the rest of the thesis, namely the concepts of service, annotation, annotation units, and semantic annotation.

4.1.1 Basic definitions

Definition 1 (Service). A service is a well-defined unit of work performed by a component and packaged for easy access [Bro08].

Definition 2 (Annotation). The term annotation implies, very generally speaking, to attach data to some other piece of data [OMS+06]: it establishes, within some context, a relation between the annotated data and the annotating data.

Definition 3 (Annotation Units). An annotation unit [PDM08] is a triple < s, p, o >, where s represents the subject being described (E.g., service parameters), p represents a property that describes it (E.g., type), and o represents a describing object or value (class or property value).

Definition 4 (Semantic annotation). A semantic annotation SM is a set of one or more annotation units. The term Semantic Annotation refers to a process as well as the outcome of such process. It refers to i) the process of adding semantic data or metadata to the content, given an agreed ontology, and ii) the semantic data or metadata itself as a result of this process [Han05].

4.2 Definitions - Problem formulation

Before defining the model and process to provide semantic annotations of Web services, we introduce some concepts and notations that we will use throughout this chapter.

4.2.1 Service URLs

The URL U of a service constitutes a reference to it, and serves as the identifier of the service.
4.2.2 Service Description

Given a service identified by its URL $U$, we formally define the service as the following tuple:

$$S = < \text{Schema}, \text{InstanceSet} >$$

where $\text{Schema}$ represents the schema of the service and $\text{InstanceSet}$ represents the instances of the service.

The schema of the service, $\text{Schema}$, is defined as:

$$\text{Schema} = < M, OPT, I, O >$$

where:

- $M = \{ m_1, m_2, \ldots, m_n \}$ is a set of methods (actions, operations), where every $m_i$ can be represented as a literal. For example:
  $$M = \{ \text{getCountry}, \text{getInfo}, \ldots, \text{getFeatureCity} \}.$$

- $OPT = \{ opt_1, opt_2, \ldots, opt_n \}$ is a set of options that describe some non-functional characteristics of a Web service. For example:
  $$OPT = \{ \text{keywords}, \text{user}, \ldots, \text{password} \}.$$

- $I = \{ i_1, i_2, \ldots, i_n \}$ is a set of service input parameters with $i_i = <\text{label}, \text{value}>$ where label is a literal and $value = \{v_1, v_2, \ldots, v_n\}$
  For example, $I = \{ <\text{countryCode}, \text{ES}>, <\text{lat}, 15>, \ldots, <\text{lng}, 20> \}$.

- $O = \{ o_1, o_2, \ldots, o_m \}$ is a set of service output parameters, with $o_i = <\text{label}, \text{value}>$, where label is a literal and $value = \{v_1, v_2, \ldots, v_m\}$
  For example:
  $$O = \{ <\text{countryName}, \text{Spain}>, <\text{city}, \text{Madrid}>, \ldots, <\text{continent}, \text{Europe} > \}.$$

- $m_i \subseteq I \times O$, the cartesian product of $I$ and $O$, is a set of ordered pairs $(i, o)$, where $i \in I$ and $o \in O$. For example:
  $${\{\{\text{countryCode}, \text{countryName}\}, \{\text{countryCode}, \text{city}\}, \{\text{countryCode}, \text{continent}\}\}, \{\{\text{lat}, \text{countryName}\}, \{\text{lat}, \text{city}\}, \{\text{lat}, \text{continent}\}\}, \{\{\text{lng}, \text{countryName}\}, \{\text{lng}, \text{city}\}, \{\text{lng}, \text{continent}\}\}}$$
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

The content of the \textit{InstanceSet} is defined as:

\[ \text{InstanceSet} = \langle M, OPT, I, O, Inv \rangle \]

where

- The \(M, OPT, I,\) and \(O\) sets, as defined previously.
- A \(Inv : I \rightarrow O\) function. This function called invocation service allows populating InputValue and OutputValue classes.

4.2.3 Ontology and SPARQL

Based on the definition provided in [ES07], we can define an ontology \(O\) as the following tuple:

\[ O = \langle OS, KB \rangle \]

where \(OS\) represents the ontology schema and \(KB\) represents the knowledge base.

An ontology \(OS\) is defined through the following tuple:

\[ OS = \langle C, A, R \rangle \]

where:

- \(C = \{c_1, \ldots, c_n\}\) is a finite set of classes.
- \(A = \{a_1, \ldots, a_n\}\) is a finite set of attributes, where every \(a_i \subseteq C \times \text{Literal}\).
- \(R = \{r_1, \ldots, r_n\}\), a finite set of binary relations, where every \(r_i \subseteq C \times C\).

A knowledge base is a structure:

\[ KB; = \langle C, A, R, Ins, t_C, t_A, t_R \rangle \]

where:

- three sets \(C, A,\) and \(R\) as defined before.
- a set \(I = \{i_1, \ldots, i_n\}\) whose elements are called instance identifiers.
- a function \(t_C : C \rightarrow Ins\) called class instantiation.
- a function \(t_A : A \rightarrow Ins\) called attribute instantiation.
- a function \(t_R : R \rightarrow InsxIns\) called relations instantiation.
4.2.4 External Terminological Resources

We formally define an External Resource with the following tuple:

\[ ER = < EI, EO > \]

where:

- \( EI = \{ e_1, e_2, \ldots, e_n \} \) represents a set of terms.
  
  For example, \( EI = \{ \text{countryName}, \text{countryCode}, \text{country\_name}, \text{lat}, \ldots, \text{lng} \} \).

- \( EO = \{ e_1, e_2, \ldots, e_n \} \) represents a set of terms depending on \( EI \), with
  
  \( eo_i = < \text{label}, \text{value} > \) where label is a literal and \( \text{value} = \{ v_1, v_2, \ldots, v_n \} \)

  For example, \( EO = \{ \text{country, name} \}, \{ \text{country, code} \}, \{ \text{country, name} \}, \ldots, \} \).

4.2.5 Special Parameters

We formally define Special Parameter as follows:

\[ EP = < SP, SPV > \]

A Special parameter \((SP)\) is a possible infinite set of values \( \{ ep_1, ep_2, \ldots, \} \), which constitutes the set of values for variables. Value Special Parameter denoted \( VSP \), is a set of values \( \{ vsp_1, vsp_2, \ldots, \} \) that may contain the special parameter.

4.3 A Model for Service Description

As we described in the state of the art, there are reasonably well-established approaches for the semantic annotation of Web services (OWL-S, WSMO, etc.). All these approaches consider their own service description model, normally focused on describing characteristics of the inputs and outputs of the services, and sometimes their relationships. For these reasons, we propose a unified model (Figure 4.1) based on these models, and we add the possibility of describing the Web service response and the ontological instances with which such result can be annotated to improve the description of Web service parameters.

That is, in our approach we propose to abstract away from these specific standards in order to proceed with annotation. Our approach proposes the use of the Object Oriented (OO) model in order to describe Web services syntactically and semantically. The
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

OO modeling, a classical modeling method, is used to produce a conceptual schema or semantic data model of a system, often stored in a relational database. This model has been used successfully as a standard in object oriented modeling for decades. Namely, most of models, which are now being upgraded to web-based systems, are based on OO.

The main contribution of our work consists of a model for Web services and a method for annotating service descriptions in a machine-processable format so that their use can be automated. Figure 4.1 shows a UML-based model with the classes that we consider as necessary to represent a Web service syntactically.

4.3.1 Syntactic description of a Web service

We distinguish the schema, where we store the structure of the service, and its instantiation, where we store instances (information) of the service. This model is composed of the following classes:

- The Web service is represented as a class called Source. The Source class is the central point for the description of the elements that address the desired service itself. It holds references to classes that consider different aspects of the service description.

- Method (schema) is used to describe Web service operations.

- Input (schema) represents the parameters that the Web service expects and the format in which they need to be represented.

- Output (schema) represents the outputs of an operation.

- Invocation (instance) allows registering a complete Web service invocation. Invoking a Web Service involves passing messages between the client and the server. It specifies how we should format requests to the server, and how the server should format its responses.

- InputValue (instance) specifies the request value used by the method to invoke the Web service.

- OutputValue (instance) registers the response generated by the method invocation.
4.3 A Model for Service Description

Figure 4.1: Model - Syntactic and Semantic description model
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

4.3.2 Semantic description of a Web Service

In order to describe a Web service semantically, we define a model to represent the relationships of the different service and method parameters with diverse resources used for semantic annotations as illustrated in figure 4.1. This model is based on the service description approaches defined in the state of the art. Moreover, our proposed model considers a list of possible links and considers values (instances) associated with different parameters as additional information about Web services.

This model is composed of the following elements:

- **Parameter.** This class provides a list of all the parameters collected from different services. The relation `hasCollection` relates Parameter with OntologyElement. Every parameter can be related to any number of classes or properties (from 0 to N).

- **OntologyElement.** This class contains classes and properties of the used ontologies related to parameters of each service. This class is related to the class instances by means of the relation `hasCollection`. Ontologies can be related to any number of instances (from 0 to N).

- **OntologyInstance.** This class collects values from a SPARQL Endpoint, where a parameter may have one or more associated resources.

Thus for example, if we have a Web service that contains a parameter called `country` and we are using the DBpedia ontology (this ontology has classes called `country`, `countryName`), then following our model, the `country` parameter will have a list of `OntologyElement` (in this case: country, countryName). Furthermore, each `OntologyElement` has a list of `OntologyInstances`. For example, the `Country` class has a list of `OntologyInstances` such as: Spain, Ecuador, France, etc.

For retrieving instances of a `OntologyElement`, we can use a SPARQL query, like the following:

**Listing 4.1:** SPARQL query for retrieving instances of a class

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?val where { ?val a ?c }
```
4.4 Semantic Annotation Process

Listing 4.2: SPARQL query for retrieving instances of a property

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

Listing 4.3: SPARQL query for retrieving instances of a class/property

```
SELECT DISTINCT ?instance
WHERE {
    {
        ?object <ClassOrPropertyURI> ?instance FILTER isLiteral(?instance)
    } UNION {
        ?object <ClassOrPropertyURI> ?resource.
        ?resource rdfs:label ?instance
    }
} ORDER BY RAND() LIMIT InstanceNumber
```

4.4 Semantic Annotation Process

In this section we present a description of our approach for the semantic annotation of services. The aim of our solution is to process services and generate their enriched description models based on several resources, such as ontologies, external resources and technical information provided by each service. Figure 4.2 depicts a general schema of the proposed solution. The following sections give a deeper description of the components used in our approach.

4.4.1 Step 1. Search for services

Services are normally located (and executed) in web servers from the organization that is offering them. These services may or may not be registered in service registries (e.g., programmableWeb.com for RESTful services, SEEKDA.com for WS-*, fmepe-dia.safe.com for OGC services), and hence these service registries are not necessarily exhaustive. Furthermore, they are not necessarily up to date, and may contain descriptions of services that do not exist anymore or that are already outdated.

When a user needs to find a service that is relevant for the purpose that he/she is looking for, there are several possibilities: the user may search in the aforementioned specialised registries, or she/he may send general keyword-based queries to general purpose search engines (e.g., Google, Bing, etc.) with the aim of finding those services. Both approaches share the limitation of being too focused on keywords, what makes these searches inaccurate in general, especially for complex tasks.
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

Figure 4.2: Service - syntactic and semantic process
4.4 Semantic Annotation Process

4.4.2 Step 2. Analyse, invoke and register services

Once a relevant service has been discovered (we refer to it by its URL $U$), we can extract its main characteristics: what type of data it requires as input, what type of data it produces as output, extra metadata, how to invoke it, etc. Invoking a service by hand is a laborious process, so automating this process as much as possible makes sense.

A number of Web service standards have been developed for describing interfaces and providing access to the data. As described in chapter 2, the two most common protocols used for accessing Web services are SOAP, where both the inputs and outputs of the service are encoded in XML, and REST, where the input attributes are encoded in the URL and the output is an XML or JSON document. In addition, OGC Web services are a special type of Web service that manages geospatial data and these services are encoded in XML.

For SOAP-based services the interface definition language is called WSDL, while for REST-based services the most important interface definition language is called WADL. In the case of OGC-based services the interface definition is called capabilities. Whatever the standard, these languages allow service providers to describe syntactically operations that they provide in terms of which input each operation expects and which output it will produce.

Furthermore, services use different approaches for modeling and representing the data that they consume as input or that they produce as output. Examples of these include JSON, HTML, XML, GML, etc. In general, it is normally easier to integrate services that share the same data model.

When a user knows what service she/he needs, the user needs to invoke the service to view the result and read these results to understand the meaning. So, the next question is: how can I invoke this service?. The invocation process in our approach is performed as follows: first, it takes the input parameters and their values, which are given to the service as part of a URL. Then, the system invokes the service, which translates our "service call" into a specific query to the service, including the URL and related parameters. In our work, we use only the XML format response for describing a service. These XML responses are processed using an API, which enables navigating through the XML and extracting output parameters of each service. The result of this
4. A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

Invocation process is a syntactic definition of services in XML, which can be expressed using a UML model or stored into a relational model. In this work we use a relational model as our data model.

In table 4.1 we show some characteristics of the main types of Web services. For example in order to call a WS-* service, we can use a SOAP client; to call an OGC service, we may use Geotools; and, we normally use HTTP connections for REST services.

<table>
<thead>
<tr>
<th>Web Service</th>
<th>Definition Language</th>
<th>Output format</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS-*</td>
<td>WSDL</td>
<td>XML (SOAP)</td>
<td>SOAP client</td>
</tr>
<tr>
<td>RESTful</td>
<td>WADL</td>
<td>XML, JSON, html, xhtml, text, csv</td>
<td>HTTP, specific client</td>
</tr>
<tr>
<td>OGC</td>
<td>GML</td>
<td>GML</td>
<td>geotools, FDO</td>
</tr>
</tbody>
</table>

Table 4.1: Web service characteristics

4.4.3 Step 3. Selecting ontologies

Before performing a semantic annotation, it is necessary to know the domain of services that we need to annotate, that is, we need to know more information about this service.

An important issue in the selection of ontologies is the availability of a SPARQL Endpoint that contains instances from them. This is a requirement associated with our evaluation process, as we describe in chapter 6.

4.4.4 Step 4. Include additional resources

In order to increase the expressivity of our model and, to enrich service annotation, additional resources are used in our process. The selected resources are related to the domain and parameters of services. Figure 4.3 shows the designed model to relate external resources with other classes of our model.

The access to external resources is made by using an API or REST client. The obtained response is often an XML document, which is processed to obtain required data. All collected data are stored into the database, which allows maintaining a cache for querying about data.
4.4 Semantic Annotation Process

Examples of external resources that can be used are suggestions, synonyms, etc. Suggestion services allow retrieving a list of words for parameters with spelling mistakes and synonym services allow retrieving possible synonyms for a certain parameter. In table 4.2, we show examples of external resources.

4.4.5 Step 5. Detect Special parameters

After an analysis of different kinds of Web services (RESTful, WFS OGC, WS-*), we have discovered with our research that some service parameters are useless in terms of the semantic annotation processes. Thus, these special parameters entail the lack of possible annotations that we can make with them or the lack of meaningful results that we can obtain from them. After an exhaustive analysis performed in this research, we classify these parameters into three groups:

![Diagram of External Resource (ER), Input Term, and Output Term]

**Table 4.2: External Resources**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Suggestion</th>
<th>Synonym</th>
<th>GeoSpatial</th>
</tr>
</thead>
<tbody>
<tr>
<td>countryName</td>
<td>country name</td>
<td>area</td>
<td>-</td>
</tr>
<tr>
<td>continent</td>
<td>continent</td>
<td>area</td>
<td>-</td>
</tr>
<tr>
<td>lat</td>
<td>latitude</td>
<td>latitude</td>
<td>latitude</td>
</tr>
</tbody>
</table>
• Navigation parameters. Those parameters refer to the navigation process through the service results, for example: link, userID, hits, total, pages, etc.

• Unknown parameters. Those parameters are named with only one character (e.g., q, l, s, etc.), and hence not sufficiently descriptive for our automated approach to find any correspondence.

• Geospatial parameters. Those parameters refer to the geospatial domain, for example: lat, lng, bbox, etc.

For example, REST services provide lists of results and also support, in general, pagination of those results. Rather than receiving as many as tens or hundreds thousands of records upon a GET request for collection objects, one will instead receive a list containing a smaller subset of those records, and have the ability to page through the entire list of records. Thus, users can control how many records are returned per page and which page of records to retrieve, by adding query parameters to their GET request. Something similar happens with WS-* and OGC services. Further details are described in section 6.2.3.

We propose a model to represent these special parameters. The SpecialParameter represents a parameter of the Web service and ValueSpecialParameter contains a list of possible values that SpecialParameter might contain. This model is persisted into the relational database to maintain a cache of special parameters and improve the semantic annotation process.

4.4.6 Step 6. Match service properties with ontology concepts

In this step, we need to inspect parameters that are concepts or instances of an ontology, and also explain a special meaning such as: lat, lng, latitude, etc. So, after analyzing the parameters of the Web service, we send the parameters one by one to the ontology for determining their concepts.

At this stage, the list of syntactic parameters, which are obtained in section 4.4.2 and represented using the model described in section 4.3 is used to query the SPARQL Endpoints that have been selected and to retrieve associated results for each parameter as follows: First, the approach suggests retrieving all classes from ontologies whose names have a match with any parameter of the Web service. In this matching process we recommend two different techniques:
4.4 Semantic Annotation Process

- We use an exact match to compare the Web service parameter name with the labels of the ontology classes and properties.

- We propose using a combination of various similarity metrics (Jaro, JaroWinkler and Levenshtein metrics) to compare parameter names with labels of the elements of these ontologies. This proposal allows matching strings like CountryName, country name or country, for example.

If there is a correspondence from the matching process, we use these concepts individually to retrieve samples (concept instances) from the ontology SPARQL Endpoints. Likewise, when a parameter matches an ontology class related to some special parameters, our approach retrieves samples from the special SPARQL Endpoint or from our repository.

Next, the approach suggests to find correspondences between parameters of the Web service and ontology properties. If there is some correspondence, it uses these ontology properties individually to retrieve information of SPARQL Endpoints of the ontologies, as described above. Furthermore, this information (URI) is registered as a possible correct link value for the corresponding parameters.

Finally, with the obtained classes and properties (possible annotations), the approach suggests calls to the ontology SPARQL Endpoints to retrieve values (instances).

In our approach, we obtain all parameters and they are compared with used ontology concepts. In this step, we assume that the Web service $S$, ontologies $Ont$ and $SPARQLx$ are the inputs and we want to find matches between parameters and ontology concepts. For that, we propose algorithm [1].

We formally define a matching process between service parameters and ontological concepts as the following tuple:

$$ P = < S, OS, SPARQLx, t_{pi}, t_{po} > $$

where:

- The $S$, $OS$, and $SPARQLx$ sets are defined before.

- A function $t_{pi} : P \rightarrow L_i$ called input links, where $L_i$ is defined as $L_i = < I, C > \cup < I, A >$. 

Algorithm 1 match algorithm

Require: term

parameter ← localSearch(term);
if parameter then
    return parameter
else
    parameter.name ← term
loadOntologies(Ont \(1\), Ont \(2\),..Ont \(n\));
\(C_{Ont_1}\) ← retrieve\(C_{Ont_1}\)(term)
for all class ∈ \(C_{Ont_1}\) do
    ontologyResource.link ← class
    ontologyResource.type ← ‘class’
    parameter.addOntologyResource(ontologyResource)
\(I_{Ont_1}\) ← retrieve\(I_{Ont_1}\)(class)
for all instance ∈ \(I_{Ont_1}\) do
    ontologyInstance.value ← instance
    ontologyResource.addInstance(ontologyInstance)
end for
end for
\(P_{Ont_1}\) ← retrieve\(P_{Ont_1}\)(term)
for all property ∈ \(P_{Ont_1}\) do
    ontologyResource.link ← property
    ontologyResource.type ← ‘property’
    parameter.addOntologyResource(ontologyResource)
\(I_{Ont_1}\) ← retrieve\(I_{Ont_1}\)(property)
for all instance ∈ \(I_{Ont_1}\) do
    ontologyInstance.value ← instance
    ontologyResource.addInstance(ontologyInstance)
end for
end for
Same process for all ontologies
end if
return parameter
4.4 Semantic Annotation Process

- A function $t_{po}: P \rightarrow L_o$ called output links, where $L_o$ is defined as $L_o = < O, C > \cup < O, A >$.

4.4.7 Step 7. Enrich parameters with external resources

In the previous step we described semantically the list of parameters. However, sometimes this is not sufficient, due to the lack of match between concepts and parameters. We propose algorithm 2 for increasing the amount of metadata over the original service parameters.

The $S$ obtained in the previous step is processed to find possible additional parameters based on the original. Thus, we use external resources to improve the original definition, for example our approach tries to find synonyms, suggestions, etc. of the original service parameters.

As we said before, our system looks for matches with classes and properties of the ontology. Hence it is possible to have parameters with no correspondences identified, since there are many lexical and syntactic variations that the parameter names may have, and because in some cases the information that is being requested may not be available in any of the external sources that are consulted. In order to annotate semantically the parameters that did not match any ontology concept, we use additional external resources for each parameter, such as suggestion and synonym services. Below we describe algorithm 2 for external services (suggestions and synonyms) that we consider.

We formally define functions to realize the match between Web service parameters and external resources as follow:

- A function $t_{match^i}: I \rightarrow EI$ called input external resource.
- A function $t_{match^o}: O \rightarrow EO$ called output external resource.

4.4.8 Step 8. Match the new enriched parameters with ontologies

With this extended model populated, the Input/Output parameters may have a list of terms related to some external resources, for example synonyms, suggestions, etc. In this step, we propose to use an algorithm 1 to find possible matches between the external resources and ontological concepts, as described in subsection 4.4.6.
Algorithm 2 genericAlgorithmSuggSyn

Require: $t$, parameter

term ← localSearch($t$);
if term then
    return term
else
    listSuggestions ← sug($t$)
    if listSuggestions == null then
        $t ←$ deleteSpecialCharacters($t$
        listSuggestions ← sug($t$
    end if
    for all $s ∈$ listSuggestions do
        suggestion.name ← $s$
        term.addSuggestion(suggestion)
        parameterSuggestion.name ← $s$
        parameter.addSuggestion(parameterSuggestion)
        processAlgorithmOntologyResource(parameterSuggestion);
        listSynonyms ← syn($s$
        termSugg.name ← $s$
        listSynonyms ← syn($s$
        for all $sy ∈$ listSynonyms do
            termSugg.addSynonyms($sy$
            parameterSynonym.name ← $sy$
            processAlgorithmOntologyResource(parameterSynonym);
        end for
    end for
    listSynonyms ← syn($t$
    for all $sy ∈$ listSynonyms do
        term.addSynonyms($sy$
        parameterSynonym.name ← $sy$
        processAlgorithmOntologyResource(parameterSynonym);
        parameter.addSynonyms(parameterSynonym)
    end for
    return term
end if
4.4 Semantic Annotation Process

We formally define a matching process between service parameters (including External Resources) and ontological concepts as the following tuple:

\[ PER = < S, OS, SPARQLx, t_p^i, t_p^o, ER, t_{match}^i, t_{match}^o > \]

where:

- The four \( S, OS, SPARQLx \), and \( ER \) sets were defined before. Furthermore, the \( t_p^i, t_p^o \) functions were defined previously.
- A function \( t_{match}^i : match \rightarrow L_i \) called input links, where \( L_i \) is defined as \( L_i = < I, C > \cup < I, A > \).
- A function \( t_{match}^o : match \rightarrow L_o \) called output links, where \( L_o \) is defined as \( L_o = < O, C > \cup < O, A > \).

4.4.9 Step 9. Check the Semantic Annotation of Web Services

In order to check collected sample individuals and the initial semantic annotations obtained as a result of the previous process, our approach suggests invoking the Web services that were already registered in the repository using our model and validating the input and output parameters for checking which is the best option to describe each parameter.

We formally define functions to realize the validation of semantic annotations between Web services and ontological resources as follows:

- A function \( t_i : L_i \rightarrow EI \) called input validation.
- A function \( t_o : L_o \rightarrow EO \) called output validation.

4.4.9.1 Input Validation

In order to validate input parameters, our approach selects, for each parameter, a random subset of the example instances (associated with classes and/or properties) coming from the ontologies that we have obtained and registered before. Next, it makes several invocations of the Web service iterating over these registered values. The system does not check this with all the possible combinations of collected instances for all parameters for two reasons: first, because of the combinatorial explosion that may
be produced in such a case, and second because many Web services have invocation limitations.

When a service has one or more than one input parameter, the system obtains randomly some instances of this parameter for the validation process. Each parameter generates a collection (list) of instances from our repository. Then, the system joins instances to obtain a table of all combinations of each parameter. If the service returns results from the invocation, then the service is considered as executable, and the corresponding annotations are marked as valid. If a service cannot be invoked successfully, the service is classified as non-executable and is automatically discarded from the list of services that can be automatically annotated.

4.4.9.2 Output Validation

For the validation of the output parameters, our approach only takes into account executions with the correct inputs from the input sets that have been considered before. Next, the system compares the outputs obtained after execution with the information already stored in the repository due to the initial retrieval processes done before. If the output can be matched, our system considers that the output annotation is valid.

4.4.10 Step 10. Publish semantic annotations

The correspondences established between the different parameters of the Web service and the ontologies are registered and stored in the repository, so that they can be used later. In such a way, the Web service is annotated semantically and it allows generating semantic descriptions or annotations of any of the types that were identified in the state of the art section (OWL-S, WSMO, SAWSDL, WADL, hREST, SA-REST, etc.).

4.5 Minimal Service Model vs Our Approach

As we described in 2.3.2.1, MSM was designed to capture the core of both classical Web services and Web APIs in a common model.

In table 4.3, we summarize the main characteristics of the Minimal Service Model compared with our proposed approach. We compare the annotation models on the following attributes:
4.6 Conclusions

- Schema is the model that has been generated for describing a Web service.

- Invocation is a class extension that we propose to register the automatic invocation of a Web service.

- InvalidInvocation is a class extended for registering the fact that the Web service has some error.

- Possible links is a relation between different Web services.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>MSM</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Invocation</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Invalid invocation</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Possible links</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 4.3: Models comparison

In our extension of the allowed values for the type of service, we introduce a property named typeService, whose value is a literal taking one out of three choices: SOAP, RESTful or OGC. Also, we introduce the Invocation, InputValue and OutputValue classes; they represent the value of calling a Web service. Moreover, we introduce the InvalidInvocation class that allows responses registering including errors. The extended model is illustrated in figure 4.4.

4.6 Conclusions

In this chapter we have shown a novel way to semantically describe Web services (RESTful and WFS) without interfering with the regular service operation and by not introducing a new resource that would have to be maintained along the service. This innovation gives Web services much of the power than traditional SOAP-based descriptions services have.

The detailed process in this chapter provides the set of steps to be followed in order to automate the process of semantic annotation of the aforementioned Web services. Likewise, our process defines a data model for managing both, the syntactic and the
A GENERAL MODEL AND METHOD FOR THE SEMANTIC ANNOTATION OF WEB SERVICES

Figure 4.4: MSM extended -

semantic issues. Furthermore, this proposal does not focus only on schemes of services and ontologies, but it allows dealing with data services, which are converted to instances of different used ontologies.

With respect to the annotation process of Web services, our proposal uses a combination of different resources in order to discover meanings for each service parameter. It improves the results of this annotation process and makes easier that a user may select and perform semantic annotations of any kind of aforementioned services.

Finally, we present a comparison between our approach and MSM (Minimal Service Model) in this chapter. This comparison is performed taking into account several attributes (schema, invocation, invalid invocation, and possible links). The obtained results show that our process and the defined model improve description of Web services.
5

RESTful Semantic Annotation Process

As described in the introduction of this document a large number of public and private organizations are generating APIs to allow access to their data, both externally and internally. Nowadays the largest online repository of information about Web 2.0 mashups and APIs is ProgrammableWeb.com. As of December 2014, this aggregator site provide information on 6,962 mashups and 12,504 APIs that were registered between September 2005 and December 2014. Mashups tagged as mapping represent a 33% of the total number of mashups (2,468 mashups), what represents the importance of geospatial information in the generation of REST-based applications. With respect to APIs, GoogleMaps is the most used with a 41%, that is, this API is used on 2,206 mashups. These data show the importance of geospatial information in the context of the RESTful services world.

The previous numbers and the statistics shown in chapter 2 show that RESTful services are gaining traction over WS-* ones. As with WS-* services, their semantic annotation can provide benefits in tasks related to their discovery, composition and mediation.

The reason for this phenomenon is that the current service organization mode mainly concentrates on the implementation and execution of web services from a syntax-based level perspective. It limits the whole service discovery techniques to keyword-based.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

Service description of RESTful services capabilities is essential for classifying, discovering and using a service. We need to consider functional (e.g. what a service does, sequencing of messages) as well as non-functional (e.g. service category, security, authentication and privacy) attributes of the service. Furthermore, RESTful services need to be understandable by humans as well as by machines. This means that each service attribute must be described in both a syntactic and a semantic level. Syntactic descriptions are concerned with the implementation aspects of a RESTful service and thus custom towards the programmers’ requirements. Semantic descriptions are concerned with the conceptual aspects of a RESTful service aiming to facilitate end-users the understanding of it by shielding off the lower-level technical details, as well as to facilitate developers to find services that best match their needs and to enable automatic service selection and composition. Therefore, by adopting domain ontologies as the provider of the semantic context, we can develop different new approaches to annotating syntax-based RESTful services and converting them into semantically described web services.

Figure 5.1: RESTful services - An overview of the semantic annotation system

In this chapter, we discuss the problem of syntactic registration and semantic annotation of RESTful services using the following resources: A cross-domain ontology like DBpedia, two domain ontologies like GeoNames and the WGS84 vocabulary, and additional external resources like suggestion and synonym services. Furthermore, we
describe the main components of the process of syntactic registration and semantic annotation. Finally, we show that an approach that invokes the RESTful service frequently to compare values works better than a manual annotation.

We must note that this process is the implementation of the process presented in Chapter 4 applied to RESTful services.

5.1 An automated approach for the semantic annotation of RESTful services

In this section, we present our approach, which is summarized graphically in Figure 5.1 for automating the annotation of RESTful services. Our system consists of three main components, including the invocation and registration components, which are in charge of the syntactic description of the service, the repository, which stores service descriptions, and the semantic annotation components, which make use of diverse external resources. Next, we briefly describe these components, illustrating the descriptions with some sample services in the geospatial domain.

The following services, taken from the ProgrammableWeb site, are two representative geospatial RESTful services:

- **Service 1.** http://ws.geonames.org/countryInfo?country=ES. This service retrieves information related to a country. More specifically, it returns information about the following parameters: capital, population, area (km2), and bounding box of mainland (excluding offshore islands). In the specified URL, we retrieve information about Spain (ES). This service is considered as a type of service with parameters as described in section 2.2.2.2.

- **Service 2.** http://api.eventful.com/rest/venues/search?app_key=p4t8BFcLDtCzpxdS&location=Madrid. This service retrieves information about places (venues). More specifically, given a city name it returns parameters like: venue_name, region_name, country_name, latitude, longitude, etc. In the specified URL, we retrieve information about Madrid. XML response of two sample RESTful services. This service is also considered as a type of service with parameters as described in section 2.2.2.2.

These examples will from now on serve as an illustration for our findings.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

5.2 Syntactic description of RESTful services: Registering invocation details in a repository

There are different manners to encapsulate the HTTP request. Below, we list some of them:

- HTTP request. Using any programming language, we can invoke this. Furthermore, we can use a browser to call this.

- Using an API, that is specific to each group of services. Generally, each site gives support to a set of programming language, such as Java, PHP, Phyton, etc.

- Using a general API, which allows access in a transparent way. This type of API abstracts the process to call. For example, Service Data Object (SDO), SoapUI, etc.

As aforementioned, RESTful services are normally registered in sites like programmableWeb by means of their URLs. Besides, they are described with some natural language descriptions, tags, and execution examples. Hence, the first step in our approach is to take as input the URL of a RESTful service (for instance, it has been discovered by a user by browsing this site, it has been maybe sent to the user by a friend, or it has been crawled automatically) and generate a syntactic service description out of it.

Next, we will first describe a service model for a RESTful service and then the annotation process that we propose.

5.2.1 Syntactic annotation process

Given the URL of a service, we invoke the RESTful service with some sample parameters, obtained from those examples that are normally provided together with the URL (if this information is not available, our system cannot continue without further human intervention), and analyzes the response to obtain a basic syntactic description of the input and output parameter set.

The service invocation of a RESTful service may return diverse formats (e.g., JSON, XML, etc) as described in section 2.2.2.2. Here we show how we handle XML responses, which are represented in a structured manner and can be easily consumed by different
5.2 Syntactic description of RESTful services: Registering invocation details in a repository

technologies. The results of an invocation of our sample services are shown in listings 5.1 and 5.2.

Listing 5.1: Service 1. XML response

```xml
<geonames>
  <country>
    <countryCode>ES</countryCode>
    <countryName>Spain</countryName>
    <isoNumeric>724</isoNumeric>
    <isoAlpha3>ESP</isoAlpha3>
    <fipsCode>SP</fipsCode>
    <continent>EU</continent>
    <capital>Madrid</capital>
    <areaInSqKm>504782.0</areaInSqKm>
    <population>40491000</population>
    <currencyCode>EUR</currencyCode>
    <languages>es−ES , ca , gl , eu</languages>
    <geonameId>2510769</geonameId>
    <bboxWest>−18.169641494751</bboxWest>
    <bboxNorth>43.791725</bboxNorth>
    <bboxEast>4.3153896</bboxEast>
    <bboxSouth>27.6388</bboxSouth>
  </country>
</geonames>
```

Listing 5.2: Service 2. XML response

```xml
<venue id="V0−001−000154997−6">
  <url>http://eventful.com/madrid/venues/laancha−/V0−001−000154997−6</url>
  <country_name>Spain</country_name>
  <name>La Ancha</name>
  <venue_name>La Ancha</venue_name>
  <description></description>
  <venue_type>Restaurant</venue_type>
  <address></address>
  <city_name>Madrid</city_name>
  <region_name></region_name>
  <region_abbr></region_abbr>
  <postal_code></postal_code>
  <country_abbr>ES</country_abbr>
  <country_abbr>ESP</country_abbr>
  <longitude>−3.68333</longitude>
  <latitude>40.4</latitude>
  <geocode_type>City Based GeoCodes</geocode_type>
</venue>
```

87
The resulting XML responses can be simple (e.g., as with Service 1) or complex (e.g., as with Service 2).

In any case, we manipulate this XML response automatically using the SDO [Res07] model. The invocation process is performed as follows: first, we take the input parameters and their values, which are given to the service as part of a URL. Then, the system invokes the service that translates our "RESTful service call" into a query to a specific service, including the URL and related parameters. Finally, responses are processed using SDO, which enables navigation through them and extraction of the output parameters of each service. The final result of this invocation process is a syntactic definition of the RESTful service, which can be expressed in description languages like WADL or hREST microformats, or stored into a relational model.

Table 5.3 shows the different input/output parameters for each of our example services. We can observe by manual inspection that there is some degree of similarity (e.g., countryName and country_name) since they have similar names and return similar types of values (e.g., Spain). These similarities between parameters are dealt later in the annotation process.

Listing 5.3: Syntactic description of our two sample RESTful services

```plaintext
Service 1:
countryInfo (\$country, bBoxSouth, isoNumeric, continent, fipsCode, areaInSqKm, lang
uates, isoAlpha3, countryCode, bBoxNorth, population, bBoxWest, currencyCode, bBoxE
ast, capital, geonameId, countryName)

Service 2:
rest/venues/search (\$location, \$app_key, id, link_count, page_count, longitude, trackback_count, version, page_size, country_abbr, first
item, last_item, page_size, country_abbr2, comment_count, geocode_type
pe, search_time, venue_name)
```

We assume that each RESTful service can be represented as a relational view definition, even if most RESTful services on the Web provide tree structured data in XML. It may not always be obvious how to best flatten that data into a set of relational
5.2 Syntactic description of RESTful services: Registering invocation details in a repository

tuples (indeed in some case it may not even be possible to do so and still preserve the intended meaning of data).

5.2.2 RESTful service model

The RESTful service model plays an essential role in the process of syntactic description. A RESTful service designated as RS, is a sequence of input and output parameters.

**Definition 1** (RESTful service model).

Following our definitions from section 4.2, a RESTful service RS ⊆ S, can be defined as a tuple:

\[ RS = (URL, rsName, I, O) \]

where URL is the address of the service. rsName is the service name, I is the service parameters input set which is defined as \( I = \{i_1, i_2, \ldots, i_n\} \), and O is the service parameter output set, which is defined as \( O = \{o_1, o_2, \ldots, o_m\} \)

Each individual \( I_i \) can be used to invoke a RESTful service.

**Definition 2** (Input parameter model).

In a RESTful service input set \( I = \{i_1, i_2, \ldots, i_n\} \), \( \forall I_i \in I(1 \leq i \leq n) \) its formalization is modeled as a three tuple:

\[ i_i = \{\text{name}, \text{value}, \text{dataType}\} \]

where

- **name** is the input parameter name.
- **value** is the value associated to the name.
- **dataType** is the type of value.

For example (country, ES, String)

**Definition 3** (Output parameter model).

In a RESTful service output set \( O = \{o_1, o_2, \ldots, o_n\} \), \( \forall O_i \in O(1 \leq i \leq m) \) its formalization is modeled as a three tuple:

\[ o_i = \{\text{name}, \text{value}, \text{dataType}\} \]

where

- **name** is the output parameter name.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

- **value** is the value associated to the name.

- **dataType** is type of value.

For example (capital, madrid, String), (lat, 45.8, decimal)

**Definition 4** (Invocation model (Function)).

A call of a RESTful service is a process of invocation and can be designed as

\[ \text{Inv} = (I) \]

where \( I \) is a set of input parameters and \( O \) (after invoke the RESTful service) is a set of output parameters.

\[ \text{Inv}(\text{country} = \text{ES}, \ldots) \rightarrow \text{Inv}(I_1), \ldots \]

**Definition 5.**

The result set produced by the call of RS with \( I \) as parameter over an original service is denoted Result. For example

\[ \text{Inv}(\text{countryName} = \text{Spain}, \text{lat} = 0.34, \text{long} = 45.4, \ldots) \rightarrow \text{Result}(O_1, O_2, O_3, \ldots) \]

### 5.2.3 Invocation and Registration algorithm

The algorithm used to invoke and register a RESTful service takes as input a URL for the new service (found) and uses it to invoke the service. The algorithm describing the overall process is shown in algorithm 3.

The first step in the algorithm is to retrieve the input parameters and invoke the new service with a set of input parameters. This invocation is realized using an API called A. The response of this invocation is an XML file. The XML is processed using SDO to flatten and transform in a view relation (table). With the XML flattened, the algorithm retrieves a set of output parameters. Furthermore, the algorithm retrieves values associated with each input/output parameters. With all of these data, the algorithm creates a structure using the model described in section 5.2.3.1. This model is stored into a relational database.

The next step in the algorithm is to generate a standard format used to represent RESTful services such as WADL or hRESTS.
5.2 Syntactic description of RESTful services: Registering invocation details in a repository

Algorithm 3 Invocation algorithm

Require: URL
Ensure: Syntactic description rs(URL, rsName, I, O)

rs ← 0;
rs.URL ← URL
rs.I ← getInputParameters(URL); \{Extract list of parameters with associated value and data type\}
rs.rsName ← getName(URL); \{The name is the string without parameters\}
xml ← A(I); \{Call the service using API(SDO) A, the result is an xml file\}
rs.O ← getOutputParameters(xml); \{Using SDO, iterate over xml to extract the xml tags with value\}
rs.Inv[1] ← Inv.create(I);
rs.Res[1] ← Res.create(O);
rs ← callWithoutParameters(rs);
wadl ← wadlGeneration(rs);
hRESTS ← hRESTSGeneration(rs);
storeObjectModel(rs);
return rs

5.2.4 Calling without parameters

Once the RESTful service is registered, our system invokes the service without any of its associated parameters to retrieve a list of associated instances that contribute to collect more information in order to help to the annotation process of RESTful services.

For example: \[\text{Service1}. \text{http://ws.geonames.org/countryInfo?}\]

The result obtained is showed in listing 5.4. This XML includes the part of XML related to the original service, namely, the service with country=ES as input parameters. We can see that the value ES corresponds to the countryCode parameter in the XML result. It suggests a possible list of values for all entities with countryCode as input parameter that may be used to call the RESTful service. In this case, we assume that the country input parameter is the same as the countryCode. With this assumption, we obtain a list of values to invoke the service and test if the call is right, thus we obtained for this example a list of values such as ES, EC, PU, FR, etc. that correspond to countryCode attribute. With the total of possible values, we take the 10% of data

\[^{1}\text{This is an example of invocation (Service 1) without any of the associated parameters.}\]
to invoke. If from the 10% of call, the 75% are valid invocations, we can infer that \( \text{country} = \text{countryCode} \) and process the remainder XML to assign as valid input and output.

**Listing 5.4:** XML result: calling without parameters

```xml
<geonames>
  <country>
    <countryCode>AD</countryCode>
    <countryName>Andorra</countryName>
    <isoNumeric>020</isoNumeric>
    <isoAlpha3>AND</isoAlpha3>
    <fipsCode>AN</fipsCode>
    <continent>EU</continent>
    <capital>Andorra la Vella</capital>
    <arealnSqKm>468.0</arealnSqKm>
    <population>84000</population>
    <currencyCode>EUR</currencyCode>
    <languages>ca</languages>
    <geonameId>3041565</geonameId>
    <bboxWest>1.422111</bboxWest>
    <bboxNorth>42.658695</bboxNorth>
    <bboxEast>1.780389</bboxEast>
    <bboxSouth>42.435081</bboxSouth>
  </country>

  <country>
    <countryCode>ES</countryCode>
    <countryName>Spain</countryName>
    <isoNumeric>724</isoNumeric>
    <isoAlpha3>ESP</isoAlpha3>
    <fipsCode>SP</fipsCode>
    <continent>EU</continent>
    <capital>Madrid</capital>
    <arealnSqKm>504782.0</arealnSqKm>
    <population>46505963</population>
    <currencyCode>EUR</currencyCode>
    <languages>es − ES, ca, gl, eu, oc</languages>
    <geonameId>2510769</geonameId>
    <bboxWest>−18.169638</bboxWest>
    <bboxNorth>43.791721</bboxNorth>
    <bboxEast>4.315389</bboxEast>
    <bboxSouth>27.638819</bboxSouth>
  </country>

  <country>
    <countryCode>EC</countryCode>
    <countryName>Ecuador</countryName>
    <isoNumeric>218</isoNumeric>
    <isoAlpha3>ECU</isoAlpha3>
    <fipsCode>EC</fipsCode>
    <continent>SA</continent>
    <capital>Quito</capital>
  </country>
</geonames>
```
5.2 Syntactic description of RESTful services: Registering invocation details in a repository

On the other hand, our system also considers service URLs as http://www.foo.org/weather/Madrid. These services belong to a specific RESTful entity and they are always invoked with their associated parameters.

In this way, the system invokes the service for retrieving a collection of instances (countries) related to the service. The results of this invocation are stored into the object-oriented model using the model described in section 5.2. Thus, this process allows collecting additional information about a service (output parameters and instances), which is registered in our system, and retrieving it for future processes without the need to invoke the original service.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

5.2.5 Generation of Syntactic Description

With the result of the invocation and registration, we can use the result to describe and generate different formats for describing a RESTful service in a syntatic way. In the following subsection we show how the process is performed.

5.2.5.1 Object Oriented Model

We register and store this syntactic description (Listing 5.3) into a repository using an object oriented model. This repository is implemented as a database that is specifically designed to store syntactic descriptions of RESTful services and parameters’ values of invocations. We selected this storage model in order to increase the efficiency of the recovery of the RESTful services. Algorithm 3 uses this model to store into the database. In figure 5.2 the model extended to allow the syntactic description of RESTful services is shown.

![Figure 5.2: RESTful - Model for service description](image)

5.2.5.2 HTML documentation generation

The majority of RESTful services’ descriptions are given in the form of unstructured text in a Web page (HTML). In some cases, there are RESTful services without an associated description or the existing description is really poor. In these cases, our approach generates an HTML description for each service registered in our system. In order to generate these descriptions automatically, our approach uses the SoapUI API\(^1\) (a software tool based on Java and used to test web services) and the result obtained

\(^1\)www.soapui.org
5.2 Syntactic description of RESTful services: Registering invocation details in a repository

with algorithm [3]. In this way, the obtained description supports users and makes it easy to create manually semantic RESTful service descriptions through existing tools (e.g.: Sweet[1]). An example of the obtained HTML description is shown in Figure 5.3. These files are used as the basis to generate other types of descriptions such as hRESTS as described in later sections.

5.2.5.3 Generation of a WADL description

With the URL and with its input parameters, we can generate its WADL description (Syntactic Description), which can be used as the input to the next annotation process. Listing 5.5 shows an excerpt of the WADL description of Service 1. Moreover, the WADL description contains additional information about the main characteristics of parameters, for instance: data type, required, style, default language, etc.

**Listing 5.5: WADL description of service 1**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <doc xml:lang="en" title="1"/>
  <resources base="http://ws.geonames.org">
    <resource path="countryInfo" id="countryInfo">
      <doc xml:lang="en" title="countryInfo"/>
      <param name="country" default="ES" type="xs:string" required="false" style="query"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      <method name="GET" id="Method_1">
        <doc xml:lang="en" title="Method_1"/>
        <request/>
        <response status="0">
          <representation mediaType=""/>
        </response>
        <response status="200">
          <representation mediaType="text/xml" element="geonames"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

5.2.5.4 Generation of an hRESTS description

Using the HTML file generated in section 5.2.5.2 and our model, we can generate syntactic hRESTS descriptions. Listing 5.6 shows a syntactic description using microformats of the aforementioned service (service 1) according to hRESTS.

---

4http://sweet.kmi.open.ac.uk/
Figure 5.3: HTML - Automatic description of a RESTful service (Service 1)
5.3 Semantic Annotation Process

The syntactic description of RESTful services is normally not enough to fully understand them and the whole process of web service discovery at the keyword-based level. According to [MKP09, AMT09], the main difficulties are related to the lack of machine-processable descriptions, which create more work for humans to find a service. Furthermore, there are some problems with documentation, due to the fact that users have to read the service documentation for understanding and using them. With respect to technological issues, WS-* and RESTful semantic annotation approaches are manual and complex.

Taking into account these issues, we rely on techniques and processes for an automatic semantic annotation of RESTful services: a) using only the syntactic description of the services and their input/output parameters, or b) identifying a set of example values that allow the automatic invocation of the service.

In this subsection, we show our proposal to map this I/O set into ontology concepts. The starting point of the semantic annotation process is the list of syntactic parameters obtained previously (a WADL file or the model stored into our relational database). Once the RESTful service is syntactically described with all its identified input and

---

**Listing 5.6:** Annotation of (Service 1) following the hRESTS format

```xml
<...<div class="operation" id="op1">
  <a href="#http://ws.geonames.org#countryInfo">http://ws.geonames.org/countryInfo</a></div>
  <code class="address">http://ws.geonames.org/countryInfo</code>
  <...<h4 id="http://ws.geonames.org#Method1">GET</h4>
  <h6>request query parameters</h6>
  <table>
    <tr><td><strong>country</strong></td>
    ...</tr>
    <tr><td><strong>countryCode</strong></td>
    ...</tr>
  </table>
</...>
```

---

5.3 Semantic Annotation Process
output parameters, we proceed with its semantic annotation. We follow a heuristic approach that combines a number of external services and semantic resources to propose annotations for the parameters as shown in Figure 5.4. Next, we describe the main components of our semantic annotation approach.

![Figure 5.4: RESTful parameters - semantic annotation process](image)

5.3.1 A model for describing RESTful service parameters semantically

In order to describe semantically these services we define a model to represent the relationships of the different service parameters with diverse resources used for semantic annotation. This model is based on the service description approaches used by hREST, SAWSDL, and microWSMO as described in 2.2.2 and 2.3.2. Moreover, our model considers values (instances) associated with different parameters as additional information about services. These instances come from semantic data sources (DBpedia and GeoNames).

Some of the elements of this model are domain-independent, while others are domain dependent (in our examples we use these elements related to the geospatial domain, where we have performed our experiments in order to evaluate the feasibility of our
5.3 Semantic Annotation Process

approach). With respect to the domain-independent component, we use DBpedia as the 
main source of background knowledge for supporting the semantic annotation process. 
This is complemented by the domain dependent component. In the context of our 
examples, we use GeoNames as a source related to geospatial information. This model 
(Figure 6.5), where each parameter has a relationship with input or output of a service, 
defines the following components:

Figure 5.5: Model Parameter - For the description of geospatial RESTful service 
parameters

- **Parameter.** This class provides a list of all parameters (inputs and outputs) 
collected from different services. Likewise, we search for additional information 
for each parameter (such as suggestions and synonyms) for enriching the initial 
description of parameters. The relation `hasCollection` relates `Parameter` with 
`Ontologies`. Every parameter can be related to any number of ontology classes or 
properties (from 0 to N).

- **Ontologies.** This class contains classes and properties of the DBpedia and GeoN-
ames ontologies, and WGS84 vocabulary related to the parameters of each service. This class is related to the classes `DBpediaInstance` and `GeoNamesInstance` by means of the relation `hasCollection`. `Ontologies` can be related to any number 
of DBpedia or GeoNames instances (from 0 to N).

- **DBpediaInstance.** This class collects values from DBpedia, where a parameter 
may have one or more associated resources.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

- GeonamesInstance. This class collects geospatial information related to latitude, longitude, and bounding box parameters from GeoNames that we set up with information (RDF) of this data source.

The information related to each parameter of the RESTful service (semantic annotations) is stored only once in the system repository. By doing this, we avoid to duplicate information related to the same parameters, hence storing annotations independently of services and increasing the efficiency of our system.

5.3.1.1 Ontological Resources

In our approach we use three widespread ontologies, such as DBpedia (a cross-domain ontology), GeoNames and the WGS84 vocabulary (domain ontologies).

DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. The DBpedia ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia. The ontology currently covers over 259 classes which form a subsumption hierarchy and are described by 1,200 different properties. Moreover, this ontology currently contains about 1,478,000 instances (413,000 instances belong to Place class).

GeoNames is a geographical database that contains over 8 million geographical names. The structure behind the data is the GeoNames ontology, which closely resembles the flat-file structure. A typical individual in the database is an instance of type Feature and has a Feature Class (administrative divisions, populated places, etc.), a Feature Code (subcategories of Feature Class) along with latitude, longitude, etc. associated with it.

The WGS84 Vocabulary is a basic RDF vocabulary that provides a namespace for representing latitude, longitude and other information about spatially-located things, using WGS84 as a reference datum.

In this context, DBpedia represents a general domain and GeoNames and WGS84 represent a specific domain. These ontologies are a subset of O as we described in chapter 4.

Definition 6. (General Ontology - DBpedia) is defined as three tuple:

\[ OntDBpedia = \{ C_{DBpedia}, P_{DBpedia}, I_{DBpedia} \} \]
5.3 Semantic Annotation Process

- \(C_{DBpedia}\) represents all ontology concepts in the general domains.
- \(P_{DBpedia}\) includes all properties of OntDBpedia.
- \(I_{DBpedia}\) consists of all instances that belong to ontology.

For example,

\[
\begin{align*}
\text{classes} & \rightarrow \{\text{Country, City, Capital, \ldots}\} \\
\text{properties} & \rightarrow \{\text{city, lat, lon, \ldots}\} \\
\text{instances} & \rightarrow \{\text{Madrid, 40.4153, -3.7073, \ldots}\}
\end{align*}
\]

**Definition 7.** (General Ontology - GeoNames) is defined as a three tuple:

\[\text{OntGeoNames} = \{C_{GeoNames}, P_{GeoNames}, I_{GeoNames}\}\]

- \(C_{GeoNames}\) represents all ontology concepts in the general domains.
- \(P_{GeoNames}\) includes all properties of OntGeoNames.
- \(I_{GeoNames}\) consists of all instances that belong to the ontology.

An ontology has a set of classes denoted \(\mathbf{C}\) and a set of properties denoted \(\mathbf{P}\). For example,

\[
\begin{align*}
\text{classes} & \rightarrow \{\text{Adm, alternativeName, \ldots}\} \\
\text{instances} & \rightarrow \{\text{Madrid, Madrid, \ldots}\} \\
\text{properties} & \rightarrow \{\text{lat, lon, \ldots}\} \\
\text{instances} & \rightarrow \{40.4153, -3.7073, \ldots\}
\end{align*}
\]

**Definition 8.** (Intances) Set of instances of classes or properties.

\[IC = \{C_{DBpedia}, I_{CDBpedia}\} \cup \{C_{GeoNames}, I_{CGeoNames}\}\]

where \(I_{CDBpedia}\) is the set of instances of \(C_{DBpedia}\). For example, ....,

Set of instance of properties

\[IP = \{P_{DBpedia}, I_{PDBpedia}\} \cup \{P_{GeoNames}, I_{PGeoNames}\}\]

where \(I_{PDBpedia}\) is the set of instances of \(P_{DBpedia}\) For example , , , ,


5. RESTFUL SEMANTICANNOTATION PROCESS

Algorithm 4 Algorithm to retrieve possible links between parameters and ontology resources

Require: term

\[\text{parameter} \leftarrow \text{localSearch}(\text{term});\]

if parameter then

return parameter

else

\[\text{parameter.name} \leftarrow \text{term};\]

loadOntologies(OntDBpedia, OntGeonames);

\[\text{CDBpedia} \leftarrow \text{retrieveCDBpedia}(\text{term})\]

for all \(\text{class} \in \text{CDBpedia}\) do

\[\text{ontologyResource.link} \leftarrow \text{class};\]

\[\text{ontologyResource.type} \leftarrow 'class';\]

\[\text{parameter.addOntologyResource(ontologyResource)}\]

\[\text{IDBpedia} \leftarrow \text{retrieveIDBpedia}(\text{class})\]

for all \(\text{instance} \in \text{IDBpedia}\) do

\[\text{ontologyInstance.value} \leftarrow \text{instance}\]

\[\text{ontologyResource.addInstance(ontologyInstance)}\]

end for

end for

\[\text{PDBpedia} \leftarrow \text{retrievePDBpedia}(\text{term})\]

for all \(\text{property} \in \text{PDBpedia}\) do

\[\text{ontologyResource.link} \leftarrow \text{property}\]

\[\text{ontologyResource.type} \leftarrow 'property';\]

\[\text{parameter.addOntologyResource(ontologyResource)}\]

\[\text{IDBpedia} \leftarrow \text{retrieveIDBpedia}(\text{property})\]

for all \(\text{instance} \in \text{IDBpedia}\) do

\[\text{ontologyInstance.value} \leftarrow \text{instance}\]

\[\text{ontologyResource.addInstance(ontologyInstance)}\]

end for

end for

end if

return parameter
5.3 Semantic Annotation Process

5.3.1.2 Using ontological sources in the annotation process

At this stage, the list of syntactic parameters obtained previously is used to query
the DBpedia \( S_{DBpedia} \) and GeoNames \( S_{GeoNames} \) SPARQL Endpoints and retrieve
associated results for each parameter, as showed in algorithm \[1\]. Next, we provide the
main details of this algorithm.

The **first** step is to search in a local cache. If the parameter is not found, the
process of retrieving possible links between parameters and ontology resources starts.
To do this, the DBpedia and GeoNames ontologies are loaded in memory using the
Jena API.

In the **second** step, the algorithm retrieves all the classes from the DBpedia ontol-
ogy \( \text{retrieveC}_{\text{DBpedia}} \) whose names have a match with each parameter of the RESTful
service. In this matching process we test two different techniques:

- We use an exact match to compare parameters of the RESTful service with the
  labels of the ontologies’ classes and properties.

- We use a combination of various similarity metrics (Jaro, JaroWinkler and Lev-
  enshtein metrics) to compare parameters with labels of the elements of these
  ontologies. This proposal allows matching between strings such as `countryName`,
  `country_name`, or `country`, for example.

If the system obtains correspondences from the matching process, it uses these DB-
pedia concepts individually to retrieve samples (concept instances) from the DBpedia
SPARQL Endpoint \( \text{retrieveI}_{\text{DBpedia}} \). In listing \[5.7\] we show the SPARQL query used
to retrieve instances of a class (e.g. Country class) and the response shown in the table
\[5.1\]

Listing 5.7: SPARQL query to retrieve instances of class

1  select distinct ?val where {?val a <http://dbpedia.org/ontology/Country>}

In the **third** step, the algorithm retrieves all the properties from the DBpedia
ontology \( \text{retrieveP}_{\text{DBpedia}} \) whose names have a match with each parameter of the
RESTful service using the techniques described in the second step. If the system
obtains correspondences from the matching process, it uses these DBpedia concepts
individually to retrieve samples (concept instances) from the DBpedia SPARQL End- 
point (retrieveDBpedia). In the list 5.8, we shown the SPARQL query used to retrieve 
instances of a property (e.g. city property) and the response is shown in the table 5.2.

Listing 5.8: SPARQL query to retrieve instances of property

```
1 select distinct ?val where { ?val <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat.  
2 ?s <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?lng } limit ?offset ?
```

Finally, using data obtained in the previous step, the algorithm stores the semantic 
annotations using the model described in section 5.3.1 into the relational database.

Likewise, when a parameter matches an ontology class/property related to some 
geospatial information; such as latitude, longitude or bounding box our system retrieves 
samples from the GeoNames SPARQL Endpoint. These geospatial parameters are 
retrieved in a combined manner (for instance, a latitude goes together with a longitude), 
that is, when the annotation process of a RESTful service requires information about 
latitude and longitude, our system invokes the GeoNames SPARQL Endpoint.

Listing 5.9: Geospatial query

```
1 select ?lat ?lng where { ?s <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat.  
2 ?s <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?lng } limit ?offset ?
```

With query 5.9, our system requests a list of latitudes and longitudes. The values 
of this list are recovered randomly (using limit and offset SPARQL parameters), in
order to avoid recovering always the same values for different queries. The resulting information is suggested automatically to the system and registered as a possible value for the corresponding parameter. When a parameter matches more than one concept in the DBpedia ontology, our system only considers those concepts that have information (instances), and it automatically discards those ontology concepts without instances.

With this process, we have a list of possible links to ontologies. For example to the Service 1 to the country parameter has the following possible links (see appendix A).

### 5.3.1.3 Use of External Resources for enriching the semantic annotations

As a step to automatically generate descriptions of the RESTful services, we describe how external resources can be used to link semantics to RESTful services. Our system looks for matches with DBpedia (and GeoNames) classes and properties. Hence it is possible to have parameters with no correspondences identified, since there are many lexical and syntactic variations that parameter names may have, and because in some cases the information that is being requested may not be available in any of the external sources (suggestion and synonym services) that are consulted.

In order to annotate semantically the parameters that did not match any DBpedia resource, we use additional external services to enrich the results. Below we describe the main characteristics of the external services that we consider.

**Spelling suggestions.** Web search engines (e.g. Google, Yahoo, and Microsoft) usually try to detect and solve users’ typing mistakes. Spelling Suggestion services, also called Did You Mean, are algorithms which aim at solving these spelling mistakes. For example, when a user writes countryName these algorithms suggest country and name separately.
5. RESTFUL SEMANTIC ANNOTATION PROCESS

In our system we use the Yahoo Boss service\(^1\) to retrieve suggestions about the parameters that we have obtained in the previous steps and for which we have not obtained any candidate in our semantic resources. Thus, for each parameter that the system did not find a correspondence with classes or properties in DBpedia (nor GeoNames), this service is invoked to obtain a list of suggestions to query DBpedia (and GeoNames) again. The output is registered and stored into the repository. Following the previous example, the parameter *countryName* is not found in the DBpedia ontology. Nevertheless, the added service allows separating this parameter in *country* and *name*, and then it calls to the DBpedia SPARQL Endpoint with these new strings for obtaining results.

**Synonyms services.** This external service\(^2\) is incorporated into the system to retrieve possible synonyms for a certain parameter. This service tries to improve the semantic annotation process when our system does not offer results for the previous steps, that is, when we still have parameters in a RESTful service without any potential annotations. As an example, we may have a parameter called *address*. The invocation process uses the synonyms service to retrieve a set of synonyms of address such as extension, reference, mention, citation, denotation, destination, source, cite, acknowledgment, and so on. These outputs are registered and stored into the repository, and then, the service calls to the DBpedia (and GeoNames) SPARQL Endpoints for results again.

Both spelling suggestion and synonym services use the matching process described above to find possible matches between the output of these services and the components of the used ontologies. The results of these matches are stored in a cache in a local relational database (for scaling reason, one cannot keep all data in main memory). This cache was developed to overcome query limitations of both suggestion and synonym services (approximately 1,000 invocations per day). Figure 5.6 shows the simple model built for this cache.

Next, we define more formally the external resources used in our approach.

\(^{1}\)http://developer.yahoo.com/search/boss/boss_guide/Spelling_Suggest.html
\(^{2}\)http://www.synonyms.net/
5.3 Semantic Annotation Process

5.3.1.4 Using external resources in the annotation process

The algorithm used to invoke the external resources to increase the possibility to describe a parameter semantically is shown in listing 5. The process starts with a term, in this case a parameter name. The algorithm searches in a local repository. If the term is not found, the algorithm invokes the service of suggestions with the term as a parameter. The results of the invocation are a list of terms suggested by the service. With these terms the process calls the step 4 to add possible links between suggestions and ontological resources. If the result of the invocation of suggestions is empty, the algorithm deletes special characters present in the term, for example, −. The algorithm invokes the service of suggestions with the new term without special characters. With suggestions the algorithm calls the process to add links of possible ontological resources. To increase the possibility of the semantic annotation of parameters, the algorithm uses a service of synonyms to extract terms with same mean that the original parameters as well suggestions terms. With these synonyms term, the algorithm invokes the process of linking the parameters with ontological resources.

5.3.2 Definition of Semantic RESTful Annotation

The semantic annotation of a RESTful service is defined as a tuple:

\[ SAR = (RS, Ont, E) \]

where RS is a RESTful service, Ont is a set of ontologies and E are external resources.
Algorithm 5 algorithmSuggSyn

Require: $t$, parameter

term ← localSearch($t$);
if term then
  return term
else
  listSuggestions ← sug($t$)
  if listSuggestions == null then
    $t$ ← deleteSpecialCharacters($t$)
    listSuggestions ← sug($t$)
  end if
  for all $s ∈ listSuggestions$ do
    suggestion.name ← $s$
    term.addSuggestion(suggestion)
    parameterSuggestion.name ← $s$
    parameter.addSuggestion(parameterSuggestion)
    processAlgorithmOntologyResource(parameterSuggestion);
    listSynonyms ← syn($s$)
    termSugg.name ← $s$
    listSynonyms ← syn($s$)
    for all sy ∈ listSynonyms do
      termSugg.addSynonyms(sy)
      parameterSynonym.name ← sy
      processAlgorithmOntologyResource(parameterSynonym);
    end for
  end for
  listSynonyms ← syn($t$)
  for all sy ∈ listSynonyms do
    term.addSynonyms(sy)
    parameterSynonym.name ← sy
    processAlgorithmOntologyResource(parameterSynonym);
    parameter.addSynonyms(parameterSynonym)
  end for
  return term
end if
5.3 Semantic Annotation Process

5.3.2.1 Possible semantic definition of RESTful services

The use of the algorithms described in the last sections allows generating of a possible semantic definition of a RESTful service with respect to the ontological resources. The system takes each parameter and tries to assign a link to ontological resources. For example, the `countryName` parameter that is an output parameter of the Service 1 may have the following possible links.

Listing 5.10: Possible links of countryName parameter

```plaintext
countryName:
  http://dbpedia.org/ontology/country ---> 1649 instances
  http://dbpedia.org/ontology/Country ---> 1839 instances
  Suggest: country:
  http://dbpedia.org/ontology/capitalCountry ---> 0 instances
  http://dbpedia.org/ontology/usingCountry ---> 145 instances
  http://dbpedia.org/ontology/twinCountry ---> 38 instances
  http://dbpedia.org/ontology/countryWithFirstSpaceflight ---> 0 instances
  http://dbpedia.org/ontology/fastestDriverCountry ---> 0 instances
  http://dbpedia.org/ontology/poleDriverCountry ---> 0 instances
  http://dbpedia.org/ontology/firstDriverCountry ---> 0 instances
  http://dbpedia.org/ontology/secondDriverCountry ---> 0 instances
  http://dbpedia.org/ontology/countryWithFirstSatelliteLaunched ---> 0 instances
  http://dbpedia.org/ontology/country ---> 1839 instances
  http://dbpedia.org/ontology/managementCountry ---> 0 instances
  http://dbpedia.org/ontology/governmentCountry ---> 0 instances
  http://dbpedia.org/ontology/sourceConfluenceCountry ---> 0 instances
  http://dbpedia.org/ontology/sourceCountry ---> 137 instances
  http://dbpedia.org/ontology/thirdDriverCountry ---> 0 instances
  http://dbpedia.org/ontology/countryWithFirstAstronaut ---> 1 instances
  http://dbpedia.org/ontology/countryWithFirstSatellite ---> 0 instances
  http://dbpedia.org/ontology/country ---> 1649 instances
  http://dbpedia.org/ontology/countryOrigin ---> 67 instances
  http://dbpedia.org/ontology/locationCountry ---> 1771 instances
  http://dbpedia.org/ontology/mouthCountry ---> 144 instances
  Synonyms: dry land:
  Synonyms: area:
  http://dbpedia.org/ontology/areaWater ---> 1851 instances
  http://dbpedia.org/ontology/percentageOfAreaWaterRound ---> 0 instances
  http://dbpedia.org/ontology/locatedInArea ---> 1727 instances
  http://dbpedia.org/ontology/PopulatedPlace/areaUrban ---> 193 instances
  http://dbpedia.org/ontology/PopulatedPlace/areaMetro ---> 0 instances
  http://dbpedia.org/ontology/councilArea ---> 1154 instances
  http://dbpedia.org/ontology/areaTotal ---> 1999 instances
  http://dbpedia.org/ontology/Planet/surfaceArea ---> 38 instances
  http://dbpedia.org/ontology/areaOfCatchment ---> 658 instances
  http://dbpedia.org/ontology/principalArea ---> 730 instances
  http://dbpedia.org/ontology/lieutenancyArea ---> 614 instances
  http://dbpedia.org/ontology/GeopoliticalOrganisation/areaMetro ---> 0 instances
```

109
5. RESTFUL SEMANTIC ANNOTATION PROCESS

http://dbpedia.org/ontology/areaCode
http://dbpedia.org/ontology/percentageOfAreaWater
http://dbpedia.org/ontology/areaUrban
http://dbpedia.org/ontology/areaLand
http://dbpedia.org/ontology/Building/floorArea
http://dbpedia.org/ontology/floorArea
http://dbpedia.org/ontology/areaMetro
http://dbpedia.org/ontology/area
http://dbpedia.org/ontology/ProtectedArea
http://dbpedia.org/ontology/broadcastArea
http://dbpedia.org/ontology/Lake/areaOfCatchment
http://dbpedia.org/ontology/areaOfSearch
http://dbpedia.org/ontology/SkiArea
http://dbpedia.org/ontology/surfaceArea
http://dbpedia.org/ontology/filmFareAward

Synonyms: rural area:
http://dbpedia.org/ontology/area

Synonyms: terra firma:
http://dbpedia.org/ontology/id
http://dbpedia.org/ontology/ground

Synonyms: solid ground:
http://dbpedia.org/ontology/id
http://dbpedia.org/ontology/ground

Synonyms: state of matter:
http://dbpedia.org/ontology/state

Synonyms: res publica:
http://dbpedia.org/ontology/state

Synonyms: landed estate:
http://dbpedia.org/ontology/state

Synonyms: body politic:

Synonyms: democracy:

Synonyms: soil:

Synonyms: commonwealth:

Synonyms: estate:
http://dbpedia.org/ontology/sourceState
http://dbpedia.org/ontology/state
http://dbpedia.org/ontology/sourceConfluenceState

Synonyms: farming:
http://dbpedia.org/ontology/arm

Synonyms: domain:
http://dbpedia.org/ontology/domain

Synonyms: kingdom:
http://dbpedia.org/ontology/kingdom

Synonyms: earth:
http://dbpedia.org/ontology/lowerEarthOrbitPayload
http://dbpedia.org/ontology/Rocket/lowerEarthOrbitPayload

Synonyms: nation:
http://dbpedia.org/ontology/maximumInclination
http://dbpedia.org/ontology/nationality
http://dbpedia.org/ontology/unitedStatesNationalBridgeId

http://dbpedia.org/ontology/nationalFilmAward
http://dbpedia.org/ontology/nationalTeam
http://dbpedia.org/ontology/denomination
5.3 Semantic Annotation Process

<table>
<thead>
<tr>
<th>URI</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://dbpedia.org/ontology/nationalOrigin">http://dbpedia.org/ontology/nationalOrigin</a></td>
<td>1027</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/NationalCollegiateAthleticAssociation">http://dbpedia.org/ontology/NationalCollegiateAthleticAssociation</a></td>
<td>374</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/goalsInNationalTeam">http://dbpedia.org/ontology/goalsInNationalTeam</a></td>
<td>81</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/appearancesInNationalTeam">http://dbpedia.org/ontology/appearancesInNationalTeam</a></td>
<td>0</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/inclination">http://dbpedia.org/ontology/inclination</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/destination">http://dbpedia.org/ontology/destination</a></td>
<td>81</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/appearancesInNationalTeam">http://dbpedia.org/ontology/appearancesInNationalTeam</a></td>
<td>0</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/inclination">http://dbpedia.org/ontology/inclination</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/destination">http://dbpedia.org/ontology/destination</a></td>
<td>81</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/appearancesInNationalTeam">http://dbpedia.org/ontology/appearancesInNationalTeam</a></td>
<td>0</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/inclination">http://dbpedia.org/ontology/inclination</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/destination">http://dbpedia.org/ontology/destination</a></td>
<td>81</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/appearancesInNationalTeam">http://dbpedia.org/ontology/appearancesInNationalTeam</a></td>
<td>0</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/inclination">http://dbpedia.org/ontology/inclination</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/destination">http://dbpedia.org/ontology/destination</a></td>
<td>81</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/appearancesInNationalTeam">http://dbpedia.org/ontology/appearancesInNationalTeam</a></td>
<td>0</td>
</tr>
<tr>
<td><a href="http://dbpedia.org/ontology/inclination">http://dbpedia.org/ontology/inclination</a></td>
<td>1</td>
</tr>
</tbody>
</table>

Synonyms: acres:

Synonyms: ground:

http://dbpedia.org/ontology/ground --> 1835 instances

http://dbpedia.org/ontology/background --> 623 instances

Synonyms: demesne:

Synonyms: land:

http://dbpedia.org/ontology/dorlandsSuffix --> 2000 instances

http://dbpedia.org/ontology/numberOfIslands --> 32 instances

http://dbpedia.org/ontology/landingDate --> 321 instances

http://dbpedia.org/ontology/island --> 121 instances

http://dbpedia.org/ontology/landingSite --> 0 instances

http://dbpedia.org/ontology/landingvehicle --> 19 instances

http://dbpedia.org/ontology/lunarLandingSite --> 0 instances

http://dbpedia.org/ontology/dorlandsPrefix --> 225 instances

http://dbpedia.org/ontology/Island --> 1541 instances

http://dbpedia.org/ontology/areaLand --> 1999 instances

http://dbpedia.org/ontology/majorIsland --> 71 instances

Synonyms: state:

http://dbpedia.org/ontology/lowestState --> 1 instances

http://dbpedia.org/ontology/stateDelegate --> 0 instances

http://dbpedia.org/ontology/mouthState --> 0 instances

http://dbpedia.org/ontology/state --> 1524 instances

http://dbpedia.org/ontology/unitedStatesNationalBridgeId --> 202 instances

http://dbpedia.org/ontology/sourceState --> 0 instances

http://dbpedia.org/ontology/stateOfOrigin --> 1829 instances

http://dbpedia.org/ontology/federalState --> 63 instances

http://dbpedia.org/ontology/sourceConfluenceState --> 15 instances

http://dbpedia.org/ontology/highestState --> 1 instances

http://dbpedia.org/ontology/SupremeCourtOfTheUnitedStatesCase --> 1751 instances

http://dbpedia.org/ontology/states --> 1937 instances

Synonyms: realm:

http://dbpedia.org/ontology/province --> 1879 instances

Synonyms: republic:

Suggest: Name:

http://dbpedia.org/ontology/namedAfter --> 5 instances
5. RESTFUL SEMANTIC ANNOTATION PROCESS

The combination of each parameter (with one or more links) with other parameters can be used to generate a large list of possible definitions. The challenge is how to obtain the optimal real combination. As a first step, we discard all links that have not instances or where the number of instances is \(< 10\). As we can see, each of the possible links to the ontology has a list of instances that may be used to obtain the best definition.

5.3.3 Checking the semantic annotation of RESTful services

In order to check the collected sample individuals and the initial semantic annotations obtained as a result of the previous process, our system invokes the RESTful services that were already registered in the repository (as we describe in the section related to the syntactic description) and validates the input and output parameters for checking which is the best option to describe each parameter.

For the validation of the input parameters, our system selects, for each parameter, a random subset of the example instances (of classes and/or properties) coming from the DBpedia (and GeoNames) ontology that we have obtained and registered before. Next, it makes several invocations to the RESTful service iterating over these registered values. The system does not check this with all the possible combinations of collected instances for all parameters for two reasons: first, because of the combinatorial explosion that may be produced in such a case, and second because many RESTful services have invocation limitations.

When a service has several input parameters, the system obtains randomly some instances for each parameter, to be used for the validation process. Each parameter
Algorithm 6 Verification algorithm

Require: URL RESTful

Ensure: Verification (URL,inputParameters)

\[
rs \leftarrow 0; \\
rs.URL \leftarrow URL \\
rs.I \leftarrow \text{getInputParameters(URL)}; \text{\{Extract list of input parameters\}} \\
rs.rsName \leftarrow \text{getName(URL)}; \text{\{The name is the string without parameters\}} \\
\text{instancesDBpediaParameters} \leftarrow \text{getInstancesDBpedia(rs.I)} \text{\{Retrieve a list of instances from DBpedia SPARQL endpoint\}} \\
\text{instancesGeonamesParameters} \leftarrow \text{getInstancesGeonames(rs.I)} \text{\{Retrieve a list of instances from GeoNames SPARQL endpoint, only if the parameter are geospatial\}}
\]

\[
\text{for all instance} \in \text{instancesDBpediaParameters do} \\
\text{result} \leftarrow \text{callRESTful(instance)}; \\
\text{if result then} \\
\text{\quad invocation successfully} \\
\text{end if} \\
\text{end for}\text{\{for geospatial parameters\}} \\
\text{for all instance} \in \text{instancesGeonamesParameters do} \\
\text{result} \leftarrow \text{callRESTful(instance)}; \\
\text{if result then} \\
\text{\quad invocation successfully} \\
\text{end if} \\
\text{end for}
\]
5. RESTFUL SEMANTIC ANNOTATION PROCESS

generates a collection (list) of instances from our repository. Then, the system joins instances to obtain a table of all combinations of each parameter. Likewise, the geospatial parameters, specifically latitude and longitude parameters, are combined to obtain some values (instances) that can be used for this invocation.

If the service returns results from the invocation, then the service is considered as executable, and the corresponding annotations are marked as valid. If a service cannot be invoked successfully, the service is classified as non-executable and is automatically discarded from the list of services that can be automatically annotated.

For the validation of the output parameters, our system only takes into account executions with the correct inputs from the input sets that have been considered before. Next, the system compares the outputs obtained after execution with the information already stored in the repository due to the initial retrieval processes done before. If the output can be matched, our system considers that the output annotation is valid.

Finally, the correspondences established between different parameters of the RESTful service and the DBpedia (and GeoNames) ontology are registered and stored in the repository, so that they can be used later. In such a way, the RESTful service is annotated semantically and it allows generating semantic descriptions or annotations of any of the types that were identified in the related work section (WADL, hREST, SA-REST, etc.).

In table 5.11 we summarize the interface of one of the candidate concrete services in terms of operation name, input parameters and return value.

Listing 5.11: Semantic description of our sample RESTful services (service 1)

```
```

Table 5.11 provides an abbreviated form of this description for our exemplar Service 1.
5.4 Conclusions

5.3.3.1 SA-REST

Furthermore, the HTML code shown in listing 5.12 has been generated using the HTML descriptions shown in Figure 5.3 and table 5.6. These tables provide a semantic annotation piece of the aforementioned service (Service 1) according to the hREST and SA-REST formats, respectively.

Listing 5.12: SA-REST description of our sample RESTful services

```html
<dive class="section domain-re1" lang="en" title="sarest:Service">
  <span class="domain-re1" title="sarest:Operation">
    <a href="#http://ws.geonames.org#countryInfo">http://ws.geonames.org/countryInfo</a>
  </span>
  <span class="domain-re1" title="sarest:inputMessage">
    <strong>country</strong>
  </span>
  <td>
    <a href="/xsd" class="sem-re1" title="http://dbpedia.org/ontology/country">
      This is the input service
    </a>
  </td>
  <span class="domain-re1" title="sarest:outputMessage">
    <strong>countryName</strong>
  </span>
  <td>
    <a href="/xsd" class="sem-re1" title="http://dbpedia.org/ontology/country">
      This is the output service
    </a>
  </td>
</div>
```

5.4 Conclusions

In this chapter we have shown an implementation of the general process described in Chapter 4. We describe the developed concepts to achieve a generation of enriched RESTful service description models based on existing syntactic service description. Starting from a model for representing RESTful services, which depicts the information that the RESTful service provides, the Chapter separately considers the different
5. RESTFUL SEMANTIC ANNOTATION PROCESS

aspects of the envisioned solution. For this, it provides a precise definition of the defined model for representing RESTful services. This model is one of the main contributions of this work. Subsequent sections consider the definition of steps which are used to create service descriptions based in this model and the storing of these into a relational data base. The Chapter explains a prototypical implementation of the introduced concepts.
In this chapter, we present the outline of a novel semantic annotation process of OGC Web Services (OWS), concretely Web Feature Services (WFS). A WFS is a Web service that allows querying and retrieving vector data and alphanumeric information linked to the WFS. Currently, a number of public and private organizations are generating WFS to allow access to their geographical data (e.g., Geospatial Data Service Centre, USGS Framework Web Feature Services, Infraestructura de Datos Espaciales de España, and so on). These organizations, who may be responsible of Spatial Data Infrastructures (SDIs), usually describe the WFS services that they offer in their websites using HTML pages. These pages contain information about how the services can be accessed, provide descriptions of the entities involved, restrictions on the service usage and allowed values, and more.

The lack of comprehensive repositories of WFS services makes it difficult to deal with them in an automatic way, forcing the manual search of services through common search or browse through websites of different SDIs.

In this Chapter we introduce an automated approach for the semantic annotation of this kind of geospatial services. Section 6.2 describes the syntactic description process.
representing the algorithm and used tools. Section 5.3 uses the result of syntactic description to define the semantic annotation process of WFS. Finally, we explain how we mix the WFS with Semantic Web technologies to enrich the geospatial Web with semantic descriptions.

6.1 An automated approach for the semantic annotation of WFS

The semantic annotation of WFS services can provide benefits in tasks related to the discovery, composition and mediation of geospatial information. However, few works are focused on the study of syntactic registration and semantic annotations of this type of services as we described in Chapter 2 in section 2.4.

In this section, we describe our proposed process for automating the semantic annotation of WFS services based on the use of ontological resources and two external resources (suggestion and synonyms services) for extracting information that can be useful for the annotation process.

The following service, taken from the World Health Organization’s website, is a representative WFS service, which we will use throughout this chapter to illustrate our approach. This service contains health information controlled by the United Nations system.

- **Service 1.** http://apps.who.int/tools/geoserver/wfs?REQUEST=GetCapabilities &SERVICE=WFS

In figure 6.1, we show the main components of the process for automating the syntactic registration and semantic annotation of WFS services. As in the previous chapter, we must note that this process is the implementation of the generalised process presented in chapter 4.

Our proposal consists of three main components: a) the invocation and registration components, which are in charge of the syntactic description of three methods (GetCapabilities, DescribeFeatureType, and GetFeature); b) the repository, which stores service descriptions; and c) the semantic annotation component, which makes use of diverse external resources to characterise a WFS service. Next, we briefly describe these components, illustrating the descriptions with some sample services.
6.2 Syntactic description of WFS services: Registering invocation details in a repository

The majority of OGC Web services can be accessed and invoked via HTTP GET and HTTP POST operations. However, these services do not support SOAP. Hence, WFS services are not equivalent to the W3C SOAP-based Web services (TA06).

With respect to WFS, they can be accessed in different ways.

- Sending and receiving WFS geospatial data through the HTTP protocol (e.g. through a using Web browser).

- Using specialised clients. There are many third-party clients able to consume WFS services. For instance: Geotools\(^1\) Feature Data Objects (FDO\(^2\)) etc.

All standard WFS services support three mandatory operations: GetCapabilities, DescribeFeatureType, and GetFeature. Next, we provide some details of these operations.

6.2.1 GetCapabilities

The GetCapabilities operation is used to obtain a description of the functionality and the data offered by an OGC WFS service. So, this operation returns a list of the

---

\(^1\)http://www.geotools.org/
\(^2\)http://fdo.osgeo.org/
6. OGC WEB FEATURE SERVICE ANNOTATION PROCESS

géographic feature types that can be serviced, normally encoded in GML. When the additional descriptions are provided as a result of this invocation, they mostly consist of plain-text. Besides the feature type list, we can also obtain metadata about the service. Each WFS provides a document describing its capabilities, which is an XML document composed of:

- **Service**. This section provides information about the service itself. Therefore, it contains service metadata (name, title, abstract, keyword, etc.).

- **Capabilities**. This section describes the operations that are supported by a specific service. According to the WFS specification, this service must support at least GetCapabilities, the one that we are using here, DescribeFeatureType and GetFeature operations.

- **FeatureTypeList**. This section contains a list of feature types that a WFS can service. Besides, it defines transactions and query elements that are supported by each feature type.

The invocation of GetCapabilities starts by taking as input the URL of an available WFS service, which may be typed by a user or found by a crawler. Once the URL is added, our system invokes the WFS service through its GetCapabilities operation and analyzes the response to obtain information that allows determining the main characteristics of the service. The URL pattern to invoke a WFS service is shown in listing 6.1. In response to this request the WFS generates a Capabilities document that contains the elements previously discussed.

**Listing 6.1: URL to invoke a WFS service**

```plaintext
http://www.example.com/wfs?service=ws&version=1.1.0&request=GetCapabilities
```

In the case of the aforementioned Service 1, the URL following the model of invocation of listing 6.1 is http://apps.who.int/tools/geoserver/wfs?REQUEST=GetCapabilities &SERVICE=WFS

In listing 6.2 we show a fragment of the response (XML) to this call, from which we obtain information about the features serviced by this service. The full list of capabilities can be found in annex B. After the namespaces, there are different service
6.2 Syntactic description of WFS services: Registering invocation details in a repository

metadata (title, abstract, keywords, service type, version, and access constants). Likewise, the XML response of used WFS shows two feature types related to country data and monitoring geometry measures, with their descriptions and their bounding boxes.

**Listing 6.2:** Service 1. Example of the resulting XML from GetCapabilities

```xml
<wfs:WFS_Capabilities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wfs" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:cte="http://www.opengeospatial.net/cite"
xmlns:it.geosolutions="http://www.geo-solutions.it"
xmlns:tiger="http://www.census.gov"
xmlns:geoserver="http://geoserver.sf.net"
xmlns:topp="http://www.openplans.org/topp"
xmlns:sp="http://www.openplans.org/spearfish"
xmlns:WHO="http://www.who.int/"
xmlns:nurc="http://www.nurc.nato.int" version="1.1.0"
xsi:schemaLocation="http://www.opengeospatial.net/wfs http://apps.who.int/geo/" tools/geoserver
>schemas/wfs/1.1.0/wfs.xsd" updateSequence="528">
  <ows:ServiceIdentification>
    <ows:Title>WHO WFS</ows:Title>
    <ows:Abstract>
      Welcome to the World Health Organization Web Feature server.
    </ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>WFS</ows:Keyword>
      <ows:Keyword>WMS</ows:Keyword>
      <ows:Keyword>GEOSERVER</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WFS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>Restricted</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <FeatureType xmlns:WHO="http://www.who.int/"
    xmlns:es=2008_countrydata><Name>WHO_es2008_countrydata</Name>
    <Title>EpiFact Sheets 2008 Data</Title>
    <Abstract>Generated from WHOGeodata</Abstract>
    <ows:Keywords>
      <ows:Keyword>efe_data_for_web service</ows:Keyword>
      <ows:Keyword>WHOGeodata</ows:Keyword>
    </ows:Keywords>
    <ows:WGS84BoundingBox>
      <ows:LowerCorner>-527.400749390449 -545.3687536171334</ows:LowerCorner>
```
Both service metadata and feature type information are provided in a syntactic manner, that is, these descriptions are aimed primarily at human consumption. Hence, humans can only identify whether the service is useful or not for a specific application by looking at the results of this invocation. Furthermore, the service does not use a common and shared vocabulary for these descriptions (e.g., countries, geometry, etc.). This makes it difficult to perform tasks associated with service discovery, composition and mediation as aforementioned.

When a full list of feature types is required, the appropriate GET request would be as follows:

**Listing 6.3:** URL to retrieve a list of feature types

```plaintext
http://www.example.com/wfs?
  service=wfs&
  version=1.1.0&
  request=DescribeFeatureType
```

For our example the URL is http://apps.who.int/tools/geoserver/wfs?REQUEST=DescribeFeatureType&SERVICE=WFS. In this case, we obtain a list of 44 features, which are presented in listing 6.4.
### Listing 6.4: Features list of Service 1

<table>
<thead>
<tr>
<th>Feature Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES</td>
</tr>
<tr>
<td>2. WHO:FLUID_TEST_GIM_MONITORING_GEOMETRY_MEASURES</td>
</tr>
<tr>
<td>3. WHO:FLUID_TRAINING_GIM_MONITORING_GEOMETRY_MEASURES</td>
</tr>
<tr>
<td>4. WHO:GIM_MONITORING_GEOMETRY_MEASURES</td>
</tr>
<tr>
<td>5. WHO:YangonRoads polyline</td>
</tr>
<tr>
<td>6. WHO:admin1_world</td>
</tr>
<tr>
<td>7. WHO:admin2_world</td>
</tr>
<tr>
<td>8. WHO:admin3_world</td>
</tr>
<tr>
<td>9. WHO:admin_world</td>
</tr>
<tr>
<td>10. WHO:admin_world_centroid</td>
</tr>
<tr>
<td>11. WHO:africa_airports</td>
</tr>
<tr>
<td>12. WHO:countries_indide_points</td>
</tr>
<tr>
<td>13. WHO:disputed_borders_white</td>
</tr>
<tr>
<td>14. WHO:efs2008_countrydata</td>
</tr>
<tr>
<td>15. WHO:efs2008_sentinel_sites</td>
</tr>
<tr>
<td>16. WHO:ht_geonames</td>
</tr>
<tr>
<td>17. WHO:ith_list_new</td>
</tr>
<tr>
<td>18. WHO:leishecomp</td>
</tr>
<tr>
<td>19. WHO:leishecomp_humans</td>
</tr>
<tr>
<td>20. WHO:leishecomp_resevoirs</td>
</tr>
<tr>
<td>21. WHO:leishecomp_sandflies</td>
</tr>
<tr>
<td>22. WHO:maskline08_detailed</td>
</tr>
<tr>
<td>23. WHO:maskpoly08_detailed</td>
</tr>
<tr>
<td>24. WHO:member_states</td>
</tr>
<tr>
<td>25. WHO:member_states_label</td>
</tr>
<tr>
<td>26. WHO:mm_admin0</td>
</tr>
<tr>
<td>27. WHO:mm_admin1</td>
</tr>
<tr>
<td>28. WHO:mm_admin2</td>
</tr>
<tr>
<td>29. WHO:mm_nga_placenames</td>
</tr>
<tr>
<td>30. WHO:mm_unosat_flood_extent</td>
</tr>
<tr>
<td>31. WHO:mm_who_populatedplaces</td>
</tr>
<tr>
<td>32. WHO:mmr_damaged_bridges</td>
</tr>
<tr>
<td>33. WHO:mmr_polbnda_adm3_mimu_polygon</td>
</tr>
<tr>
<td>34. WHO:mmr_rds1_1m_vmap1_polyline</td>
</tr>
<tr>
<td>35. WHO:mmr_rds1_1m_vmap1_polyline</td>
</tr>
<tr>
<td>36. WHO:mmr_unjlc_roads</td>
</tr>
<tr>
<td>37. WHO:myanmar_health_facilities</td>
</tr>
<tr>
<td>38. WHO:v_unit_h</td>
</tr>
<tr>
<td>39. WHO:western_sahara</td>
</tr>
<tr>
<td>40. WHO:world_countries</td>
</tr>
<tr>
<td>41. WHO:world_countries_inside_points</td>
</tr>
<tr>
<td>42. WHO:world_influenza_humanhealthlab</td>
</tr>
<tr>
<td>43. WHO:world_who_offices</td>
</tr>
<tr>
<td>44. WHO:yellow_fever_outbreaks</td>
</tr>
</tbody>
</table>
6. OGC WEB FEATURE SERVICE ANNOTATION PROCESS

6.2.2 DescribeFeatureType

The purpose of the *DescribeFeatureType* operation is to allow requesting information about the list of feature types offered by the service (note that this is redundant with respect to GetCapabilities) or about a specific individual featureType. All features are composed of a set of properties, which are described in an XML schema, which can be obtained through the *DescribeFeatureType* operation. Listing 6.5 shows the URL pattern used for the invocation of this operation.

Listing 6.5: URL to retrieve a list of DescribeFeatureType

```
http://www.example.com/wfs?
service=wfs&
version=1.1.0&
request=DescribeFeatureType&
typeName=namespace:featuretype
```

So, in this case the URL used to retrieve information about the feature *WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES* is

```
http://apps.who.int/tools/geoserver/wfs?REQUEST=DescribeFeatureType&TYPE-NAME=WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES.
```

The result of this invocation is shown in listing 6.6. This listing provides a list of parameters of this featureType: region_id, country_name, iso_code, etc.

Listing 6.6: DescribeFeatureType. List of parameters

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:WHO="http://www.who.int/"
xmlns:gml="http://www.opengis.net/gml"

elementFormDefault="qualified"
targetNamespace="http://www.who.int/">
  <xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="http://apps.who.int/tools/geoserver/schemas/gml/3.1.1/
base/gml.xsd"/>
  <xsd:complexType name="FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURESType">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element minOccurs="1" maxOccurs="1" name="COUNTRY_ID" nillable="true" type="xsd:int"/>
          <xsd:element minOccurs="1" maxOccurs="1" name="REGION_ID" nillable="true" type="xsd:int"/>
          <xsd:element minOccurs="1" maxOccurs="1" name="COUNTRY_NAME" nillable="true" type="xsd:string"/>
          <xsd:element minOccurs="1" maxOccurs="1" name="ISO_CODE" nillable="true" type="xsd:string"/>
          <xsd:element minOccurs="1" maxOccurs="1" name="REGIONCODE" nillable="true" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```
6.2 Syntactic description of WFS services: Registering invocation details in a repository

```xml
<xsd:element maxOccurs="1" minOccurs="0" name="REGION_NAME" nillable="true" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="1" name="LOCATION_ID" nillable="true" type="xsd:int"/>
<xsd:element maxOccurs="1" minOccurs="0" name="COUNTRY_GEOMETRY" nillable="true" type="gml:GeometryPropertyType"/>
<xsd:element maxOccurs="1" minOccurs="0" name="GEOGRAPHIC_SPREAD" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="INTENSITY" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TREND" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="IMPACT" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NUM_OF_ILL" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="ILI_POPULATION_COVERED" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="NUM_OF_ILI_REPORTING_SITES" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NUM_OF_PATIENT_VISITS" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NUM_OF_SARI_REPORTED" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NO_OF_SARI_DEATH_REP" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="SARI_POPULATION_COVERED" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NUM_OF_ADMISSIONS_AT_SITE" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="NUM_OF_SARI_REPORTING_SITES" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_NUM_OF_DEATHS" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="ILL_SITES_REPORTING_BY_WEEK" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="SARI_DEATH_SURVEILLANCE" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="SARI_SITES_PER_WEEK" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="TOT_RESPIRATORY_DISEASE_DEATHS" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="WKLY_SUSPECTED_NEW_H1N1_CASES" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="WKLY_PROBABLE_NEW_H1N1_CASES" nillable="true" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="0" name="WKLY_LAB_CONFIRMED_NEW_H1N1_CASES" nillable="true" type="xsd:decimal"/>
```

Again, the parameters used to describe this feature type, as we show in listing 6.6, are not adequately described for an easy consumption. The used vocabulary is not common and shared and therefore, every service may use a different term to refer the same type of parameter (e.g., COUNTRY_ID, Country_ID, countryCode, etc.).

6.2.3 GetFeature

The GetFeature operation requests data of a specific feature, including geometry and attribute values. After obtaining the description of feature WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES using the DescribeFeatureType request, properties that define the instances of the feature are retrieved.

The simplest way to run a GetFeature command is without any arguments as listing 6.7 shows in the URL pattern.

Listing 6.7: GetFeature operation

```xml
http://www.example.com/wfs?
service=wfs&
```
6.2 Syntactic description of WFS services: Registering invocation details in a repository

We can also restrict our request with a filter, providing specific values of any properties that are applicable to the feature type. For example, in order to retrieve the characteristics and location of the WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES entity for country with id 9 (Australia), we call the GetFeature request with 9 as the value of COUNTRY_ID parameter. This executes the actual query parameters such as bounding box and any other filters, as appropriate, and the WFS service returns a GML, which contains detail about geometry and attributes. Listings 6.8 and 6.9 show the request and response respectively.

Listing 6.8: GetFeature of a feature type

```xml
http://apps.who.int/tools/geoserver/wfs?REQUEST=GetFeature&
&TYPENAME=WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES&FILTER=<Filter>
<COUNTRY_ID(EqualTo(PropertyName)><Literal>9</Literal></PropertyIsEqualTo></Filter>
```

Listing 6.9: Get feature response

```xml
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ows="http://www.opengis.net/ows"
xmlns:wfs="http://www.opengis.net/wfs" numberofFeatures="1"
timeStamp="2011-07-13T14:07:15.855+02:00"
xsi:schemaLocation="http://www.who.int/
http://apps.who.int/tools/geoserver/wfs?service=WFS&version=1.1.0&
request=DescribeFeatureType&typeName=WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES
http://www.opengis.net/wfs-http://apps.who.int/tools/geoserver/schemas/wfs/1.1.0/
wfs.xsd">
<wfs:featureMembers>
<WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES gml:id="FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES.204">
<WHO:COUNTRY_ID>9</WHO:COUNTRY_ID>
<WHO:REGION_ID>6</WHO:REGION_ID>
<WHO:COUNTRY_NAME>Australia</WHO:COUNTRY_NAME>
<WHO:LOCATION_ID>46</WHO:LOCATION_ID>
<WHO:COUNTRY_GEOMETRY>
<gml:MultiSurface srsName="urn:o-ogc:def:crs:EPSG:4326">
<gml:surfaceMember>
<gml:Polygon>
<gml:exterior>
<gml:LinearRing>
```

127
6.2 Syntactic description of WFS services: Registering invocation details in a repository

6.2.4 Implementation of invocation and syntactic registration

Taking into account the described characteristics of WFS services, the challenge associated with our approach is how to automatically manipulate these XML files related to this kind of service. For that, we use the Geotools\(^1\) API, which allows invoking and processing a WFS service from its URL. Besides, this API also allows navigating through collected results and extracting the characteristics of WFS. As we said before, the service invocations of a specific WFS service return data in GML format. The result of this invocation is a syntactic description of WFS, which can be expressed in the model described in chapter 4 and stored into a relational model. As described for RESTful services in chapter 5, the relational model is implemented as a database that is specifically designed to store syntactic descriptions of WFS services. We selected this storage model in order to increase the efficiency in the recovery of the WFS services.

The algorithm used for the registration of a WFS service takes as input a URL for the new service (found) and uses it to invoke the service. The algorithm describes the overall process (see algorithm\(^7\)). The first step is to retrieve the input parameters and invoke the new service with a set of input parameters. This invocation is performed using the Geotools API. The invocation response is a GML document that contains the definition of a WFS service. The GML is processed using Geotools to retrieve a list of features and taking that list, the algorithm retrieves a set of output parameters of each feature. Furthermore, our algorithm retrieves associated values with each input/output parameters. The obtained parameter value of each feature is transformed into a relational view (table). These data are used by our algorithm for creating a structure according to the model described in section 6.3.1. For speed reasons of the data processing, this model is stored into a relational database. The next step in the algorithm is to generate standard models used to represent WFS services.

\(^{1}\)Geotools is an open source (LGPL) Java code library that provides standards compliant methods for the manipulation of geospatial data and is based on Open Geospatial Consortium (OGC) specifications
**Algorithm 7** Invocation algorithm

**Require:** URL

**Ensure:** Syntactic description \( w(\text{URL}, \text{wName}, I, O) \)

\[
\begin{align*}
&\text{w} \leftarrow 0; \\
&w.\text{URL} \leftarrow \text{URL} \\
&w.\text{capabilities} \leftarrow \text{getCapabilities(\text{URL})}; \text{\{Extract a XML file that describe the WFS service\}} \\
&w.\text{properties} \leftarrow \text{getProperties(w.capabilities)}; \\
&w.\text{features} \leftarrow \text{getFeatures(w.capabilities)}; \\
&\text{for all feature} \in w.\text{features} \text{ do} \\
&\quad w.\text{properties} \leftarrow \text{getPropertyFeatures(feature)}; \\
&\quad w.\text{parameters} \leftarrow \text{getParametersFeature(feature)}; \\
&\quad w.\text{instances} \leftarrow \text{getInstancesFeatures(feature)}; \\
&\text{for all instance} \in w.\text{instances} \text{ do} \\
&\quad w.\text{output} \leftarrow \text{instance.value}; \\
&\text{end for} \\
&\text{end for} \\
&\text{storeObjectModel(rs);} \\
&\text{return rs}
\end{align*}
\]
6.2 Syntactic description of WFS services: Registering invocation details in a repository

As a result of the aforementioned process, we obtain a basic syntactic description of the service. Listing 6.10 shows different parameters of our sample WFS service, where there is a list of parameters that can be annotated. The rest of the description is annotated according to the service model described in section 6.2.

Listing 6.10: Attributes list of Service 1 (WFS)

1 WHO-FLUID, DEV, GIM, MONITORING, GEOMETRY, MEASURES (COUNTRY_ID, REGION_ID, COUNTRY_NAME,
2 ISO_CODE, REGION_CODE, REGION_NAME, LOCATION_ID, COUNTRY_GEOMETRY, GEOGRAPHIC_SPREAD,
3 INTENSITY, TREND, IMPACT, TOT_NUM_OF_ILI, ILL_POPULATION_COVERED, TOT_NUM_OF_PATIENT
4 VISITS, NUM_OF_ILI_REPORTING_SITES, TOT_NUM_OF_SARI_REPORTED, TOT_NO_OF_SARI_DEATH
5 _REP, SARI_POPULATION_COVERED, TOT_NO_OF_ADMISSIONS_AT_SITE, NUM_OF_SARI_REPORTING_
6 SITES, TOT_NUM_OF_DEATHS, ILL_SITES.Reporting_BY_WEEK, SARI_DEATH_SURVEILLANCE, SARI_
7 SITES_PER_WEEK, TOT_RESPIRATORY_DISEASE_DEATHS, WKLY_SUSPECTED_NEW_H1N1_CASES, WKLY_
8 PROBABLE_NEW_H1N1_CASES, WKLY_LAB_CONFIRMED_NEW_H1N1_CASES, WKLY_LAB_CONFIRMED_NEW_
9 H1N1_DEATHS, CUMUL_SUSPECTED_H1N1_CASES, CUMUL_PROBABLE_H1N1_CASES, CUMUL_LAB_
10 CONFIRMED_H1N1_CASES, CUMUL_LAB_CONFIRMED_H1N1_DEATHS, LAST_UPDATED_DATE, UUID.)

The WFS service model plays an essential role in the process of syntactic and semantic description, because we need to register and to store this description into a repository. This model is based on the WSMO-Lite\(^1\) and POSM\(^2\) ontologies. Both of them are used to describe the schema of our WFS services. The main classes of our model to describe WFS services are illustrated in Figure 6.2.

![Figure 6.2: WFS - A model for WFS service description](image)

The WFS class inherits from the Source class, which is common to all classes of services. Regarding the relation between this metadata model and OGC WFS, the Service class may have information about the capabilities of WFS extracted using a getCapabilities operation as we described in section 6.2.1. The Property class may have

\(^1\)http://www.wsmo.org/ns/wsmo-lite/

\(^2\)http://www.wsmo.org/ns/posm/0.1/
metadata about parameters listing in the capabilities document such as keyword, name, etc. Furthermore, information about specific characteristics of a feature such as name, abstract, keywords, WGS84BoundingBox, etc., are also treated. Likewise, the Method class may have also data about of each feature extracted using DescribeFeatureType operation. The Output class may have data about parameters of each feature collected with GetFeature operation as we described in section 6.2.3. The OutputValue class may have data related to values of each parameter extracted with GetFeature operation as we described in section 6.2.3.

Definition 1 (WFS service model).

Following our definitions from section 4.2 in the Chapter 4, a WFS service \( w \subseteq S \) and can be defined as triple:

\[ RS = (URL, wName, F) \]

where \( URL \) is an address of a WFS service. \( wName \) is the WFS service name, \( F \) is the service features set, which is defined as \( F = \{ f_1, f_2, \ldots, f_n \} \). Specially, each \( F_i(1 \leq f \leq n) \) can be used to invoke a WFS service.

Definition 2 (Feature).

A service feature set is modeled as a tuple:

\[ f_i = \{ \text{name}, \text{O} \} \]

where

- \text{name} is the feature name.
- \( \text{O} \) is the list of parameters associated with the feature.

For example: \( \langle \text{country}, \langle \text{ES}, \text{String} \rangle \rangle \)

Definition 3 (Output parameter model).

In a WFS service output set \( O = \{ o_1, o_2, \ldots, o_n \}, \forall O_i \in O(1 \leq i \leq m) \) its formalization is modeled as a three tuple:

\[ o_i = \{ \text{name}, \text{value}, \text{dataType} \} \]

where

- \text{name} is the output parameter name.
• **value** is the value associated with the name.

• **dataType** is type of value.

For example (capital, madrid, String), (lat, 45.8, decimal)

**Definition 4** (Invocation model (Function)).

A call of a WFS service is a process of invocation and can be designed as

\[ \text{Inv} = (F) \]

where \( F \) is a name of the feature and \( O \) (after invoke the WFS service) is a set of output parameters.

\[ \text{Inv}(\text{country} = \text{ES}, \ldots) \rightarrow \text{Inv}(I_1), \ldots \]

### 6.3 Semantic Annotation Process

Some of the difficulties that arise in the semantic annotation of WFS services were briefly described in [MSD09](#) and Chapter 3. In order to cope with them, we rely on techniques and processes that permit: a) semantic annotation using only the syntactic description of the services and their parameters, and b) semantic annotation by identifying a set of example values that allow the automatic invocation of the service.
6. OGC WEB FEATURE SERVICE ANNOTATION PROCESS

The starting point of the semantic annotation process is the list of syntactic parameters obtained previously, concretely our approach uses information stored in the relational database. Once the WFS service is syntactically described with all its identified parameters, we proceed into its semantic annotation. We follow a very similar heuristic approach that combines a number of external services and semantic resources to propose annotations for the parameters as shown in Figure 6.4. Next, we describe the main components of the semantic annotation.

![Figure 6.4](image_url) - Use of external resources for the semantic annotation process

6.3.1 A model for describing semantically WFS service parameters

In order to describe semantically these services, we define a model (Figure 6.5) to represent relationships of different service parameters with diverse resources used for semantic annotations. This model is based on the service description used in Chapter 4. Nevertheless, the main distinction in the case of the WFS services is that fields of
each feature are considered as parameters. An explanation of this model is summarized in Chapter 4.

Thus, for example, the country_ID parameter belonging to the feature described in section 6.2.3 may be matched to concepts of an ontology, such as: country, countryName, etc. Furthermore, each concept may have a list of instances, for example, the country concept has the following instances such as Spain, Ecuador, Germany, etc. Likewise, each parameter may be associated with elements (object and data properties) of different ontologies.

## 6.3.2 Annotation of WFS service capabilities

The goal of this process is to represent semantically the structure of WFS services using ontologies such as POSM and WSMO-lite, as we illustrated in figure 6.6.

The representation technique was used to describe the WFS service based on the POSM ontology, that is, defining the relationship between them. Moreover, the matching technique was used to find links between WFS characteristics and ontology concepts.

The capabilities annotation process was divided into two parts. Each of these parts comprises its own process. The former was based on the RDF generation using an ontological model. The later was based on to find correspondences between characteristics of getCapabilities and ontology concepts.
The designed process is based on the Semantic Annotations in OGC Standards document proposed by Maue (MSD09). The approach is applied to a sample that is registered syntactically using our model described in section 6.3.1.

The capability annotation process was selected because it is the most suitable for testing the research hypotheses proposed in this work. However, it may incur into errors in case the matching process is unable find coincidences.

Firstly, all the properties of the getCapabilities operation are matched with ontology concepts to find possible links between them. Secondly, an RDF based on the POSM ontology is generated. Finally, the list of features are extracted and processed to generate an RDF based on the POSM ontology. In order to classify capabilities properties into classes and properties of the ontology, a similarity metric was used.

First, the totality of the feature (Method class) is retrieved from the E-R model and linked to POSM classes. Then, each feature in the E-R model is aggregated to the POSM ontology as an instance. They are generated in RDF format as show in figure 6.3.

In the listing 6.11 the WFS service is represented as a rdf:Class and a subclass of the posm:service. GetFeatureType and GetFeature operation inherit from posm:Operation and both provide input and output messages.

**Listing 6.11: WFS-POSM RDF**

```plaintext
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```
6.3 Semantic Annotation Process

@prefix : <http://purl.org/oeg/wfs#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#> .
@prefix posm: <http://www.wsmo.org/ns/posm/0.2#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

:WFSWebService a rdfs:Class ;
   :WFSWebService a posm:service ;
       rdfs:subClassOf posm:Service .

:GetCapabilities a rdfs:Class ;
   rdfs:subClassOf posm:Operation .

:GetCapabilitiesOutput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart .

:GetCapabilitiesInput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart .

:ServiceDescription a rdfs:Class ;
   rdfs:subClassOf [ definir cual] .

:DefaultGetCapabilities a GetCapabilities ;
   posm:hasOutput :DefaultGetCapabilitiesOutput ;
   posm:hasInput :DefaultGetCapabilitiesInput .

:DefaultGetCapabilitiesOutput a GetCapabilitiesOutput ;
   rdfs:label xsd: String .

:DefaultGetCapabilitiesInput a GetCapabilitiesInput ;
   rdfs:label xsd: String .

:DescribeFeatureType a rdfs:Class ;
   rdfs:subClassOf posm:Operation .

:DescribeFeatureTypeInput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart .

:DescribeFeatureTypeOutput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart ;
   rdfs:seeAlso <[URL WFS_describeFeatureTypeResponse] > .

:GetFeature a rdfs:Class ;
   rdfs:subClassOf posm:Operation .

:GetFeatureInput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart .

:GetFeatureOutput a rdfs:Class ;
   rdfs:subClassOf posm:MessagePart ;
   rdfs:seeAlso <[URL WFS_featureCollection] > .

:FeatureTypeDescription a rdfs:Class ;
6. OGC WEB FEATURE SERVICE ANNOTATION PROCESS

The links generated in this process would be checked by an expert. These links are always annotated using the property type. The validation of these links is visual, because, we do not know how to do this validation automatically.

6.3.3 DescribeFeatureType Annotations

Our feature type annotation process is focused exclusively on the name of parameters collected of each feature. The main sources of parameters is collected from already existing WFS, which are available on the internet as we described in section 2.4.1.1. DBpedia, GeoNames ontologies and WGS84 vocabulary are used in the process to find match between WFS parameters and ontological concepts.

The used design in this study is based on Semantic Annotations in OGC Standards described by Maue (MSD09). In this design, parameters are extracted and compared with ontology Concepts using algorithms of similarity metrics and are also stored with possible links into the database using the model described in section 6.3.1. To sum up, the designed process generally permits extracting many possible links between parameters and concepts. At the same time it offers a possibility to test with different values of similarity in the process to check the optimal value. As a consequence, this method produces a consistent match of the parameters and concepts as long as the similarity values are tested.

First, the totality of feature parameters is extracted at the beginning of the process. Within the same step, they are compared with classes and/or properties of the used ontologies using similarity metrics. Thus, most parameters are matched with some ontology components. Nonetheless, the unmatched parameters are stored and marked
as without links in order to process them later. Immediately after, every parameter with possible links is stored into a database using the model described before. Furthermore, the possible links are generated in RDF format based on the ontological model.

Parameters are classified as ontological concepts based on the similarity value (for example, a country parameter is allocated to in the class Country, if it is a value greater than or equal to the similarity value, and discarded if it is a value less than or equal to the similarity value). In order to evaluate this process, there are two criteria for designing a good matching between ontological concepts and the parameters: the selection of the similarity value, and the selection of ontologies and vocabularies.

6.3.4 Enriching with External Resources and annotating

Given definitions size, it is clear that we need to improve the definition list and maximize the number of possible definitions. Hence, we make use external resources to achieve our goal, namely, we used the algorithms proposed in Chapter 4.

6.3.5 Using GetFeature to validate the DescribeFeatureType Annotations

In order to check the collected individuals sample and the initial semantic annotations obtained as a result of the previous process, our system invokes the WFS services that were already registered in the repository and validates the invocation. For the validation of the WFS service, our system selects a random subset of the example instances (of classes and/or properties) coming from GeoNames. Next, it makes several invocations to the WFS service iterating over these registered values.

If the service returns results from the invocation, then the service is considered as executable, and the corresponding annotations are marked as valid. If a service cannot be invoked successfully, the service is classified as non-executable and is automatically discarded from the list of services that can be automatically annotated.

6.3.5.1 Definition of Semantic WFS Annotation

The semantic annotation is defined as a tuple:

\[ SAR = (W, Ont, E) \]

where \( W \) is a WFS service, \( Ont \) is a set of ontologies and \( E \) are external resources.
Algorithm 8 Verification algorithm

Require: URL WFS capabilities

Ensure: Verification (DescribeFeatureType)

\[ \text{verification} \leftarrow 0; \]
\[ \text{verification.URL} \leftarrow \text{URL} \]
\[ \text{instancesDBpediaParameters} \leftarrow \text{getInstancesDBpedia(\text{verification.I}) \{Retrieve a list of instances from DBpedia SPARQL endpoint\}} \]
\[ \text{instancesGeonamesParameters} \leftarrow \text{getInstancesGeonames(\text{verification.I}) \{Retrieve a list of instances from GeoNames SPARQL endpoint, only if the parameter are geospatial\}} \]

for all instance \( \in \text{instancesDBpediaParameters} \)

\[ \text{result} \leftarrow \text{callWFSFilter(instance)}; \]
\[ \text{if result then} \]
\[ \text{invocation successfully} \]
\[ \text{end if} \]

end for{for geospatial parameters}

for all instance \( \in \text{instancesGeonamesParameters} \)

\[ \text{result} \leftarrow \text{callWFSFilter(instance)}; \]
\[ \text{if result then} \]
\[ \text{invocation successfully} \]
\[ \text{end if} \]

end for
6.3.5.2 Possible semantic definition of WFS services

The use of algorithms described in Chapter 4 allows generating a semantic definition of a WFS service with respect to the used ontology resources. The system takes each parameter and tries it to a link with ontological resources. For example, the country NAME parameter that is an output parameter of the Service 1 may have the following possible links (Listing 6.12).

```
Listing 6.12: Possible ontological definition

1 country
2 countryName
3 Country
```

The combination of each parameter (with one or more links) with other parameters can be used to generate a large list of possible definitions. In this sense, the challenge that arises is to determine the best real combination. As a first step, we discard all the links having no instances or a number of instances less than 10. As it can be seen, each link to ontology has a list of instances that may be used to obtain the best definition.

6.3.6 Checking the semantic annotation of WFS services

In order to check the collected sample individuals and the initial semantic annotations obtained as a result of the previous process, our system invokes the WFS services that were already registered in the repository and validates the input and output parameters for checking which is the best option to describe each parameter.

This process is similar to the one described in the previous Chapter.

6.4 Conclusions

We have presented an approach for registering and semantically annotating based on existing syntactic service description. The process starts from a model used for representing WFS services, which depicts the information that the WFS service provides. Furthermore, we provide a precise definition of the established model used for representing WFS services. This model is one of the main contributions of this work. We have shown the benefits obtained from our process, not only on syntactic registration but also on meaning. So far, it is possible to define these semantic annotations automatically and with limited ontologies. Future work will concentrate on methods that
allow the automatic insertion of new ontologies into the system. The Chapter explains a prototypical implementation of the introduced concepts.
7

Evaluation

In the previous chapters we presented our approach for annotating RESTful and OGC Web services. This chapter presents the results of the evaluation of the two approaches presented in the preceding chapters. It describes an evaluation study of semantic annotation of Web services. Given ontologies and a set of Web services, we analyze the possibility of finding a match between Web service parameters and ontology concepts.

Our evaluation confirms that such automatic process is faster and produces more homogeneous metadata than manual annotations. Hence validating hypothesis H1 and H4.

7.1 Gold Standard

We have generated a gold standard with the studied services where we have assigned manually the annotations that have to be produced for all input/output parameters of these services.

We have collected a set of 60 active services. These services were manually reviewed by an expert. The expert assigned links for all parameters belonging to the service. The existing annotations served as a gold standard for us to measure the accuracy of the automated process.

7.1.1 Method

As mentioned above, there is no gold standard for evaluating existing semantic annotation processes of RESTful and WFS services.
7. EVALUATION

To obtain a diverse sample, we selected services with different characteristics: number of parameters, optional parameters, special parameters, type of result, available, etc. All services in the set are related to the geospatial domain, as shown in appendix F.

The expert was: a web service consultant with extensive experience in content development of web application using RESTful and WFS Web services. The expert went through the following 4 steps to assess the validity of the services:

1. **Establish initial criteria.** The initial criteria required the expert to validate the services. The validation represents the checking of services through their invocation. Using these criteria, the expert selected a set of 100 services for RESTful and 15 for WFS.

2. **Expert review to refine the criteria.** In this step, the services selected above were compared and analyzed. It was found that, for 40 services the results of the invocation contain numerous special parameters such as navigation parameters, pagination parameters, parameters without meaning, etc. Using the refined criteria the expert decided to eliminate these, as a consequence, the expert generated a list with the of 60 most representative RESTful services and 5 WFS services. Since quite a few services restrict access to registered users, we only tested those services for which we could obtain licence keys. Furthermore, we filtered out services that do not provide information about geospatial data, because these services included parameters that were not in our domain model.

3. **Expert stores the results of the invocation.** In this step, the services selected were invocated and stored in separate files. Each service was invoked several times using different input parameters. Furthermore, each service was analyzed to extract a list of parameters that may be annotated with ontologies.

4. **Parameter annotations.** For each service, the expert selected all parameters with their respective values and analyzed the RDF repository of each ontology used to find possible links and decide the best option. For the analysis of the RDF repository, the expert used a SPARQL query. As a result, the expert obtained a list of correspondences between parameters and ontology concepts (class/property).
7.2 Similarity metrics

7.1.2 Selecting ontologies

Selecting an appropriate ontology for semantic annotation is a step of the semantic annotation process, we provide a brief introduction to ontologies selected to annotate semantically the RESTful services. These ontologies were selected because they have geospatial information. We have used the ontologies described in the Chapter 2.

7.2 Similarity metrics

In the experiments, different procedures were used to decide equality between different values of the same type. Some of the equality procedures (and thresholds) used for different types are listed in table 7.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Equality Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude, longitude</td>
<td>accuracy bound of +-0.002</td>
</tr>
<tr>
<td>bbox</td>
<td>accuracy bound of +-0.005</td>
</tr>
<tr>
<td>city,country,etc.</td>
<td>specialised equality procedure</td>
</tr>
</tbody>
</table>

Table 7.1: Equality procedures used in the experiments

For all types not listed in the table, substring matching was used to test equality between values.

We consider different similarity metrics to compare parameters with the labels of the elements of these ontologies. Each metric evaluates to a similarity value in the range [0, 1], with higher values indicating a greater similarity. The used metrics are classified into two groups: equality and similarity. The former collects metrics that evaluate to a similarity value like 0 or 1. This group is composed of a set of metrics, such as: Block distance, Cosine similarity, Dices Coefficient, Jaccard Similarity, Matching Coefficient, and Overlap Coefficient. The latter gathers metrics that evaluate to a similarity value between 0 and 1. This group is composed of diverse metrics, such as: Jaro Winkler, Levenshtein Distance, Monge Elkan distance, Needleman-Wunch distance, and Smith-Waterman-Gotoh. These proposals allow matching between strings, which are related to service parameters and ontology components, such as countryName, country_name, or country. An example of this matching process, using metrics of similarity group, is shown in Table 7.2.
7. EVALUATION

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DBpedia ontology (classes)</th>
<th>Jaro Winkler</th>
<th>Levenshtein</th>
<th>Monge Elkan</th>
<th>Needleman-wunch</th>
<th>Smith-Waterman-Goto</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
<td>County</td>
<td>0.94</td>
<td>0.58</td>
<td>1.0</td>
<td>0.54</td>
<td>1.0</td>
</tr>
<tr>
<td>countryName</td>
<td>County</td>
<td>0.95</td>
<td>0.64</td>
<td>1.0</td>
<td>0.64</td>
<td>1.0</td>
</tr>
<tr>
<td>city</td>
<td>City</td>
<td>0.89</td>
<td>0.44</td>
<td>1.0</td>
<td>0.54</td>
<td>1.0</td>
</tr>
<tr>
<td>population</td>
<td>populationDensity</td>
<td>0.60</td>
<td>0.50</td>
<td>0.80</td>
<td>0.56</td>
<td>0.80</td>
</tr>
<tr>
<td>currencyCode</td>
<td>Currency</td>
<td>0.96</td>
<td>0.67</td>
<td>1.0</td>
<td>0.42</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 7.2: Equality procedures used in the experiments

7.3 RESTful Evaluation

In order to evaluate our approach in the geospatial domain we have used 60 different RESTful services from http://www.programmableweb.com/ selected by the expert. The list of these services can be found in our experiment website (http://www.oeg-upm.net/files/RESTFulAnnotationWeb/ SourcesList/sources.ods).

7.3.1 Syntactic Registration

In the syntactic registration of all these services in the system, by means of introducing the list of their URLs, our system successfully registered 56 of them into the repository (4 services could not be registered due to an invocation error). As a result of this syntactic registration, the system has produced a complete list of 1,014 total parameters (152 input parameters and 862 output parameters) with some duplications. Of the 1,014 parameters, our system obtains 394 parameters without duplications (52 input parameters and 342 output parameters). In order to detect possible duplicates in the combination with input and output parameters, the system performs the union of both (input and output) parameters. From this union, 369 parameters are obtained without duplications.

7.3.2 Semantic Annotations

7.3.2.1 Simple Match

This analysis follows the three steps described in our annotation process. First, our system identifies correctly 191 of 369 different parameters without duplications by calling directly the DBpedia and GeoNames ontologies.
7.3 RESTful Evaluation

7.3.2.2 Match with Enrichment Parameters with External Resources

Second, the system uses initial parameters plus the suggestion service and calls the DBpedia and GeoNames ontologies. In this case, it adds 57 additional names for parameters to the initial ones and identifies 33 correspondences with used ontologies. Third, the system uses the initial parameters plus the synonyms service, and calls the DBpedia and GeoNames ontologies. It incorporates 1,147 additional names for parameters into the system and identifies 126 correspondences. Finally, the system combines all the resources that result from the enrichment process and calls again the DBpedia and GeoNames SPARQL Endpoint. Here it adds 1,573 more names for initial parameters and identifies 159 correspondences. A detailed view of these results is shown in Table 7.3.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Unique-Parameters</th>
<th>Additional names</th>
<th>Parameters and names</th>
<th>Matching with used ontologies</th>
<th>Total matchings</th>
</tr>
</thead>
<tbody>
<tr>
<td>a)Parameters</td>
<td>369</td>
<td>-</td>
<td>369</td>
<td>191</td>
<td>191</td>
</tr>
<tr>
<td>b)Alternative Spellings (Suggestions)</td>
<td>-</td>
<td>57</td>
<td>426</td>
<td>33</td>
<td>244</td>
</tr>
<tr>
<td>c)Synonyms</td>
<td>-</td>
<td>1147</td>
<td>1516</td>
<td>126</td>
<td>357</td>
</tr>
<tr>
<td>d)Suggestions + Synonyms</td>
<td>-</td>
<td>1204</td>
<td>1573</td>
<td>159</td>
<td>330</td>
</tr>
</tbody>
</table>

Table 7.3: Enriching initial parameters with external resources

7.3.2.3 Input Parameter Evaluation

With respect to the validation of input parameters (Table 7.4), our system recognizes 152 inputs (with duplications) of the initial list, of which 80 parameters can be annotated automatically with the DBpedia (35 parameters) and GeoNames (45 parameters) ontologies.

<table>
<thead>
<tr>
<th>RESTful services</th>
<th>Total parameters</th>
<th>Annotated Parameters</th>
<th>Annotated parameters (DBpedia)</th>
<th>Annotated parameters (GeoNames)</th>
<th>Special parameters</th>
<th>Service Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input parameters</td>
<td>152</td>
<td>80</td>
<td>35</td>
<td>41</td>
<td>72</td>
<td>56</td>
</tr>
<tr>
<td>Output parameters</td>
<td>362</td>
<td>335</td>
<td>202</td>
<td>113</td>
<td>298</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.4: Results of the input and output parameters
7. EVALUATION

7.3.2.4 Special Parameter Evaluation

Likewise, we have discovered with our evaluation that some other parameters are useless in terms of semantic annotation processes, due to lack of annotations or meaningful results. These parameters are called special ones. After the exhaustive analysis performed in this evaluation, we classify these parameters into two groups:

- **Navigation.** Those parameters referred to the navigation process through the RESTful service results, for example: link, userID, hits, total, pages, etc.

- **Unknown.** Those parameters described by only one letter (e.g., q, l, s, etc.), and hence not sufficiently descriptive for our automated approach to find any correspondence.

These parameters are not considered for this validation, concretely 155 navigation parameters and 72 unknown parameters, which were detected manually and a list of them is collected in this website (http://www.oeg-upm.net/files/RESTFulAnnotationWeb/parameters/Parameters.ods).

One aspect of our system is that we cannot always guarantee a successful annotation, because in some cases the system cannot find any correspondence between the service parameters and components of the DBpedia, GeoNames or WGS84 ontologies/vocabulary. This is common, for instance, when RESTful services contain the aforementioned parameters. In these cases the parameters should be shown to users for a manual description of them. Our system takes them out automatically from the service registration process.

In summary, for 56 of the 60 initial geospatial RESTful services we have obtained correct input parameter associations, except for 4 cases where we could not find any correspondence.

7.3.3 Output Parameter Evaluation

With respect to the validation of output parameters (7.4), our system recognizes 862 outputs (with duplications) that belong to the 56 services whose input parameters have been validated. This total of output parameters is divided into 315 whose correspondences can be found using the DBpedia (202 parameters) and GeoNames (113
parameters) ontologies, and 387 parameters (navigation (298) and unknown (89)) that are not considered for this validation, because they cannot be found.

While in the context of the input parameters we are interested in determining whether we can call the service or not, in the case of output parameters, we are interested in the precision and recall metrics of the annotation process. Hence, we have generated a gold standard with the studied services in order to assign manually the annotations that have to be produced for all output parameters of these services, and we have performed an evaluation of the results obtained from the system for the parameters that are found. Regarding the parameters that are found (Table 7.5), our system annotates 315 of them automatically, from which 315 parameters are annotated correctly according to the gold standard, while 160 parameters are not annotated by the system. This provides us with an average value for precision (the fraction of annotated parameters that are retrieved, that is, (Right parameters ) / (Annotated parameters ) equal to 0.85 and recall (the fraction of relevant annotated parameters that are retrieved, that is, (Right parameters ) / total parameters ? Special parameters equal to 0.56 for both metrics.

<table>
<thead>
<tr>
<th>RESTful service</th>
<th>Total parameters</th>
<th>Not found Parameters</th>
<th>Annotated</th>
<th>Not Annotated</th>
<th>Annotated and not annotated</th>
<th>Right parameters</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output parameters</td>
<td>862</td>
<td>89</td>
<td>315</td>
<td>160</td>
<td>475</td>
<td>315</td>
<td>0.85</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 7.5: Output parameters metrics

7.3.4 Conclusions

To the best of our knowledge, there are no available results from existing research works to compare our results against. Likewise, these preliminary results prove the feasibility of our system and highlight that it is possible to carry out an assisted semantic annotation of RESTful services. Hence validating hypothesis H1 and H4.

7.4 WFS Evaluation

In this section we evaluate our solution related to the annotation of WFS services. The aim is to check whether and to what extent developed solutions satisfy the requirements
7. EVALUATION

as an infrastructure to automatically generate semantic descriptions for WFS services
as identified in Chapter 6.

In order to evaluate our approach, we have used 71 different WFS services obtained
from LPRANI+12, which we have selected randomly from those that were available.
The list of these services can be found in appendix F

7.4.1 Syntactic Registration

The process of syntactic registration of all these services in the system, by means of
introducing the list of their URLs, is described in (Sections 6.1 and 6.2).

The syntactic descriptions of the WFS contain data from different areas consisting
of specific features, while other consisting of different features. Each feature represents
an area in the real world whose shape and location is represented by a geometry.

After the initial invocation of the 71 WFS services, our system reported that suc-
cessfully invoked 44 WFS services (27 could not be invoked due to invocation error).
Of the 44 WFS services, 22 services have features with parameters and 22 only have
the name of the feature without parameters. The total features of WFS services are
4792, which 2495 features have parameters and 2333 features have not parameters.

Our system successfully registered 22 of them into the repository (49 services could
not be registered due to an invocation error or due to the lack of parameters). As a
result of this syntactic registration, the system has produced a complete list of 30360
total parameters (all of them output parameters) with some duplications. Of the 30360
parameters, our system obtains 5418 parameters without duplications.

7.4.2 Semantic Annotation

7.4.2.1 Simple Match and Match with External Resources

This analysis follows the three steps described in our annotation process (See Chapter
4). First, our system calls used ontologies (DBpedia, WFS and Geonames) using pa-
parameters related to different WFS services. Then, it identifies correctly 1980 of 5148
different parameters without duplications by calling directly the DBpedia, WFS and
GeoNames ontologies.

Second, the system uses initial parameters plus the suggestion service and calls the
DBpedia, WFS and GeoNames ontologies. In this case, it adds 768 additional names for
parameters to the initial ones and identifies 325 correspondences with used ontologies. Third, the system uses the initial parameters plus the synonyms service, and calls the DBpedia, WFS and GeoNames ontologies. It incorporates 2025 additional names for parameters into the system and identifies 235 correspondences. Finally, the system combines all the resources that result from the enrichment process and calls again the DBpedia, WFS and GeoNames SPARQL Endpoint. Here it adds 8229 more names for initial parameters and identifies 198 correspondences. A detailed view of these results is shown in Table 7.6.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Unique-Parameters</th>
<th>Additional names</th>
<th>Parameters and names</th>
<th>Matching with used ontologies</th>
<th>Total matchings</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Parameters</td>
<td>5418</td>
<td>-</td>
<td>5418</td>
<td>198</td>
<td>1980</td>
</tr>
<tr>
<td>(b) Alternative Spellings (Suggestions)</td>
<td>-</td>
<td>786</td>
<td>6204</td>
<td>125</td>
<td>2305</td>
</tr>
<tr>
<td>(c) Synonyms</td>
<td>-</td>
<td>2025</td>
<td>7443</td>
<td>235</td>
<td>2540</td>
</tr>
<tr>
<td>(d) Suggestions + Synonyms</td>
<td>-</td>
<td>2811</td>
<td>8229</td>
<td>198</td>
<td>2738</td>
</tr>
</tbody>
</table>

Table 7.6: Enriching initial parameters with external resources

7.4.2.2 Input Parameter Evaluation

With respect to the validation of input parameters, our system does not evaluate it, because, a WFS service has not input parameters. Therefore, our system only takes URL of this kind of services. This is a special characteristic of this type of services with respect to RESTful services.

7.4.2.3 Special Parameter Evaluation

Likewise, we have discovered with our evaluation that some other parameters are useless in terms of semantic annotation processes, due to lack of annotations or meaningful results. These parameters are called special ones. After the exhaustive analysis performed in this evaluation, we classify these parameters into two groups:

- **Geospatial.** Those parameters referred to geospatial data that appear within the WFS service results, for example: gid, lat, long, area, bbox, etc.

- **Unknown.** Those parameters described by only one or two letter (e.g., ac, acm, q, l, s, etc.), and hence not sufficiently descriptive for our automated approach to find any correspondence.
7. EVALUATION

These special parameters are not considered for this validation, concretely 7 geospatial parameters and 48 unknown parameters, which were detected manually and a list of them is collected in our website.

One aspect of our system is that we cannot always guarantee a successful annotation, because in some cases the system cannot find any correspondence between the service parameters and components of the DBpedia, GeoNames or WFS ontologies/vocabulary. This is common, for instance, when WFS services contain the aforementioned parameters. In these cases the parameters should be shown to users for a manual description of them. Our system takes them out automatically from the service registration process.

7.4.3 Output Parameter Evaluation

With respect to the validation of output parameters (see Table 7.7), our system recognizes 5,418 outputs (with duplications) that belong to the 22 services whose input parameters have been validated. This total of output parameters is divided into 1,453 whose correspondences can be found using aforementioned ontologies, and 2,680 parameters that are not considered for this validation, because they cannot be found.

In the output parameters, we are interested in the precision and recall metrics of the annotation process. Hence, we have generated a gold standard with the studied services in order to manually assign the annotations that have to be produced for all output parameters of these services. Taking this gold standard into account, we have performed an evaluation of the results obtained from the system for the parameters that have been found. Regarding the parameters that are found (7.7), our system annotates 1,453 of them automatically, from which 763 parameters are annotated correctly according to the gold standard, whereas 1,285 parameters are not annotated by the system. This provides us with an average precision value (the fraction of annotated parameters that are retrieved, that is, (Right parameters)/(Annotated parameters) equal to 0.53 and recall (the fraction of relevant annotated parameters that are retrieved, that is, (Right parameters) / total parameters - Special parameters equal to 0.54 for both metrics.

\[ http://www.oeg-upm.net/files/WFSAnnotationWeb/parameters/Parameters.ods \]
Table 7.7: Output parameters metrics

<table>
<thead>
<tr>
<th>WFS services</th>
<th>Total parameters</th>
<th>Not found Parameters</th>
<th>Annotated</th>
<th>Not Annotated</th>
<th>Annotated and not annotated</th>
<th>Right parameters</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output parameters</td>
<td>5418</td>
<td>2680</td>
<td>1453</td>
<td>1285</td>
<td>2738</td>
<td>763</td>
<td>0.53</td>
<td>0.54</td>
</tr>
</tbody>
</table>

7.4.4 Conclusions

We evaluated our approach in the context of WFS services. In the evaluation process, we used a set of WFS services and compare them with a developed gold standard. With the obtained results, we managed to achieve a significant improvement with respect the initial parameters, this is, we enrich the parameters with external resources. To the best of our knowledge, there are no available results from existing research works to compare our results against.

Likewise, these preliminary results prove the feasibility of our system and highlight that it is possible to carry out an assisted semantic annotation of WFS services. Hence validating hypothesis H1 and H4.
7. EVALUATION
8

Conclusions and future work

This thesis began with the observation that the semantic descriptions of Web services is a prerequisite for the Semantic Web. Despite the tremendous success of service-based applications and environments during the last years, the adequate support for service search and consumption remains an unsolved problem. Clearly, there is a large annotation problem that cannot be successfully approached by using a manual annotation. This approach has shown how the problem of semantic annotation of geospatial Web services (REST and WFS) can be tackled by means of a comprehensive automatic annotation process. Furthermore, we have demonstrated that automatic semantic annotation of geospatial Web services is inherently more powerful than manual annotation. These problems can be characterized by the presence of manual descriptions found in Web services providers, therefore, it is possible to approach them successfully by the automatic semantic annotation tools and techniques subject of this thesis. For that reason, this thesis addressed, the overall research questions of how can semantic annotation for existing geospatial services be created.

Taking into account the aforementioned scenario, in this thesis we have proposed an approach to support the semantic annotation of geospatial RESTful and WFS services. This process is implemented in a system that takes into account ontologies for a general, and specific geospatial annotation, as well as different external resources such as synonyms and suggestion services. Our system uses combinations of these resources in order to discover meanings for each of the parameters of the RESTful and WFS services for performing semantic annotations of them. To illustrate our work and guide the explanations of the proposed semantic annotation process, we have used exem-
8. CONCLUSIONS AND FUTURE WORK

plary RESTful and WFS services related to the geospatial domain. Besides, we have presented some preliminary experimental results that prove the feasibility of our approach, in the geospatial domain, and have shown that it is possible to assist semantic annotation of RESTful and WFS services in this domain.

8.1 Contributions

This section reviews the main contributions of this thesis and how we have solved the open research problems. We defined the following objectives:

- O1. The definition of a conceptual model for annotating Web services.
- O2. The definition of a process related to semantic annotation of a Web service.
- O3. The creation of a system that implements the annotation process.
- O4. The definition of a gold standard for evaluating annotation processess for RESTful and OGC WFS services.
- O5. The development of a system to evaluate the quality of the annotation process.

We have generated the following contributions:

C1. A model for representing a Web service. Given the Web service, we identified a model that allows 1) describing available RESTful and WFS services and 2) including data (instances) related to a Web service.

C2. A method for annotating Web service using ontological and external resources. In relation to the semantic annotation process, the result of our investigation is a general method aimed to contribute towards the unified Web services process annotation. It allows the semantic enriching of the parameters that a Web service have. This thesis show how to exploit ontological and external resources easily for annotation of Web services.

C3. A system that implements the annotation process. The software library covers the annotation of Web services and reduces the effort of semantic annotations. This library can be applied to RESTful and OGC WFS services and
8.2 Opportunities for Further Research

employs a comprehensive set of technological modules including registration and invocation services, automatic annotation, automatic validation, and external resources registration and invocation. This library has been prototypically implemented.

C4. **A gold standard for evaluating the annotation of RESTful and OGC WFS services.** A gold standard generated to RESTful and OGC WFS services using a set of studied services. This gold standard has been created by experts.

C5. **Algorithms for evaluating the annotations.** The thesis shows how to evaluate semantic annotations using an automatic process applied within a gold standard. It provides a case study for evaluating human annotation in comparison with automatic annotation evaluating the annotation library.

8.2 Opportunities for Further Research

In this thesis we have tackled many open research problems within the context of semantic annotation of geospatial Web services, which is a novel and relatively unexplored approach. It is natural that it raises new questions while it answers old ones. Hence, the general problem of automatic semantic annotation of Web services remains interesting. In this section, we present our future work:

- Development of a GUI to facilitate semantic annotation of existing services by users, probably it will incorporate in any existing RESTful or WFS semantic annotation tool/utility suite.

- We also plan to make improvements to the proposed system through the analysis of instances retrieved in the matching process. In the same sense, we will also aim at improving the SPARQL queries to DBpedia and other cross-domain semantic resources (such as, OpenCyc, http://www.opencyc.org/, UMBEL, etc.), freebase, in order to better explore these resources in the annotation process, and optimize the use of suggestion and synonyms services.

- We also plan to generate Linked Data of services annotations. The reason for this is that, we want to have a RDF repository of description of Web services.
8. CONCLUSIONS AND FUTURE WORK

• We will combine the semantic annotation with inductive logic programming approaches to learn relationships between sources and parameters. We believe that heuristic techniques developed in the inductive logic programming community may be applicable to the annotation problems.

• We will incorporate more specific domain ontologies in the semantic annotation process for taking advantage of domain specific characteristics. It is important to comment that the success of our approach is strongly based on the existence of domain ontologies that provide good coverage for the domain of the annotated services.

• We will incorporate methods for discovering automatically ontologies related with services and discovering the mean of services using identity recognition approaches and data mining techniques.
Appendices
Appendix A

Possible definitions

Below we show the intermediate results of the annotation process of RESTful service.

Listing A.1: Possible definition before semantic process

```plaintext
1 Syntaxic definition: http://ws.geonames.org/countryInfo
2 countryInfo($country, isoAlpha3, continent, capital, countryCode, bBoxEast, fipsCode,
3  countryName, population, bBoxWest, languages, isoNumeric, geonameId, areaInSqKm,
4  bBoxSouth, currencyCode, bBoxNorth)
5 Possible semantic definitions:
6  Input parameters:
7    country:
8      http://dbpedia.org/ontology/countryWithFirstSpaceflight --> 7 instances
9      http://dbpedia.org/ontology/locationCountry --> 1758 instances
10     http://dbpedia.org/ontology/sourceConfluenceCountry --> 0 instances
11     http://dbpedia.org/ontology/managementCountry --> 2 instances
12     http://dbpedia.org/ontology/countryOrigin --> 152 instances
13     http://dbpedia.org/ontology/capitalCountry --> 0 instances
14     http://dbpedia.org/ontology/fastestDriverCountry --> 65 instances
15     http://dbpedia.org/ontology/countryWithFirstSatellite --> 16 instances
16     http://dbpedia.org/ontology/poleDriverCountry --> 66 instances
17     http://dbpedia.org/ontology/Country --> 1846 instances
18     http://dbpedia.org/ontology/mouthCountry --> 326 instances
19     http://dbpedia.org/ontology/country --> 1679 instances
20     http://dbpedia.org/ontology/usingCountry --> 288 instances
21     http://dbpedia.org/ontology/secondDriverCountry --> 75 instances
22     http://dbpedia.org/ontology/firstDriverCountry --> 75 instances
23     http://dbpedia.org/ontology/thirdDriverCountry --> 77 instances
24     http://dbpedia.org/ontology/sourceCountry --> 298 instances
25     http://dbpedia.org/ontology/twinCountry --> 65 instances
26     http://dbpedia.org/ontology/governmentCountry --> 0 instances
27     http://dbpedia.org/ontology/countryWithFirstAstronaut --> 19 instances
28     http://dbpedia.org/ontology/countryWithFirstSatelliteLaunched --> 8 instances
29  Synonyms: landed estate:
```

161
A. POSSIBLE DEFINITIONS

http://dbpedia.org/ontology/state-->1562 instances
Synonyms: domain:
http://dbpedia.org/ontology/domain-->1694 instances

http://dbpedia.org/ontology/state-->1562 instances
http://dbpedia.org/ontology/sourceState-->0 instances
http://dbpedia.org/ontology/sourceConfluenceState-->19 instances

Synonyms: province:
http://dbpedia.org/ontology/province-->1912 instances

Synonyms: democracy:

Synonyms: terra firma:

Synonyms: area:
http://dbpedia.org/ontology/councilArea-->1565 instances
http://dbpedia.org/ontology/Building/floorArea-->1330 instances
http://dbpedia.org/ontology/PopulatedPlace/areaMetro-->855 instances
http://dbpedia.org/ontology/areaLand-->1999 instances
http://dbpedia.org/ontology/areaMetro-->895 instances
http://dbpedia.org/ontology/area-->1814 instances
http://dbpedia.org/ontology/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/filmFareAward-->43 instances
http://dbpedia.org/ontology/percentageOfAreaWaterRound-->0 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances
http://dbpedia.org/ontology/PopulatedPlace/areaUrban-->276 instances
http://dbpedia.org/ontology/Lake/areaOfCatchment-->682 instances
http://dbpedia.org/ontology/lieutenancyArea-->769 instances
http://dbpedia.org/ontology/areaUrban-->276 instances
http://dbpedia.org/ontology/areaTotal-->2000 instances
http://dbpedia.org/ontology/locatedInArea-->1632 instances

Synonyms: republic:

Synonyms: body politic:

Synonyms: land:
http://dbpedia.org/ontology/lunarLandingSite-->0 instances
http://dbpedia.org/ontology/numberOfIslands-->47 instances
http://dbpedia.org/ontology/areaLand-->1999 instances
http://dbpedia.org/ontology/landingDate-->319 instances
http://dbpedia.org/ontology/majorIsland-->98 instances
http://dbpedia.org/ontology/dorlandsPrefix-->299 instances
http://dbpedia.org/ontology/landingvehicle-->0 instances
http://dbpedia.org/ontology/island-->136 instances
http://dbpedia.org/ontology/Island-->1709 instances
84 http://dbpedia.org/ontology/landingSite --> 0 instances
85 http://dbpedia.org/ontology/dorlandsSuffix --> 2000 instances
86 Synonyms: state:
87 http://dbpedia.org/ontology/lowestState --> 1 instances
88 http://dbpedia.org/ontology/sourceState --> 0 instances
89 http://dbpedia.org/ontology/SupremeCourtOfTheUnitedStatesCase --> 1935 instances
90 http://dbpedia.org/ontology/stateOfOrigin --> 1802 instances
91 http://dbpedia.org/ontology/federalState --> 1928 instances
92 http://dbpedia.org/ontology/mouthState --> 0 instances
93 http://dbpedia.org/ontology/highestState --> 11 instances
94 http://dbpedia.org/ontology/stateDelegate --> 12 instances
95 http://dbpedia.org/ontology/state --> 1562 instances
96 http://dbpedia.org/ontology/sourceConfluenceState --> 19 instances
97 http://dbpedia.org/ontology/states --> 0 instances
98 http://dbpedia.org/ontology/unitedStatesNationalBridgeId --> 278 instances
99 Synonyms: acres:
100 Synonyms: res publica:
101 Synonyms: commonwealth:
102 Synonyms: soil:
103 Synonyms: demesne:
104 Synonyms: farming:
105 http://dbpedia.org/ontology/arm --> 17 instances
106 Synonyms: solid ground:
107 http://dbpedia.org/ontology/ground --> 1846 instances
108 http://dbpedia.org/ontology/id --> 859 instances
109 Synonyms: kingdom:
110 http://dbpedia.org/ontology/kingdom --> 1967 instances
111 Synonyms: earth:
112 http://dbpedia.org/ontology/lowerEarthOrbitPayload --> 49 instances
113 http://dbpedia.org/ontology/Rocket/lowerEarthOrbitPayload --> 49 instances
114 Synonyms: nation:
115 http://dbpedia.org/ontology/unitedStatesNationalBridgeId --> 278 instances
116 http://dbpedia.org/ontology/NationalCollegiateAthleticAssociationAthlete --> 388 instances
117 http://dbpedia.org/ontology/inclination --> 1 instances
118 http://dbpedia.org/ontology/orbitalInclination --> 56 instances
119 http://dbpedia.org/ontology/nationalOlympicCommittee --> 1 instances
120 http://dbpedia.org/ontology/nationalTeam --> 34 instances
121 http://dbpedia.org/ontology/goalsInNationalTeam --> 0 instances
122 http://dbpedia.org/ontology/nationalRanking --> 22 instances
123 http://dbpedia.org/ontology/nationalOrigin --> 0 instances
124 http://dbpedia.org/ontology/maximumInclination --> 0 instances
125 http://dbpedia.org/ontology/appearancesInNationalTeam --> 0 instances
126 http://dbpedia.org/ontology/denomination --> 462 instances
127 http://dbpedia.org/ontology/nationality --> 1772 instances
128 http://dbpedia.org/ontology/numberOfParticipatingNations --> 37 instances
129 http://dbpedia.org/ontology/nationalFilmAward --> 23 instances
130 http://dbpedia.org/ontology/nationalYears --> 0 instances
131 http://dbpedia.org/ontology/destination --> 128 instances
132 http://dbpedia.org/ontology/minimumInclination --> 0 instances
133 http://dbpedia.org/ontology/internationally --> 0 instances
134 http://dbpedia.org/ontology/destinations
135
163
A. POSSIBLE DEFINITIONS

Synonyms: rural area:

http://dbpedia.org/ontology/area—>1814 instances

Synonyms: dry land:

http://dbpedia.org/ontology/ground—>1846 instances

http://dbpedia.org/ontology/background—>1014 instances

Synonyms: state of matter:

http://dbpedia.org/ontology/state—>1502 instances

Output parameters:

isoAlpha3:

http://dbpedia.org/ontology/Continent—>9 instances

Synonyms: celibate:

capital:

http://dbpedia.org/ontology/capitalCountry—>0 instances

http://dbpedia.org/ontology/marketCapitalisation—>0 instances

http://dbpedia.org/ontology/capitalRegion—>0 instances

http://dbpedia.org/ontology/capital—>1838 instances

http://dbpedia.org/ontology/capitalPlace—>0 instances

http://dbpedia.org/ontology/capitalDistrict—>0 instances

http://dbpedia.org/ontology/capitalMountain—>0 instances

http://dbpedia.org/ontology/capitalPosition—>84 instances

http://dbpedia.org/ontology/capitalElevation—>37 instances

Synonyms: chapter:

http://dbpedia.org/ontology/Work—>1265 instances

http://dbpedia.org/ontology/capital—>1838 instances

Synonyms: capital letter:

http://dbpedia.org/ontology/capital—>1838 instances

Synonyms: cap:

http://dbpedia.org/ontology/capitalPlace—>0 instances

http://dbpedia.org/ontology/capitalPosition—>84 instances

http://dbpedia.org/ontology/capitalRegion—>0 instances

http://dbpedia.org/ontology/capitalCountry—>84 instances

http://dbpedia.org/ontology/Car/Automobile/fuelCapacity—>181 instances

http://dbpedia.org/ontology/capitalElevation—>37 instances

http://dbpedia.org/ontology/capitalMountain—>0 instances

http://dbpedia.org/ontology/capitalDistrict—>0 instances

http://dbpedia.org/ontology/marketCapitalisation—>0 instances

Synonyms: uppercase:

Synonyms: upper-case letter:

Synonyms: majuscule:

countryCode:

http://dbpedia.org/ontology/code—>190 instances

http://dbpedia.org/ontology/country—>1679 instances

http://dbpedia.org/ontology/Country—>1846 instances

Suggest: Code:

http://dbpedia.org/ontology/inseeCode—>0 instances
http://dbpedia.org/ontology/oldcode --> 1 instances
http://dbpedia.org/ontology/schoolCode --> 239 instances
http://dbpedia.org/ontology/code --> 190 instances

Suggest: country:
http://dbpedia.org/ontology/thirdDriverCountry --> 77 instances
http://dbpedia.org/ontology/sourceConfluenceCountry --> 0 instances
http://dbpedia.org/ontology/locationCountry --> 1758 instances
http://dbpedia.org/ontology/governmentCountry --> 0 instances
http://dbpedia.org/ontology/countryWithFirstAstronaut --> 19 instances
http://dbpedia.org/ontology/poleDriverCountry --> 66 instances
http://dbpedia.org/ontology/secondDriverCountry --> 75 instances
http://dbpedia.org/ontology/firstDriverCountry --> 75 instances
http://dbpedia.org/ontology/countryOrigin --> 152 instances
http://dbpedia.org/ontology/country --> 1679 instances
http://dbpedia.org/ontology/countryWithFirstSatelliteLaunched --> 8 instances
http://dbpedia.org/ontology/countryWithFirstSatellite --> 16 instances
http://dbpedia.org/ontology/mouthCountry --> 326 instances
http://dbpedia.org/ontology/fastestDriverCountry --> 65 instances
http://dbpedia.org/ontology/usingCountry --> 288 instances
http://dbpedia.org/ontology/sourceCountry --> 298 instances
http://dbpedia.org/ontology/Country --> 1846 instances

bboxEast:

fipsCode:
http://dbpedia.org/ontology/code --> 190 instances
Suggest: fips:
Suggest: Code:
http://dbpedia.org/ontology/code --> 190 instances
http://dbpedia.org/ontology/oldcode --> 1 instances
http://dbpedia.org/ontology/schoolCode --> 239 instances
http://dbpedia.org/ontology/inseeCode --> 0 instances

countryName:
http://dbpedia.org/ontology/country --> 1679 instances
http://dbpedia.org/ontology/Country --> 1846 instances
Suggest: country:
http://dbpedia.org/ontology/twinCountry --> 65 instances
http://dbpedia.org/ontology/governmentCountry --> 0 instances
http://dbpedia.org/ontology/countryOrigin --> 152 instances
http://dbpedia.org/ontology/country --> 1679 instances
http://dbpedia.org/ontology/poleDriverCountry --> 66 instances
http://dbpedia.org/ontology/countryWithFirstAstronaut --> 19 instances
http://dbpedia.org/ontology/countryWithFirstSatelliteLaunched --> 8 instances
http://dbpedia.org/ontology/managementCountry --> 0 instances
A. POSSIBLE DEFINITIONS

http://dbpedia.org/ontology/fastestDriverCountry --> 65 instances
http://dbpedia.org/ontology/mouthCountry --> 326 instances
http://dbpedia.org/ontology/secondDriverCountry --> 75 instances
http://dbpedia.org/ontology/locationCountry --> 1758 instances
http://dbpedia.org/ontology/thirdDriverCountry --> 77 instances
http://dbpedia.org/ontology/sourceCountry --> 298 instances
http://dbpedia.org/ontology/Country --> 1846 instances
http://dbpedia.org/ontology/countryWithFirstSpaceflight --> 7 instances
http://dbpedia.org/ontology/firstDriverCountry --> 75 instances
http://dbpedia.org/ontology/usingCountry --> 288 instances
http://dbpedia.org/ontology/capitalCountry --> 0 instances
http://dbpedia.org/ontology/sourceConfluenceCountry --> 0 instances

Suggest: Name:
http://dbpedia.org/ontology/birthName --> 2000 instances
http://dbpedia.org/ontology/WomensTennisAssociationTournament --> 48 instances
http://dbpedia.org/ontology/formerName --> 1450 instances
http://dbpedia.org/ontology/meshName --> 503 instances
http://dbpedia.org/ontology/cornishName --> 0 instances
http://dbpedia.org/ontology/manxName --> 0 instances
http://dbpedia.org/ontology/welshName --> 0 instances
http://dbpedia.org/ontology/iupacName --> 1999 instances
http://dbpedia.org/ontology/namedAfter --> 5 instances
http://dbpedia.org/ontology/teamName --> 648 instances
http://dbpedia.org/ontology/scottishName --> 0 instances
http://dbpedia.org/ontology/tournamentRecord --> 72 instances
http://dbpedia.org/ontology/personName --> 40 instances
http://dbpedia.org/ontology/irishName --> 0 instances
http://dbpedia.org/ontology/gaelicName --> 0 instances
http://dbpedia.org/ontology/leaderName --> 1920 instances

population:
http://dbpedia.org/ontology/populationMetroDensity --> 163 instances
http://dbpedia.org/ontology/populationTotal --> 1999 instances
http://dbpedia.org/ontology/populationAsOf --> 295 instances
http://dbpedia.org/ontology/populationUrbanDensity --> 76 instances
http://dbpedia.org/ontology/populationPlace --> 1940 instances
http://dbpedia.org/ontology/PopulatedPlace/populationUrbanDensity --> 77 instances
http://dbpedia.org/ontology/PopulationDensity --> 1990 instances
http://dbpedia.org/ontology/GeopoliticalOrganisation/populationDensity --> 0 instances
http://dbpedia.org/ontology/populationUrban --> 1713 instances
http://dbpedia.org/ontology/totalPopulation --> 1486 instances
http://dbpedia.org/ontology/PopulatedPlace/populationMetroDensity --> 165 instances
http://dbpedia.org/ontology/PopulatedPlace/populationDensity --> 2000 instances
http://dbpedia.org/ontology/populationMetro --> 956 instances

Synonyms: universe:
languages:
  http://dbpedia.org/ontology/language --> 1282 instances
  http://dbpedia.org/ontology/Language --> 1935 instances

isoNumeric:
  Suggest: Numeric:
  Suggest: is:
    http://dbpedia.org/ontology/episodeNumber --> 260 instances
    http://dbpedia.org/ontology/doctoralAdvisor --> 1895 instances
    http://dbpedia.org/ontology/TelevisionEpisode --> 1746 instances
    http://dbpedia.org/ontology/numberOfEpisode --> 697 instances
    http://dbpedia.org/ontology/garrison --> 1583 instances
    http://dbpedia.org/ontology/academicAdvisor --> 251 instances

geonamelD:
  http://dbpedia.org/ontology/id --> 859 instances
  http://dbpedia.org/ontology/area --> 1814 instances

bBoxSouth:

currencyCode:
  http://dbpedia.org/ontology/currency --> 480 instances
  http://dbpedia.org/ontology/code --> 190 instances
  http://dbpedia.org/ontology/Currency --> 312 instances
  Suggest: Code:
    http://dbpedia.org/ontology/code --> 190 instances
    http://dbpedia.org/ontology/inseeCode --> 0 instances
    http://dbpedia.org/ontology/schoolCode --> 239 instances
    http://dbpedia.org/ontology/oldcode --> 1 instances
  Suggest: currency:
    http://dbpedia.org/ontology/currency --> 480 instances
    http://dbpedia.org/ontology/Currency --> 312 instances

bBoxNorth:

--------------

--------------
A. POSSIBLE DEFINITIONS
Appendix B

Service1 Capabilities

Below we show the capabilities of a selected WFS service.

Listing B.1: Capabilities of service 1

```xml
<wfs:WFS_Capabilities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wfs" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:岱m=“http://www.opengeospatial.net/cite" xmlns:it.geosolutions=""
http://www.geo-solutions.it" xmlns:tiger="http://www.census.gov"
1.1.0/wfs.xsd" updateSequence="528">
  <ows:ServiceIdentification>
    <ows:Title>WHO WFS</ows:Title>
    <ows:Abstract>
      Welcome to the World Health Organization Web Feature server.
    </ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>WFS</ows:Keyword>
      <ows:Keyword>WMS</ows:Keyword>
      <ows:Keyword>GEOSERVER</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>WFS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>Restricted</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>World Health Organization</ows:ProviderName>
    <ows:ServiceContact>
      <ows:IndividualName>Public Health Mapping Group</ows:IndividualName>
    </ows:ServiceContact>
```

169
B. SERVICE1 CAPABILITIES

```xml
<ows:PositionName/>
<ows:ContactInfo>
  <ows:Phone>
    <ows:Voice/>
    <ows:Facsimile/>
  </ows:Phone>
  <ows:Address>
    <ows:City>Geneva</ows:City>
    <ows:AdministrativeArea/>
    <ows:PostalCode>CH-1211</ows:PostalCode>
    <ows:Country>Switzerland</ows:Country>
  </ows:Address>
</ows:ContactInfo>
</ows:ServiceProvider>
<ows:OperationsMetadata>
  <ows:Operation name="GetCapabilities">
    <ows:DCP>
      <ows:HTTP>
        <ows:Get xlink:href="http://apps.who.int/tools/geoserver/wfs?">
        </ows:Get>
        <ows:Post xlink:href="http://apps.who.int/tools/geoserver/wfs?">
        </ows:Post>
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
  <ows:Parameter name="AcceptVersions">
    <ows:Value>1.0.0</ows:Value>
    <ows:Value>1.1.0</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="AcceptFormats">
    <ows:Value>text/xml</ows:Value>
  </ows:Parameter>
  <ows:Operation name="DescribeFeatureType">
    <ows:DCP>
      <ows:HTTP>
        <ows:Get xlink:href="http://apps.who.int/tools/geoserver/wfs?">
        </ows:Get>
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
  <ows:Parameter name="outputFormat">
    <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  </ows:Parameter>
  <ows:Operation name="GetFeature">
    <ows:DCP>
      <ows:HTTP>
        <ows:Get xlink:href="http://apps.who.int/tools/geoserver/wfs?">
        </ows:Get>
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
  <ows:Parameter name="resultType">
    <ows:Value>results</ows:Value>
  </ows:Parameter>
</ows:OperationsMetadata>
```
<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  <ows:Value>GML2</ows:Value>
  <ows:Value>GML2-GZIP</ows:Value>
  <ows:Value>SHAPE-ZIP</ows:Value>
  <ows:Value>gml3</ows:Value>
  <ows:Value>json</ows:Value>
  <ows:Value>text/xml; subtype=gml/2.1.2</ows:Value>
</ows:Parameter>

<ows:Parameter name="LocalTraverseXLinkScope">
  <ows:Value>2</ows:Value>
</ows:Parameter>

<ows:Parameter name="releaseAction">
  <ows:Value>ALL</ows:Value>
  <ows:Value>SOME</ows:Value>
</ows:Parameter>

<ows:Parameter name="resultType">
  <ows:Value>results</ows:Value>
  <ows:Value>hits</ows:Value>
</ows:Parameter>

<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  <ows:Value>GML2</ows:Value>
  <ows:Value>GML2-GZIP</ows:Value>
  <ows:Value>SHAPE-ZIP</ows:Value>
  <ows:Value>gml3</ows:Value>
  <ows:Value>json</ows:Value>
</ows:Parameter>
B. SERVICE1 CAPABILITIES

136  <ows:Value>text/xml; subtype=gml/2.1.2</ows:Value>
137  </ows:Parameter>
138  </ows:Operation>
139  <ows:Operation name="Transaction">
140      <ows:DCP>
141          <ows:HTTP>
142              <ows:Get xlink:href="http://apps.who.int/tools/geoserver/wfs?"/>
143              <ows:Post xlink:href="http://apps.who.int/tools/geoserver/wfs?"/>
144          </ows:HTTP>
145      </ows:DCP>
146  </ows:Parameter name="inputFormat">
147      <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
148  </ows:Parameter>
149  <ows:Parameter name="idgen">
150      <ows:Value>GenerateNew</ows:Value>
151  </ows:Parameter>
152  <ows:Parameter name="releaseAction">
153      <ows:Value>ALL</ows:Value>
154      <ows:Value>SOME</ows:Value>
155  </ows:Parameter>
156  </ows:Operation>
157  </ows:OperationsMetadata>
158  <FeatureTypeList>
159      <Operations>
160          <Operation>Query</Operation>
161          <Operation>Insert</Operation>
162          <Operation>Update</Operation>
163          <Operation>Delete</Operation>
164          <Operation>Lock</Operation>
165      </Operations>
166      <FeatureType xmlns:WHO="http://www.who.int/">
167          <Name>WHO:efs2008_countrydata</Name>
168          <Title>EpiFact Sheets 2008 Data</Title>
169          <Abstract>Generated from WHOGeodata</Abstract>
170          <ows:Keywords>
171              <ows:Keyword>efs_data_for_websevice</ows:Keyword>
172              <ows:Keyword>WHOGeodata</ows:Keyword>
173          </ows:Keywords>
175          <ows:WGS84BoundingBox>
176              <ows:LowerCorner>-527.400749390449 -545.3687536171334</ows:LowerCorner>
177              <ows:UpperCorner>586.4983522346869 568.5303480080025</ows:UpperCorner>
178          </ows:WGS84BoundingBox>
179      </FeatureType>
180      <FeatureType xmlns:WHO="http://www.who.int/">
181          <Name>WHO:GIM_MONITORING_GEOMETRY_MEASURES</Name>
182          <Title>GIM_MONITORING_GEOMETRY_MEASURES_Type</Title>
183          <Abstract>Generated from FLUID</Abstract>
184  </FeatureType>
<ows:Keywords>
  <ows:Keyword>FLUID</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-179.99890679219 -55.373263251012</ows:LowerCorner>
  <ows:UpperCorner>179.99841379129 83.077613045033</ows:UpperCorner>
</ows:WGS84BoundingBox>

<ows:Keywords>
  <ows:Keyword>GIM</ows:Keyword>
  <ows:Keyword>MONITORING</ows:Keyword>
  <ows:Keyword>GEOMETRY</ows:Keyword>
  <ows:Keyword>MEASURES</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-183.5988799980248 -169.7464586005643</ows:LowerCorner>
  <ows:UpperCorner>183.5983869971248 197.45080839458532</ows:UpperCorner>
</ows:WGS84BoundingBox>

<ows:Keywords>
  <ows:Keyword>GIM</ows:Keyword>
  <ows:Keyword>MONITORING</ows:Keyword>
  <ows:Keyword>GEOMETRY</ows:Keyword>
  <ows:Keyword>MEASURES</ows:Keyword>
  <ows:Keyword>FLUID</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-183.5988799980248 -169.7464586005643</ows:LowerCorner>
  <ows:UpperCorner>183.5983869971248 197.45080839458532</ows:UpperCorner>
</ows:WGS84BoundingBox>

<ows:Keywords>
  <ows:Keyword>GIM</ows:Keyword>
  <ows:Keyword>MONITORING</ows:Keyword>
  <ows:Keyword>GEOMETRY</ows:Keyword>
  <ows:Keyword>MEASURES</ows:Keyword>
  <ows:Keyword>FLUID</ows:Keyword>
  <ows:Keyword>TEST</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-183.5988799980248 -169.7464586005643</ows:LowerCorner>
  <ows:UpperCorner>183.5983869971248 197.45080839458532</ows:UpperCorner>
</ows:WGS84BoundingBox>

<ows:Keywords>
  <ows:Keyword>GIM</ows:Keyword>
  <ows:Keyword>MONITORING</ows:Keyword>
  <ows:Keyword>GEOMETRY</ows:Keyword>
  <ows:Keyword>MEASURES</ows:Keyword>
  <ows:Keyword>FLUID</ows:Keyword>
  <ows:Keyword>TRAINING</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-183.5988799980248 -169.7464586005643</ows:LowerCorner>
  <ows:UpperCorner>183.5983869971248 197.45080839458532</ows:UpperCorner>
</ows:WGS84BoundingBox>
B. SERVICE1 CAPABILITIES

<ows:Keyword>FLUID TRAINING</ows:Keyword>
</ows:Keywords>
<DefaultSRS>urn:o:ogc:Def:crs:EPSG:4326</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-179.99890679219 -55.373263251012</ows:LowerCorner>
<ows:UpperCorner>179.99841379129 83.077613045033</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

Global Health Facilities (WHO) provided by the World Health Organization
</Abstract>
<ows:Keywords>
<ows:Keyword>(WHO)</ows:Keyword>
<ows:Keyword>Facilities</ows:Keyword>
<ows:Keyword>Health</ows:Keyword>
<ows:Keyword>Global</ows:Keyword>
</ows:Keywords>
<DefaultSRS>urn:o:ogc:Def:crs:EPSG:4326</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-174.2479513125514 -144.65422801543264</ows:LowerCorner>
<ows:UpperCorner>141.442462761264 171.03618602324514</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
<Name>WHO:leishecomp_resevoirs</Name>
<Title>Leishmaniasis Reservoirs</Title>
<Abstract>Generated from ems</Abstract>
<ows:Keywords>
<ows:Keyword>leishecomp_resevoirs</ows:Keyword>
<ows:Keyword>ems</ows:Keyword>
</ows:Keywords>
<DefaultSRS>urn:o:ogc:Def:crs:EPSG:4326</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-10.078279195279562 11.342904413422701</ows:LowerCorner>
<ows:UpperCorner>40.89957544985783 62.32075905856009</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
<Name>WHO:mm_admin0</Name>
<Title>Myanmar (Admin 0)</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>mm_admin0</ows:Keyword>
</ows:Keywords>
<DefaultSRS>urn:o:ogc:Def:crs:EPSG:4326</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-10.078279195279562 11.342904413422701</ows:LowerCorner>
<ows:UpperCorner>40.89957544985783 62.32075905856009</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
<ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
</ows:WGS84BoundingBox>
</FeatureType>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:mm_admin1</Name>
>Title>Myanmar (Admin 1)</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>mm_admin1</ows:Keyword>
<ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>87.01955965995819 9.401933946609503</ows:LowerCorner>
<ows:UpperCorner>106.341318330765 28.72369261741634</ows:UpperCorner>
</ows:WGS4BoundingBox>
</FeatureType>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:mm_admin2</Name>
<Title>Myanmar (Admin 2)</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>mm_admin2</ows:Keyword>
<ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>92.1866836547852 9.59136295318604</ows:LowerCorner>
<ows:UpperCorner>101.174194335938 28.5342636108398</ows:UpperCorner>
</ows:WGS4BoundingBox>
</FeatureType>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:myanmar_health_facilities</Name>
<Title>Myanmar Health Facilities (WHO)</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>WHOGeodata</ows:Keyword>
<ows:Keyword>myanmar_health_facilities</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
B. SERVICE1 CAPABILITIES

<FeatureType xmlns:WHO="http://www.who.int/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/feature/featureType http://www.opengis.net/ref-featureType.xsd"
  xmlns:ows="http://www.opengis.net/ow/sfs/1.0/ows"
  xmlns:wgs="http://www.opengis.net/xes/gml/3.2.1/wgs84BoundingBox/ows"
  xmlns:json="http://www.json.org/jsonld/1.0/">
  <ows:Title>Myanmar Information Management Unit Township Boundaries</ows:Title>
  <ows:Abstract>Generated from mm_geodata</ows:Abstract>
  <ows:Keywords><ows:Keyword>mmr_polbnda_adm3_mimu_polygon</ows:Keyword></ows:Keywords>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/feature/featureType http://www.opengis.net/ref-featureType.xsd"
  xmlns:ows="http://www.opengis.net/ow/sfs/1.0/ows"
  xmlns:wgs="http://www.opengis.net/xes/gml/3.2.1/wgs84BoundingBox/ows"
  xmlns:json="http://www.json.org/jsonld/1.0/">
  <ows:Title>Myanmar Place Names (NGA)</ows:Title>
  <ows:Abstract>Generated from WHOGeodata</ows:Abstract>
  <ows:Keywords><ows:Keyword>mmnga_placenames</ows:Keyword></ows:Keywords>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/feature/featureType http://www.opengis.net/ref-featureType.xsd"
  xmlns:ows="http://www.opengis.net/ow/sfs/1.0/ows"
  xmlns:wgs="http://www.opengis.net/xes/gml/3.2.1/wgs84BoundingBox/ows"
  xmlns:json="http://www.json.org/jsonld/1.0/">
  <ows:Title>OutbreakReported Type</ows:Title>
  <ows:Abstract>Generated from WHOGeodata</ows:Abstract>
  <ows:Keywords><ows:Keyword>OutbreakReported</ows:Keyword></ows:Keywords>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:mmr_unjlc_roads</Name>
>Title>UNJLC/UNOSAT/GISCorps Roads</Title>
<Abstract>Generated from mm_geodata</Abstract>
<Keywords>
  <Keyword>Roads_polyline</Keyword>
  <Keyword>mm_geodata</Keyword>
</Keywords>
<WGS84BoundingBox>
  <LowerCorner>94.20928409489927 15.768735643429952</LowerCorner>
  <UpperCorner>96.81582522473333 17.66406966356742</UpperCorner>
</WGS84BoundingBox>

<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:mm_who_populatedplaces</Name>
>Title>WHO Populated Places (Myanmar)</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<Keywords>
  <Keyword>mm_who_populatedplaces</Keyword>
  <Keyword>WHOGeodata</Keyword>
</Keywords>
<WGS84BoundingBox>
  <LowerCorner>92.2458038330078 9.97859954833984</LowerCorner>
  <UpperCorner>100.826187133789 27.9181041717529</UpperCorner>
</WGS84BoundingBox>

<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:world_influenza_humanhealthlab</Name>
>Title>World_InfluenzaHumanHealthLab_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<Keywords>
  <Keyword>World_InfluenzaHumanHealthLab</Keyword>
  <Keyword>WHOGeodata</Keyword>
</Keywords>
<WGS84BoundingBox>
  <LowerCorner>-119.0 -41.2833023071289</LowerCorner>
  <UpperCorner>178.447723388672 64.1350021362305</UpperCorner>
</WGS84BoundingBox>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:admin1_world</Name>
<Title>admin1_world_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
  <ows:Keyword>admin1_world</ows:Keyword>
  <ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-185.65986976318385 -180.386739346318385</ows:LowerCorner>
  <ows:UpperCorner>197.78639564208984 203.0595260589601</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:admin2_world</Name>
<Title>admin2_world_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
  <ows:Keyword>admin2_world</ows:Keyword>
  <ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-185.65986976318385 -180.386739346318385</ows:LowerCorner>
  <ows:UpperCorner>197.78639564208984 203.0595260589601</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:admin3_world</Name>
<Title>admin3_world_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>admin3_world</ows:Keyword>
<ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-82.9223022460938 -32.107723236084</ows:LowerCorner>
<ows:UpperCorner>139.170623779297 61.0971527099609</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:admin_world</Name>
<Title>admin_world_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
<ows:Keywords>
<ows:Keyword>admin_world</ows:Keyword>
<ows:Keyword>WHOGeodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-86.73649474639893 -98.66524287719723</ows:LowerCorner>
<ows:UpperCorner>135.7907473709105 123.86199924011223</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/"
<Name>WHO:africa_airports</Name>
<Title>africa_airports_Type</Title>
<Abstract>Generated from africa_airports</Abstract>
<ows:Keywords>
<ows:Keyword>africa_airports</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-178.17236328125 -54.7804794311523</ows:LowerCorner>
<ows:UpperCorner>187.30451965332 74.9679412841797</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
B. SERVICE1 CAPABILITIES

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:countries_indide_points</Name>
  <Title>countries_indide_points_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>countries_indide_points</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>-181.70124228582557 -168.23517248857726</ows:LowerCorner>
    <ows:UpperCorner>182.79214346281333 196.25821326006164</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:efs2008_sentinelsites</Name>
  <Title>efs2008_sentinelsites_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>efs2008_sentinelsites</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>-178.8167530822756 -163.63969009399426</ows:LowerCorner>
    <ows:UpperCorner>192.64175003051758 207.81881301879895</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:ht_geonames</Name>
  <Title>ht_geonames_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>ht_geonames</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>-89.19757591552734 -67.57188402481081</ows:LowerCorner>
    <ows:UpperCorner>75.49341331787114 97.11910520858767</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>
B. SERVICE1 CAPABILITIES

<ows:Keyword>ems</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-10.577716025010535 9.332362964232784</ows:LowerCorner>
<ows:UpperCorner>40.887046105311526 60.79712509455484</ows:UpperCorner>
</ows:WGS84BoundingBox>
<FeatureType xmlns:WHO="http://www.who.int/">
<Name>WHO:disputed_borders_white</Name>
<Title>maskline08_detailed</Title>
<Abstract>Generated from world_disputed_borders</Abstract>
<ows:Keywords>
<ows:Keyword>maskline08_detailed</ows:Keyword>
<ows:Keyword>world_disputed_borders</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-10.577716025010535 9.332362964232784</ows:LowerCorner>
<ows:UpperCorner>40.887046105311526 60.79712509455484</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
<FeatureType xmlns:WHO="http://www.who.int/">
<Name>WHO:maskpoly08_detailed</Name>
<Title>maskpoly08_detailed</Title>
<Abstract>Generated from world_disputed_areas</Abstract>
<ows:Keywords>
<ows:Keyword>maskpoly08_detailed</ows:Keyword>
<ows:Keyword>world_disputed_areas</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-10.577716025010535 9.332362964232784</ows:LowerCorner>
<ows:UpperCorner>40.887046105311526 60.79712509455484</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
<ows:LowerCorner>-102.98214280197605 -104.02345893298904</ows:LowerCorner>
<ows:UpperCorner>137.46933870036463 136.42802256935164</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:member_states</Name>
  <Title>member_states_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>member_states</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>-180.0 -55.9197273254395</ows:LowerCorner>
    <ows:UpperCorner>180.0 83.1138839721686</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:member_states_label</Name>
  <Title>member_states_label_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>member_states_label</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>-175.254196166992 -43.588191986084</ows:LowerCorner>
    <ows:UpperCorner>179.215225219727 65.2263412475586</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/">
  <Name>WHO:mm_unosat_flood_extent</Name>
  <Title>mm_unosat_flood_extent_Type</Title>
  <Abstract>Generated from WHOGeodata</Abstract>
  <ows:Keywords>
    <ows:Keyword>mm_unosat_flood_extent</ows:Keyword>
    <ows:Keyword>WHOGeodata</ows:Keyword>
  </ows:Keywords>
  <ows:WGS84BoundingBox>
    <ows:LowerCorner>93.67797976403166 14.687930276047855</ows:LowerCorner>
    <ows:UpperCorner>98.22127666768915 19.231227179705353</ows:UpperCorner>
  </ows:WGS84BoundingBox>
</FeatureType>
B. SERVICE1 CAPABILITIES

<FeatureType xmlns:WHO="http://www.who.int/"

<Name>WHO:mmr_damaged_bridges</Name>
<Title>mmr_damaged_bridges_Type</Title>
<Abstract>Generated from mm_geodata</Abstract>
<ows:Keywords>
  <ows:Keyword>mmr_damaged_bridges</ows:Keyword>
  <ows:Keyword>mm_geodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>94.19836776733402 15.408198051452665</ows:LowerCorner>
  <ows:UpperCorner>96.21198684692378 17.421817131042438</ows:UpperCorner>
</ows:WGS84BoundingBox>

</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"

<Name>WHO:mmr_rdsl2_1m_vmap1_polyline</Name>
<Title>mmr_rdsl2_1m_vmap1_polyline_Type</Title>
<Abstract>Generated from mm_geodata</Abstract>
<ows:Keywords>
  <ows:Keyword>mmr_rdsl2_1m_vmap1_polyline</ows:Keyword>
  <ows:Keyword>mm_geodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>94.42511634826663 11.213391113281334</ows:LowerCorner>
  <ows:UpperCorner>101.78854484558087 18.57681961059557</ows:UpperCorner>
</ows:WGS84BoundingBox>

</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"

<Name>WHO:mmr_rdsl_1m_vmap1_polyline</Name>
<Title>mmr_rdsl_1m_vmap1_polyline_Type</Title>
<Abstract>Generated from mm_geodata</Abstract>
<ows:Keywords>
  <ows:Keyword>mmr_rdsl_1m_vmap1_polyline</ows:Keyword>
  <ows:Keyword>mm_geodata</ows:Keyword>
</ows:Keywords>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>94.4891815185547 9.97994518280029</ows:LowerCorner>
  <ows:UpperCorner>101.408134460449 18.00000019073486</ows:UpperCorner>
</ows:WGS84BoundingBox>

</FeatureType>

<FeatureType xmlns:WHO="http://www.who.int/"

<Name>WHO:western_sahara</Name>
<Title>western_sahara_Type</Title>
<Abstract>Generated from WHOGeodata</Abstract>
B. SERVICE1 CAPABILITIES

<ows:LowerCorner>-178.89810597839318 -169.92774236144982
</ows:LowerCorner>
<ows:UpperCorner>185.51283681335417 194.48320043029753
</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
</FeatureTypeList>
<ogc:Filter_Capabilities>
<ogc:spatial_Capabilities>
<ogc:GeometryOperands>
<ogc:GeometryOperand><gml:Envelope>
</ogc:GeometryOperand>
<ogc:GeometryOperand><gml:Point>
</ogc:GeometryOperand>
<ogc:GeometryOperand><gml:LineString>
</ogc:GeometryOperand>
<ogc:GeometryOperand><gml:Polygon>
</ogc:GeometryOperand>
</ogc:GeometryOperands>
<ogc:spatial_Operators>
<ogc:spatial_Operator name="Disjoint"/>
<ogc:spatial_Operator name="Equals"/>
<ogc:spatial_Operator name="DWithin"/>
<ogc:spatial_Operator name="Beyond"/>
<ogc:spatial_Operator name="Intersects"/>
<ogc:spatial_Operator name="Touches"/>
<ogc:spatial_Operator name="Crosses"/>
<ogc:spatial_Operator name="Contains"/>
<ogc:spatial_Operator name="Overlaps"/>
<ogc:spatial_Operator name="BBOX"/>
</ogc:spatial_Operators>
</ogc:spatial_Capabilities>
<ogc:scalar_Capabilities>
<ogc:LogicalOperators/>
<ogc:ComparisonOperators>
<ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
<ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
<ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
<ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
<ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
</ogc:ComparisonOperators>
<ogc:ArithmeticOperators>
<ogc:SimpleArithmetic/>
<ogc:Functions>
<ogc:FunctionNames>
<ogc:FunctionName nArg="1">abs</ogc:FunctionName>
<ogc:FunctionName nArg="1">abs_2</ogc:FunctionName>
<ogc:FunctionName nArg="1">abs_3</ogc:FunctionName>
<ogc:FunctionName nArg="1">abs_4</ogc:FunctionName>
<ogc:FunctionName nArg="1">acos</ogc:FunctionName>
<ogc:FunctionName nArg="1">Area</ogc:FunctionName>
<ogc:FunctionName nArg="1">asin</ogc:FunctionName>
### B. SERVICE1 CAPABILITIES

<table>
<thead>
<tr>
<th>Function Name</th>
<th>nArgs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lessThan</td>
<td>2</td>
<td>less than</td>
</tr>
<tr>
<td>greaterThan</td>
<td>2</td>
<td>greater than</td>
</tr>
<tr>
<td>equalTo</td>
<td>2</td>
<td>equal to</td>
</tr>
<tr>
<td>lessEqualThan</td>
<td>2</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>length</td>
<td>1</td>
<td>object length</td>
</tr>
<tr>
<td>isEmpty</td>
<td>1</td>
<td>empty</td>
</tr>
<tr>
<td>isNull</td>
<td>1</td>
<td>null</td>
</tr>
<tr>
<td>isRing</td>
<td>1</td>
<td>ring</td>
</tr>
<tr>
<td>isSimple</td>
<td>1</td>
<td>simple</td>
</tr>
<tr>
<td>isValid</td>
<td>1</td>
<td>valid</td>
</tr>
<tr>
<td>isWithinDistance</td>
<td>2</td>
<td>is within distance</td>
</tr>
<tr>
<td>intersect</td>
<td>2</td>
<td>intersect</td>
</tr>
<tr>
<td>intersects</td>
<td>2</td>
<td>intersects</td>
</tr>
<tr>
<td>isClosed</td>
<td>1</td>
<td>closed</td>
</tr>
<tr>
<td>isOverlapping</td>
<td>2</td>
<td>overlapping</td>
</tr>
<tr>
<td>isPartOf</td>
<td>2</td>
<td>part of</td>
</tr>
<tr>
<td>isSubOf</td>
<td>2</td>
<td>sub of</td>
</tr>
<tr>
<td>isSuperset</td>
<td>2</td>
<td>superset</td>
</tr>
<tr>
<td>isSubset</td>
<td>2</td>
<td>subset</td>
</tr>
<tr>
<td>isSupersetOf</td>
<td>2</td>
<td>superset of</td>
</tr>
<tr>
<td>isSubsetOf</td>
<td>2</td>
<td>subset of</td>
</tr>
<tr>
<td>isSimpleWithin</td>
<td>2</td>
<td>simple within</td>
</tr>
<tr>
<td>isWithin</td>
<td>2</td>
<td>within</td>
</tr>
<tr>
<td>numGeometries</td>
<td>1</td>
<td>number of geometries</td>
</tr>
<tr>
<td>numInteriorRing</td>
<td>1</td>
<td>number of interior rings</td>
</tr>
<tr>
<td>numPoints</td>
<td>1</td>
<td>number of points</td>
</tr>
<tr>
<td>overlaps</td>
<td>2</td>
<td>overlaps</td>
</tr>
<tr>
<td>propertyExists</td>
<td>1</td>
<td>property exists</td>
</tr>
<tr>
<td>parseBoolean</td>
<td>1</td>
<td>parse boolean</td>
</tr>
<tr>
<td>parseDouble</td>
<td>1</td>
<td>parse double</td>
</tr>
<tr>
<td>parseInt</td>
<td>1</td>
<td>parse int</td>
</tr>
<tr>
<td>pi</td>
<td>0</td>
<td>pi</td>
</tr>
<tr>
<td>pointN</td>
<td>1</td>
<td>point N</td>
</tr>
<tr>
<td>pow</td>
<td>2</td>
<td>power</td>
</tr>
<tr>
<td>propertyExists</td>
<td>1</td>
<td>property exists</td>
</tr>
<tr>
<td>Quotient</td>
<td>2</td>
<td>quotient</td>
</tr>
<tr>
<td>random</td>
<td>0</td>
<td>random</td>
</tr>
<tr>
<td>relate</td>
<td>2</td>
<td>relate</td>
</tr>
<tr>
<td>relatePattern</td>
<td>3</td>
<td>relate pattern</td>
</tr>
</tbody>
</table>
<ogc:FunctionName nArgs="1">rint</ogc:FunctionName>
<ogc:FunctionName nArgs="1">round</ogc:FunctionName>
<ogc:FunctionName nArgs="1">round_2</ogc:FunctionName>
<ogc:FunctionName nArgs="1">roundDouble</ogc:FunctionName>
<ogc:FunctionName nArgs="1">sin</ogc:FunctionName>
<ogc:FunctionName nArgs="1">sqrt</ogc:FunctionName>
<ogc:FunctionName nArgs="2">StandardDeviation</ogc:FunctionName>
<ogc:FunctionName nArgs="1">startPoint</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strConcat</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strEndsWith</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strEqualsIgnoreCase</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strIndexOf</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strLastIndexOf</ogc:FunctionName>
<ogc:FunctionName nArgs="1">strLength</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strMatches</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strReplace</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strStartsWith</ogc:FunctionName>
<ogc:FunctionName nArgs="3">strSubstring</ogc:FunctionName>
<ogc:FunctionName nArgs="2">strSubstringStart</ogc:FunctionName>
<ogc:FunctionName nArgs="1">strToLowerCase</ogc:FunctionName>
<ogc:FunctionName nArgs="1">strToUpperCase</ogc:FunctionName>
<ogc:FunctionName nArgs="1">strTrim</ogc:FunctionName>
<ogc:FunctionName nArgs="2">symDifference</ogc:FunctionName>
<ogc:FunctionName nArgs="1">tan</ogc:FunctionName>
<ogc:FunctionName nArgs="1">toDegrees</ogc:FunctionName>
<ogc:FunctionName nArgs="1">toRadians</ogc:FunctionName>
<ogc:FunctionName nArgs="2">touches</ogc:FunctionName>
<ogc:FunctionName nArgs="1">toWKT</ogc:FunctionName>
<ogc:FunctionName nArgs="2">union</ogc:FunctionName>
<ogc:FunctionName nArgs="2">UniqueInterval</ogc:FunctionName>
<ogc:FunctionName nArgs="2">within</ogc:FunctionName>
</ogc:FunctionNames>
</ogc:Functions>
</ogc:ArithmeticOperators>
</ogc:ScalarCapabilities>
<ogc:IdCapabilities>
<ogc:FID/>
<ogc:EID/>
</ogc:IdCapabilities>
</ogc:FilterCapabilities>
</wfs:WFS Capabilities>
Appendix C

Service1 Syntactic description

Below we show the intermediate results of the syntactic description of a WFS service.

Listing C.1: Syntactic of WFS service 1

```plaintext
Source {idSource=0, name=null, description=null}
Method{idMethod=0, name=WHO:admin3_world}
  Output{idOutput=0, name=uuid, dataType=null, value=null}
  Output{idOutput=0, name=length, dataType=null, value=null}
  Output{idOutput=0, name=isoctry, dataType=null, value=null}
  Output{idOutput=0, name=geom, dataType=null, value=null}
  Output{idOutput=0, name=goid, dataType=null, value=null}
  Output{idOutput=0, name=lvid, dataType=null, value=null}
  Output{idOutput=0, name=isos, dataType=null, value=null}
  Output{idOutput=0, name=med, dataType=null, value=null}
  Output{idOutput=0, name=acc, dataType=null, value=null}
  Output{idOutput=0, name=med, dataType=null, value=null}
  Output{idOutput=0, name=loc, dataType=null, value=null}
  Output{idOutput=0, name=tileid, dataType=null, value=null}
  Output{idOutput=0, name=loc, dataType=null, value=null}
  Output{idOutput=0, name=tileid, dataType=null, value=null}
  Output{idOutput=0, name=loc, dataType=null, value=null}
  Output{idOutput=0, name=tileid, dataType=null, value=null}
  Output{idOutput=0, name=loc, dataType=null, value=null}
  Output{idOutput=0, name=tileid, dataType=null, value=null}
  Output{idOutput=0, name=loc, dataType=null, value=null}
  Output{idOutput=0, name=tileid, dataType=null, value=null}
```
C. SERVICE1 SYNTACTIC DESCRIPTION

```
Output{idOutput=0,name=geodb_oid ,dataType=null , value=null}
Output{idOutput=0,name=id , dataType=null , value=null}
Output{idOutput=0,name=wtc_desc ,dataType=null , value=null}
Output{idOutput=0,name=nam , dataType=null , value=null}
Output{idOutput=0,name=use_desc ,dataType=null , value=null}
Output{idOutput=0,name=ltn_desc ,dataType=null , value=null}
Output{idOutput=0,name=edg_id ,dataType=null , value=null}
Output{idOutput=0,name=acc_desc ,dataType=null , value=null}
Output{idOutput=0,name=shape_length ,dataType=null , value=null}

Method{idMethod=0,name=WHO:YangonRoads_polyline}
Output{idOutput=0,name=objectid ,dataType=null , value=null}
Output{idOutput=0,name=geom , dataType=null , value=null}
Output{idOutput=0,name=kmilname ,dataType=null , value=null}
Output{idOutput=0,name=geodb_oid ,dataType=null , value=null}
Output{idOutput=0,name=shape_length ,dataType=null , value=null}

Method{idMethod=0,name=WHO:maskpoly08_detailed}
Output{idOutput=0,name=PERIMETER ,dataType=null , value=null}
Output{idOutput=0,name=NAME , dataType=null , value=null}
Output{idOutput=0,name=HECTARES ,dataType=null , value=null}
Output{idOutput=0,name=ID ,dataType=null , value=null}
Output{idOutput=0,name=AREA ,dataType=null , value=null}

Method{idMethod=0,name=WHO:mmr_damaged_bridges}
Output{idOutput=0,name=long ,dataType=null , value=null}
Output{idOutput=0,name=geom ,dataType=null , value=null}
Output{idOutput=0,name=status ,dataType=null , value=null}
Output{idOutput=0,name=lat ,dataType=null , value=null}

Method{idMethod=0,name=WHO:maskline08_detailed}
Output{idOutput=0,name=NAME ,dataType=null , value=null}
Output{idOutput=0,name=the_geom ,dataType=null , value=null}

Method{idMethod=0,name=WHO:world_who_offices}
Output{idOutput=0,name=country ,dataType=null , value=null}
Output{idOutput=0,name=lat ,dataType=null , value=null}
Output{idOutput=0,name=institut_4 ,dataType=null , value=null}
Output{idOutput=0,name=long ,dataType=null , value=null}
Output{idOutput=0,name=city ,dataType=null , value=null}
Output{idOutput=0,name=institut_3 ,dataType=null , value=null}
Output{idOutput=0,name=institut_1 ,dataType=null , value=null}
Output{idOutput=0,name=classifica ,dataType=null , value=null}
Output{idOutput=0,name=institut_2 ,dataType=null , value=null}
Output{idOutput=0,name=sn ,dataType=null , value=null}
Output{idOutput=0,name=geom ,dataType=null , value=null}
Output{idOutput=0,name=institutio ,dataType=null , value=null}
Output{idOutput=0,name=countrycod ,dataType=null , value=null}
Output{idOutput=0,name=institut_6 ,dataType=null , value=null}
Output{idOutput=0,name=region ,dataType=null , value=null}
Output{idOutput=0,name=referencei ,dataType=null , value=null}
Output{idOutput=0,name=institut_5 ,dataType=null , value=null}
Output{idOutput=0,name=classificat ,dataType=null , value=null}
Output{idOutput=0,name=function ,dataType=null , value=null}
```
Method {idMethod=0, name=WHO:admin2_world}
  Output {idOutput=0, name=salb, name, dataType=null, value=null}
  Output {idOutput=0, name=area, dataType=null, value=null}
  Output {idOutput=0, name=iso3, dataType=null, value=null}
  Output {idOutput=0, name=cntry_err, dataType=null, value=null}
  Output {idOutput=0, name=gaul_name, dataType=null, value=null}
  Output {idOutput=0, name=lv1id, dataType=null, value=null}
  Output {idOutput=0, name=geom, dataType=null, value=null}
  Output {idOutput=0, name=who_region, dataType=null, value=null}
  Output {idOutput=0, name=unocode, dataType=null, value=null}
  Output {idOutput=0, name=continent, dataType=null, value=null}
  Output {idOutput=0, name=gaul_code, dataType=null, value=null}
  Output {idOutput=0, name=sovereign, dataType=null, value=null}
  Output {idOutput=0, name=unlocode, dataType=null, value=null}
  Output {idOutput=0, name=who_code, dataType=null, value=null}
  Output {idOutput=0, name=continent, dataType=null, value=null}
  Output {idOutput=0, name=fips_code, dataType=null, value=null}
  Output {idOutput=0, name=uuid, dataType=null, value=null}
  Output {idOutput=0, name=lv1, dataType=null, value=null}
  Output {idOutput=0, name=un_code, dataType=null, value=null}
  Output {idOutput=0, name=goid, dataType=null, value=null}
  Output {idOutput=0, name=gaul_code, dataType=null, value=null}
  Output {idOutput=0, name=who_status, dataType=null, value=null}
  Output {idOutput=0, name=iso2, dataType=null, value=null}
  Output {idOutput=0, name=length, dataType=null, value=null}
  Output {idOutput=0, name=salb_code, dataType=null, value=null}
Method {idMethod=0, name=WHO:yellow_fever_outbreaks}
  Output {idOutput=0, name=gisglobal, dataType=null, value=null}
  Output {idOutput=0, name=admin6, dataType=null, value=null}
  Output {idOutput=0, name=object_id, dataType=null, value=null}
  Output {idOutput=0, name=lv1id_1, dataType=null, value=null}
  Output {idOutput=0, name=numberofou, dataType=null, value=null}
  Output {idOutput=0, name=lv1, dataType=null, value=null}
  Output {idOutput=0, name=admin5, dataType=null, value=null}
  Output {idOutput=0, name=admin2, dataType=null, value=null}
  Output {idOutput=0, name=id, dataType=null, value=null}
  Output {idOutput=0, name=area, dataType=null, value=null}
  Output {idOutput=0, name=datetimecr, dataType=null, value=null}
  Output {idOutput=0, name=link, dataType=null, value=null}
  Output {idOutput=0, name=country, dataType=null, value=null}
  Output {idOutput=0, name=hier_type, dataType=null, value=null}
  Output {idOutput=0, name=hier_id, dataType=null, value=null}
  Output {idOutput=0, name=createdbyu, dataType=null, value=null}
  Output {idOutput=0, name=admin3, dataType=null, value=null}
  Output {idOutput=0, name=iso_ctry, dataType=null, value=null}
  Output {idOutput=0, name=lv1id, dataType=null, value=null}
  Output {idOutput=0, name=admin4, dataType=null, value=null}
  Output {idOutput=0, name=extcode, dataType=null, value=null}
  Output {idOutput=0, name=admin1_l, dataType=null, value=null}
  Output {idOutput=0, name=geom, dataType=null, value=null}
  Output {idOutput=0, name=admin1, dataType=null, value=null}
Method {idMethod=0, name=WHO:admin1_world}
  Output {idOutput=0, name=gaul_code, dataType=null, value=null}
  Output {idOutput=0, name=length, dataType=null, value=null}
  Output {idOutput=0, name=who_code, dataType=null, value=null}
C. SERVICE1 SYNTACTIC DESCRIPTION

Output{idOutput=0,name=un_code ,dataType=null ,value=null}
Output{idOutput=0,name=uuid ,dataType=null ,value=null}
Output{idOutput=0,name=sovereign ,dataType=null ,value=null}
Output{idOutput=0,name=fips_code ,dataType=null ,value=null}
Output{idOutput=0,name=goid ,dataType=null ,value=null}
Output{idOutput=0,name=gaul_name ,dataType=null ,value=null}
Output{idOutput=0,name=iso3 ,dataType=null ,value=null}
Output{idOutput=0,name=geom ,dataType=null ,value=null}
Output{idOutput=0,name=lvli_d ,dataType=null ,value=null}
Output{idOutput=0,name=salb_name ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=unlocode ,dataType=null ,value=null}
Output{idOutput=0,name=un_code ,dataType=null ,value=null}
Output{idOutput=0,name=shape_len ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}
Output{idOutput=0,name=objectid ,dataType=null ,value=null}
Output{idOutput=0,name=sovereign ,dataType=null ,value=null}
Output{idOutput=0,name=who_code ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=lvli_d ,dataType=null ,value=null}
Output{idOutput=0,name=shape_area ,dataType=null ,value=null}
Output{idOutput=0,name=iso2_code ,dataType=null ,value=null}
Output{idOutput=0,name=iso2 ,dataType=null ,value=null}
Output{idOutput=0,name=iso3_code ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=lvli_d ,dataType=null ,value=null}

Method{idMethod=0,name=WHO: western_sahara}
Output{idOutput=0,name=entry_terr ,dataType=null ,value=null}
Output{idOutput=0,name=shape_len ,dataType=null ,value=null}
Output{idOutput=0,name=iso3_code ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}
Output{idOutput=0,name=objectid ,dataType=null ,value=null}
Output{idOutput=0,name=sovereign ,dataType=null ,value=null}
Output{idOutput=0,name=who_code ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=lvli_d ,dataType=null ,value=null}
Output{idOutput=0,name=shape_area ,dataType=null ,value=null}
Output{idOutput=0,name=iso2_code ,dataType=null ,value=null}
Output{idOutput=0,name=iso2 ,dataType=null ,value=null}

Method{idMethod=0,name=WHO: v_unit_h}
Output{idOutput=0,name=longitude ,dataType=null ,value=null}
Output{idOutput=0,name=iso_ctry ,dataType=null ,value=null}
Output{idOutput=0,name=unittypename_eng ,dataType=null ,value=null}
Output{idOutput=0,name=comment ,dataType=null ,value=null}
Output{idOutput=0,name=lvli ,dataType=null ,value=null}
Output{idOutput=0,name=code ,dataType=null ,value=null}
Output{idOutput=0,name=the_geom ,dataType=null ,value=null}
Output{idOutput=0,name=unit_id ,dataType=null ,value=null}
Output{idOutput=0,name=hier_id ,dataType=null ,value=null}
Output{idOutput=0,name=objectid ,dataType=null ,value=null}
Output{idOutput=0,name=num ,dataType=null ,value=null}
Output{idOutput=0,name=who_reg ,dataType=null ,value=null}
Output{idOutput=0,name=unittypeid ,dataType=null ,value=null}
Output{idOutput=0,name=continent ,dataType=null ,value=null}
Output{idOutput=0,name=latitude ,dataType=null ,value=null}

Method{idMethod=0,name=WHO: mm_who_populatedplaces}
Output{idOutput=0,name=NAME ,dataType=null ,value=null}
Output{idOutput=0,name=CODE ,dataType=null ,value=null}

194
Output{idOutput=0,name=LAT, dataType=null, value=null}
Output{idOutput=0,name=COMMENT, dataType=null, value=null}
Output{idOutput=0,name=LvlIDH, dataType=null, value=null}
Output{idOutput=0,name=HierTypeCode, dataType=null, value=null}
Output{idOutput=0,name=UnitType, dataType=null, value=null}
Output{idOutput=0,name=NUM, dataType=null, value=null}
Output{idOutput=0,name=UnitId, dataType=null, value=null}
Output{idOutput=0,name=DependUnitId, dataType=null, value=null}
Output{idOutput=0,name=geom, dataType=null, value=null}
Output{idOutput=0,name=LONG, dataType=null, value=null}
Output{idOutput=0,name=LvlId, dataType=null, value=null}
Output{idOutput=0,name=ISO_Ctry, dataType=null, value=null}
Method{idMethod=0,name=WHO:mm_admin2}
Output{idOutput=0,name=ADMIN2_NAM, dataType=null, value=null}
Output{idOutput=0,name=geom, dataType=null, value=null}
Output{idOutput=0,name=ID_SHORT, dataType=null, value=null}
Output{idOutput=0,name=ID_short, dataType=null, value=null}
Output{idOutput=0,name=ADMIN1_NAM, dataType=null, value=null}
Output{idOutput=0,name=LABEL_ID, dataType=null, value=null}
Method{idMethod=0,name=WHO:leishcomp}
Output{idOutput=0,name=st_pointonsurface, dataType=null, value=null}
Output{idOutput=0,name=iso_2_code, dataType=null, value=null}
Output{idOutput=0,name=iso_3_code, dataType=null, value=null}
Output{idOutput=0,name=ctry_short, dataType=null, value=null}
Output{idOutput=0,name=ctry_terr, dataType=null, value=null}
Output{idOutput=0,name=sovereign, dataType=null, value=null}
Method{idMethod=0,name=WHO:countries_indide_points}
Output{idOutput=0,name=objectid, dataType=null, value=null}
Output{idOutput=0,name=un_code, dataType=null, value=null}
Output{idOutput=0,name=iso_3_code, dataType=null, value=null}
Output{idOutput=0,name=shape_area, dataType=null, value=null}
Output{idOutput=0,name=sovereign, dataType=null, value=null}
Output{idOutput=0,name=iso_2_code, dataType=null, value=null}
Output{idOutput=0,name=who_code, dataType=null, value=null}
Output{idOutput=0,name=whos_role, dataType=null, value=null}
Output{idOutput=0,name=who_status, dataType=null, value=null}
Output{idOutput=0,name=lvlid, dataType=null, value=null}
Output{idOutput=0,name=shape_len, dataType=null, value=null}
Output{idOutput=0,name=who_region, dataType=null, value=null}
Output{idOutput=0,name=geom, dataType=null, value=null}
Method{idMethod=0,name=WHO:efs2008_countrydata}
Output{idOutput=0,name=aidsorphanslow, dataType=null, value=null}
Output{idOutput=0,name=aidsdeathsslow, dataType=null, value=null}
Output{idOutput=0,name=aidsdeaths, dataType=null, value=null}
Output{idOutput=0,name=who_code, dataType=null, value=null}
Output{idOutput=0,name=prev1549high, dataType=null, value=null}
Output{idOutput=0,name=clwhlow ,dataType=null , value=null}
Output{idOutput=0,name=aidsorphans ,dataType=null , value=null}
Output{idOutput=0,name=geom ,dataType=null , value=null}
Output{idOutput=0,name=efs2008short_link ,dataType=null , value=null}
Output{idOutput=0,name=aidsorphanshigh ,dataType=null , value=null}
Output{idOutput=0,name=clwh ,dataType=null , value=null}
Output{idOutput=0,name=aidsdeathshigh ,dataType=null , value=null}
Output{idOutput=0,name=efs2008shortlink ,dataType=null , value=null}
Output{idOutput=0,name=aidsorphanshigh ,dataType=null , value=null}
Output{idOutput=0,name=clwhhigh ,dataType=null , value=null}
Output{idOutput=0,name=plwhhigh ,dataType=null , value=null}
Output{idOutput=0,name=lwh15p ,dataType=null , value=null}
Output{idOutput=0,name=lwh15plow ,dataType=null , value=null}
Output{idOutput=0,name=plwhlow ,dataType=null , value=null}
Output{idOutput=0,name=plwh ,dataType=null , value=null}
Output{idOutput=0,name=wlwh15plow ,dataType=null , value=null}
Output{idOutput=0,name=plwhhigh ,dataType=null , value=null}
Output{idOutput=0,name=wlwh15phigh ,dataType=null , value=null}
Output{idOutput=0,name=iso2 ,dataType=null , value=null}
Output{idOutput=0,name=iso3 ,dataType=null , value=null}
Output{idOutput=0,name=yearofreport ,dataType=null , value=null}
Output{idOutput=0,name=ctlr ,dataType=null , value=null}
Output{idOutput=0,name=ctry_terr ,dataType=null , value=null}
Output{idOutput=0,name=ctlr ,dataType=null , value=null}
Output{idOutput=0,name=ARTICLE_UR ,dataType=null , value=null}
Output{idOutput=0,name=the_geom ,dataType=null , value=null}
Output{idOutput=0,name=SUBTOPIC_I ,dataType=null , value=null}
Output{idOutput=0,name=ARTICLE_TI ,dataType=null , value=null}
Output{idOutput=0,name=gid ,dataType=null , value=null}
Output{idOutput=0,name=ID ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm3 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm2 ,dataType=null , value=null}
Output{idOutput=0,name=NAME1 ,dataType=null , value=null}
Output{idOutput=0,name=NAME2 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm1 ,dataType=null , value=null}
Output{idOutput=0,name=POINT5 ,dataType=null , value=null}
Output{idOutput=0,name=LENHT ,dataType=null , value=null}
Output{idOutput=0,name=name_adm1 ,dataType=null , value=null}
Output{idOutput=0,name=OBJECT_ID ,dataType=null , value=null}
Output{idOutput=0,name=OBJECT_ID12 ,dataType=null , value=null}
Output{idOutput=0,name=shapeleng ,dataType=null , value=null}
Output{idOutput=0,name=shapearea ,dataType=null , value=null}
Output{idOutput=0,name=shape ,dataType=null , value=null}
Output{idOutput=0,name=geodb_oid ,dataType=null , value=null}
Output{idOutput=0,name=objectid ,dataType=null , value=null}
Output{idOutput=0,name=objectid12 ,dataType=null , value=null}
Output{idOutput=0,name=OBJECT_ID ,dataType=null , value=null}
Output{idOutput=0,name=OBJECT_ID12 ,dataType=null , value=null}
Output{idOutput=0,name=NAME1 ,dataType=null , value=null}
Output{idOutput=0,name=NAME2 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm1 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm2 ,dataType=null , value=null}
Output{idOutput=0,name=NAME1 ,dataType=null , value=null}
Output{idOutput=0,name=NAME2 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm1 ,dataType=null , value=null}
Output{idOutput=0,name=pcode_adm2 ,dataType=null , value=null}
Output{idOutput=0,name=NAME1 ,dataType=null , value=null}

Method{idMethod=0,name=WHO:leishecomp_resevoirs}
Method{idMethod=0,name=WHO:mm_adm0}
Method{idMethod=0,name=WHO:mmr_polbnda_adm3_mimu_polygon}
Output{idOutput=0,name=id_adm1 , dataType=null , value=null}
Output{idOutput=0,name=name_adm2 , dataType=null , value=null}
Output{idOutput=0,name=pcode_adm1 , dataType=null , value=null}
Output{idOutput=0,name=label_id , dataType=null , value=null}
Output{idOutput=0,name=id_adm2 , dataType=null , value=null}
Output{idOutput=0,name=id_adm3 , dataType=null , value=null}
Output{idOutput=0,name=name_adm3 , dataType=null , value=null}
Output{idOutput=0,name=object_id , dataType=null , value=null}

Method{idMethod=0,name=WHO:FLUID_Training_GismMonitoring_Geometry_Measures}
Output{idOutput=0,name=TOT_NO_OF_ADMISSIONS_AT_SITE , dataType=null , value=null}
Output{idOutput=0,name=NUM_OF_ILL_REPORTING_SITES , dataType=null , value=null}
Output{idOutput=0,name=CUMUL_LAB(confirmed)H1N1_DEATHS , dataType=null , value=null}
Output{idOutput=0,name=CUMUL_SUSPECTED_H1N1_CASES , dataType=null , value=null}
Output{idOutput=0,name=TOT_NUM_OF_SARI_REPORTED , dataType=null , value=null}
Output{idOutput=0,name=WKLY_LAB(confirmed)NEW_H1N1_DEATHS , dataType=null , value=null}
Output{idOutput=0,name=COUNTRY_NAME , dataType=null , value=null}
Output{idOutput=0,name=WKLY_SUSPECTED_NEW_H1N1_CASES , dataType=null , value=null}
Output{idOutput=0,name=SARI_SITES_PER_WEEK , dataType=null , value=null}
Output{idOutput=0,name=NUM_OF_SARI_REPORTING_SITES , dataType=null , value=null}
Output{idOutput=0,name=REGION_ID , dataType=null , value=null}
Output{idOutput=0,name=TOT_NUM_OF_PATIENT_VISITS , dataType=null , value=null}
Output{idOutput=0,name=SARI_POPULATION_COVERED , dataType=null , value=null}
Output{idOutput=0,name=GEOGRAPHIC_SPREAD , dataType=null , value=null}
Output{idOutput=0,name=TOT_NUM_OF_DEATHS , dataType=null , value=null}
Output{idOutput=0,name=UUID , dataType=null , value=null}
Output{idOutput=0,name=COUNTRY_ID , dataType=null , value=null}
Output{idOutput=0,name=WKLY_PROBABLE_NEW_H1N1_CASES , dataType=null , value=null}
Output{idOutput=0,name=IMPACT , dataType=null , value=null}
Output{idOutput=0,name=ISO_CODE , dataType=null , value=null}
Output{idOutput=0,name=LOCATION_ID , dataType=null , value=null}
Output{idOutput=0,name=ILI_POPULATION_COVERED , dataType=null , value=null}
Output{idOutput=0,name=ILI_SITES_REPORTING_BY_WEEK , dataType=null , value=null}
Output{idOutput=0,name=REGION_NAME , dataType=null , value=null}
Output{idOutput=0,name=LAST_UPDATED_DATE , dataType=null , value=null}
Output{idOutput=0,name=INTENSITY , dataType=null , value=null}
C. SERVICE1 SYNTACTIC DESCRIPTION

```java
Output{idOutput=0,name=TOT_RESPIRATORY_DISEASE_DEATHS, dataType=null, value=null}
Output{idOutput=0,name=REGION_CODE, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_ILI, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_PROBABLE_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=COUNTRY_GEOMETRY, dataType=null, value=null}
Output{idOutput=0,name=TREND, dataType=null, value=null}
Output{idOutput=0,name=SARI_DEATH_SURVEILLANCE, dataType=null, value=null}
Output{idOutput=0,name=TOT_NO_OF_SARI_DEATH_REP, dataType=null, value=null}
Output{idOutput=0,name=WKLY_LAB_CONFIRMED_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=WHO:admin_world}
Output{idOutput=0,name=iso3, dataType=null, value=null}
Output{idOutput=0,name=geom, dataType=null, value=null}
Output{idOutput=0,name=goid, dataType=null, value=null}
Output{idOutput=0,name=lvld, dataType=null, value=null}
Output{idOutput=0,name=uuid, dataType=null, value=null}
Output{idOutput=0,name=lvl, dataType=null, value=null}
Output{idOutput=0,name=area, dataType=null, value=null}
Output{idOutput=0,name=cntry_name, dataType=null, value=null}
Output{idOutput=0,name=length, dataType=null, value=null}
Output{idOutput=0,name=iso2, dataType=null, value=null}
Output{idOutput=0,name=adm, dataType=null, value=null}
Output{idOutput=0,name=lc, dataType=null, value=null}
Output{idOutput=0,name=cc2, dataType=null, value=null}
Output{idOutput=0,name=dsig, dataType=null, value=null}
Output{idOutput=0,name=language_name, dataType=null, value=null}
Output{idOutput=0,name=dd_long, dataType=null, value=null}
Output{idOutput=0,name=nt, dataType=null, value=null}
Output{idOutput=0,name=adm2, dataType=null, value=null}
Output{idOutput=0,name=fc, dataType=null, value=null}
Output{idOutput=0,name=ufi, dataType=null, value=null}
Output{idOutput=0,name=modify_date, dataType=null, value=null}
Output{idOutput=0,name=full_name, dataType=null, value=null}
Output{idOutput=0,name=region, dataType=null, value=null}
Output{idOutput=0,name=generic, dataType=null, value=null}
Output{idOutput=0,name=dsg_name, dataType=null, value=null}
Output{idOutput=0,name=sort_name, dataType=null, value=null}
Output{idOutput=0,name=pc, dataType=null, value=null}
Output{idOutput=0,name=dd_lat, dataType=null, value=null}
Output{idOutput=0,name=adm1, dataType=null, value=null}
Output{idOutput=0,name=rc, dataType=null, value=null}
Output{idOutput=0,name=dms_long, dataType=null, value=null}
Output{idOutput=0,name=dim, dataType=null, value=null}
```

198
<table>
<thead>
<tr>
<th>idOutput</th>
<th>name</th>
<th>dataType</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>uni</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>dms_lat</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>utm</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>cc1</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>dsg_text</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>short_form</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>full_name</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>sovereign</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>country_code</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>shape_area</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>id</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>iso_2_code</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>geodb_oid</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>related_links</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>country_name</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>yellow_fever_requirement</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>yellow_fever_recommendation</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>geom</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>who_region</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>malariarisk</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>malaria_prevention</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>who_status</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>rabies_code</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>Water_Class</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>OBJECTID</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>geom</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>MOD_Date</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>geodb_oid</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>SHAPE_Area</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0</td>
<td>SHAPE_Length</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

**Method:**

```python
Method{idMethod=0, name=WHO:ith_list_new}
```
C. SERVICE1 SYNTACTIC DESCRIPTION

Output{idOutput=0,name=long ,dataType=null ,value=null}
Output{idOutput=0,name=popgroupclass ,dataType=null ,value=null}
Output{idOutput=0,name=geom ,dataType=null ,value=null}
Output{idOutput=0,name=country ,dataType=null ,value=null}
Output{idOutput=0,name=popgroup ,dataType=null ,value=null}
Output{idOutput=0,name=area ,dataType=null ,value=null}
Output{idOutput=0,name=year ,dataType=null ,value=null}
Output{idOutput=0,name=sited ,dataType=null ,value=null}

Method{idMethod=0,name=WHO:mm_nga_placenames}
Output{idOutput=0,name=pc ,dataType=null ,value=null}
Output{idOutput=0,name=MODIFY, dataType=null ,value=null}
Output{idOutput=0,name=dsg ,dataType=null ,value=null}
Output{idOutput=0,name=CC1, dataType=null ,value=null}
Output{idOutput=0,name=long ,dataType=null ,value=null}
Output{idOutput=0,name= NT, dataType=null ,value=null}
Output{idOutput=0,name= SHORT FORM, dataType=null ,value=null}
Output{idOutput=0,name= uf i , dataType=null ,value=null}
Output{idOutput=0,name=ADM1, dataType=null ,value=null}
Output{idOutput=0,name= RC, dataType=null ,value=null}
Output{idOutput=0,name=full name ,dataType=null ,value=null}
Output{idOutput=0,name= SORT NAME, dataType=null ,value=null}
Output{idOutput=0,name=UNI , dataType=null ,value=null}
Output{idOutput=0,name=CC2, dataType=null ,value=null}
Output{idOutput=0,name= FC, dataType=null ,value=null}
Output{idOutput=0,name= JOG, dataType=null ,value=null}
Output{idOutput=0,name= UTM, dataType=null ,value=null}
Output{idOutput=0,name= lat , dataType=null ,value=null}
Output{idOutput=0,name=ADM2, dataType=null ,value=null}
Output{idOutput=0,name=DMS_LAT, dataType=null ,value=null}
Output{idOutput=0,name=LC, dataType=null ,value=null}
Output{idOutput=0,name=DMS_LONG, dataType=null ,value=null}
Output{idOutput=0,name= FULL NAME, dataType=null ,value=null}
Output{idOutput=0,name=geom ,dataType=null ,value=null}

Method{idMethod=0,name=WHO:GIM_MONITORING_GEOMETRY_MEASURES}
Output{idOutput=0,name=WKLY_PROBABLE_NEW_H1N1_CASES, dataType=null ,value=null}
Output{idOutput=0,name=TOT_NO_OF_ADMISSIONS_AT_SITE, dataType=null ,value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_H1N1_DEATHS, dataType=null ,value=null}
Output{idOutput=0,name=UUID, dataType=null ,value=null}
Output{idOutput=0,name= GEOGRAPHIC_SPREAD, dataType=null ,value=null}
Output{idOutput=0,name=TOT_NUM_OF_ILI ,dataType=null ,value=null}
Output{idOutput=0,name=TOT_NUM_OF_SARI_REPORTED, dataType=null ,value
= null}
Output{idOutput=0,name=CUMUL_PROBABLE_H1N1_CASES, dataType=null ,value
= null}
Output{idOutput=0,name=COUNTRY_GEOMETRY, dataType=null ,value=null}
Output{idOutput=0,name=NUM_OF_SARI_REPORTING_SITES, dataType=null ,value
= null}
Output{idOutput=0,name=LAST_UPDATED_DATE, dataType=null, value=null}
Output{idOutput=0,name=ILI_POPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=NUM_OFILI_REPORTING_SITES, dataType=null, value=null}
Output{idOutput=0,name=SARIPOPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=TREND, dataType=null, value=null}
Output{idOutput=0,name=LOCATION_ID, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OFDEATHS, dataType=null, value=null}
Output{idOutput=0,name=INTENSITY, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_SUSPECTED_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=REGIONID, dataType=null, value=null}
Output{idOutput=0,name=COUNTRY_ID, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OFPATIENT_VISITS, dataType=null, value=null}
Output{idOutput=0,name=WKLYLAB_CONFIRMED_NEW_H1N1_DEATHS, dataType=null, value=null}
Output{idOutput=0,name=WKLY_SUSPECTED_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=TOT_NO_OF_SARI_DEATH_REP, dataType=null, value=null}
Method{idMethod=0,name=WHO:leishecomp_sandflies}
Output{idOutput=0,name=subtopic_i, dataType=null, value=null}
Output{idOutput=0,name=the_geom, dataType=null, value=null}
Output{idOutput=0,name=gid, dataType=null, value=null}
Output{idOutput=0,name=article_ti, dataType=null, value=null}
Output{idOutput=0,name=article_title, dataType=null, value=null}
Method{idMethod=0,name=WHO:FLUID_DEV_GIM_MONITORING_GEOMETRY_MEASURES}
Output{idOutput=0,name=REGION_NAME, dataType=null, value=null}
Output{idOutput=0,name=COUNTRY_NAME, dataType=null, value=null}
Output{idOutput=0,name=NUM_OF_SARI_REPORTING_SITES, dataType=null, value=null}
Output{idOutput=0,name=WKLY_SUSPECTED_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=WKLY_PROBABLE_NEW_H1N1_CASES, dataType=null, value=null}
C. SERVICE1 SYNTACTIC DESCRIPTION

Output{idOutput=0,name=LOCATION_ID, dataType=null, value=null}
Output{idOutput=0,name=SARI_SITES_PER_WEEK, dataType=null, value=null}
Output{idOutput=0,name=TOT_NO_OF_ADMISSIONS_AT_SITE, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_SUSPECTED_HINI_CASES, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_PROBABLE_HINI_CASES, dataType=null, value=null}
Output{idOutput=0,name=WKLY_LAB_CONFIRMED_NEW_HINI DEATHS, dataType=null, value=null}
Output{idOutput=0,name=TREND, dataType=null, value=null}
Output{idOutput=0,name=NUM_OF_ILI_REPORTING_SITES, dataType=null, value=null}
Output{idOutput=0,name=MORTALITY, dataType=null, value=null}
Output{idOutput=0,name=SARI_DEATH_SURVEILLANCE, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_HINI_CASES, dataType=null, value=null}
Input{idOutput=0,name=INTENSITY, dataType=null, value=null}
Output{idOutput=0,name=REGION_CODE, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_ILI, dataType=null, value=null}
Output{idOutput=0,name=ISO_CODE, dataType=null, value=null}
Output{idOutput=0,name=ILI_SITES_REPORTING_BY_WEEK, dataType=null, value=null}
Output{idOutput=0,name=WKLY_LAB_CONFIRMED_NEW_HINI CASES, dataType=null, value=null}
Output{idOutput=0,name=REGION_ID, dataType=null, value=null}
Output{idOutput=0,name=LOCATION, dataType=null, value=null}
Output{idOutput=0,name=DEATHS, dataType=null, value=null}
Output{idOutput=0,name=RESPIRATORY_DISEASE_DEATHS, dataType=null, value=null}
Output{idOutput=0,name=NEW_ILI_CASES, dataType=null, value=null}
Output{idOutput=0,name=SARI_DEATH_Reported, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_POPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=NEW_PATIENT_VISITS, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_HINI DEATHS, dataType=null, value=null}
Output{idOutput=0,name=IMPACT, dataType=null, value=null}
Output{idOutput=0,name=UUID, dataType=null, value=null}
Output{idOutput=0,name=LAST_UPDATED_DATE, dataType=null, value=null}
Output{idOutput=0,name=ILL_POPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_SARI_REPORTED, dataType=null, value=null}
Method{idMethod=0,name=WHO:disputedBorders_white}
Output{idOutput=0,name=the_geom, dataType=null, value=null}
Output{idOutput=0,name=NAME, dataType=null, value=null}
Method{idMethod=0,name=WHO:myanmarHealthFacilities}
Output{idOutput=0,name=gid, dataType=null, value=null}
Output{idOutput=0,name=who_reg, dataType=null, value=null}
Output{idOutput=0,name=num, dataType=null, value=null}
Output{idOutput=0,name=longitude, dataType=null, value=null}
Output{idOutput=0,name=iso_ctry, dataType=null, value=null}
Output{idOutput=0,name=the_geom, dataType=null, value=null}
Output{idOutput=0,name=country, dataType=null, value=null}
Output{idOutput=0,name=continent, dataType=null, value=null}
Output{idOutput=0,name=code, dataType=null, value=null}
Output{idOutput=0,name=unittypename_eng, dataType=null, value=null}
Output{idOutput=0,name=lv, dataType=null, value=null}
Output{idOutput=0,name=objectid, dataType=null, value=null}
Output{idOutput=0,name=comment, dataType=null, value=null}
Output{idOutput=0,name=unittypeid, dataType=null, value=null}
Output{idOutput=0,name=latitude, dataType=null, value=null}
Output{idOutput=0,name=unitid, dataType=null, value=null}
Method{idMethod=0,name=WHO:leishecomp}
Output{idOutput=0,name=hier_id, dataType=null, value=null}
Method{idMethod=0,name=WHO:africa_airports}
Output{idOutput=0,name=USE_DESCRI, dataType=null, value=null}
Output{idOutput=0,name=K0, dataType=null, value=null}
Output{idOutput=0,name=ZV3, dataType=null, value=null}
Output{idOutput=0,name=NA3, dataType=null, value=null}
Output{idOutput=0,name=FS, dataType=null, value=null}
Output{idOutput=0,name=FI, dataType=null, value=null}
Output{idOutput=0,name=NUM, dataType=null, value=null}
Output{idOutput=0,name=SARI, dataType=null, value=null}
Output{idOutput=0,name=ILI, dataType=null, value=null}
Output{idOutput=0,name=NUM, dataType=null, value=null}
Method{idMethod=0,name=WHO:FLUID_TEST_GIM_MONITING_GEOMETRY_MEASURES}
Output{idOutput=0,name=WKL_PROBABLE_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_PATIENT_VISITS, dataType=null, value=null}
Output{idOutput=0,name=WKL_SUSPECTED_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=REGION_CODE, dataType=null, value=null}
Output{idOutput=0,name=COUNTRY_NAME, dataType=null, value=null}
Output{idOutput=0,name=SARI_DISEASE_SURVEILLANCE, dataType=null, value=null}
Output{idOutput=0,name=LOCATION_ID, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_DEATHS, dataType=null, value=null}
Output{idOutput=0,name=WKL_LAB_CONFIRMED_NEW_H1N1_DEATHS, dataType=null, value=null}
Output{idOutput=0,name=TREND, dataType=null, value=null}
Output{idOutput=0,name=NUM_OF_ILI_REPORTING_SITES, dataType=null, value=null}
Output{idOutput=0,name=NUM_OF_SARI_REPORTING_SITES, dataType=null, value=null}
Output{idOutput=0,name=UUID, dataType=null, value=null}
Output{idOutput=0,name=REGION_NAME, dataType=null, value=null}
Output{idOutput=0,name=COUNTRY_GEOMETRY, dataType=null, value=null}
Output{idOutput=0,name=REGION_ID, dataType=null, value=null}
Output{idOutput=0,name=INTENSITY, dataType=null, value=null}
Output{idOutput=0,name=REGION_ID, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_SARI_REPORTED, dataType=null, value=null}
Output{idOutput=0,name=ILL_POPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=ISO_CODE, dataType=null, value=null}
Output{idOutput=0,name=IMPACT, dataType=null, value=null}
Output{idOutput=0,name=LAST_UPDATED_DATE, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_PROBABLE_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=TOT_NO_OF_ADMISSIONS_AT_SITE, dataType=null, value=null}
Output{idOutput=0,name=SARI_SITES_PER_WEEK, dataType=null, value=null}
Output{idOutput=0,name=GEOGRAPHIC_SPREAD, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=CUMUL_LAB_CONFIRMED_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=WKL_LAB_CONFIRMED_NEW_H1N1_CASES, dataType=null, value=null}
Output{idOutput=0,name=ILL_SITES_REPORTING_BY_WEEK, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_ILI, dataType=null, value=null}
Output{idOutput=0,name=SARI_POPULATION_COVERED, dataType=null, value=null}
Output{idOutput=0,name=TOT_NO_OF_SARI_DEATH_REP, dataType=null, value=null}
Output{idOutput=0,name=ILI_SITES_REPORTING_BY_WEEK, dataType=null, value=null}
Output{idOutput=0,name=TOT_NUM_OF_ILI, dataType=null, value=null}
Method{idMethod=0,name=WHO:leishcomp_humans}
Output{idOutput=0,name=article.url, dataType=null, value=null}
Output{idOutput=0,name=article.ti, dataType=null, value=null}
Output{idOutput=0,name=subtopic.i, dataType=null, value=null}
Output {idOutput=0, name='the_geom', dataType=null, value=null}
Output {idOutput=0, name='gid', dataType=null, value=null}
Method {idMethod=0, name='WHO:world_influenza_humanhealthlab'}
Output {idOutput=0, name='pcr_capability', dataType=null, value=null}
Output {idOutput=0, name='name_of_na', dataType=null, value=null}
Output {idOutput=0, name='objectid', dataType=null, value=null}
Output {idOutput=0, name='lvlid', dataType=null, value=null}
Output {idOutput=0, name='locationhu', dataType=null, value=null}
Output {idOutput=0, name='wbcase', dataType=null, value=null}
Output {idOutput=0, name='nberandiag', dataType=null, value=null}
Output {idOutput=0, name='othercas', dataType=null, value=null}
Output {idOutput=0, name='poultrycas', dataType=null, value=null}
Output {idOutput=0, name='nberhulab', dataType=null, value=null}
Output {idOutput=0, name='long', dataType=null, value=null}
Output {idOutput=0, name='hucas', dataType=null, value=null}
Output {idOutput=0, name='lat', dataType=null, value=null}
Method {idMethod=0, name='WHO:mmr_unjlc_roads'}
Output {idOutput=0, name='shape_length', dataType=null, value=null}
Output {idOutput=0, name='objectid', dataType=null, value=null}
Output {idOutput=0, name='kmldescrip', dataType=null, value=null}
Output {idOutput=0, name='geodb_oid', dataType=null, value=null}
Output {idOutput=0, name='kmlname', dataType=null, value=null}
Output {idOutput=0, name='geom', dataType=null, value=null}
Method {idMethod=0, name='WHO:admin_world_centroid'}
Output {idOutput=0, name='goid', dataType=null, value=null}
Output {idOutput=0, name='geom', dataType=null, value=null}
Output {idOutput=0, name='length', dataType=null, value=null}
Output {idOutput=0, name='uuid', dataType=null, value=null}
Output {idOutput=0, name='lvlid', dataType=null, value=null}
Output {idOutput=0, name='area', dataType=null, value=null}
Output {idOutput=0, name='lv', dataType=null, value=null}
Method {idMethod=0, name='WHO:member_states'}
Output {idOutput=0, name='who_status', dataType=null, value=null}
Output {idOutput=0, name='objectid', dataType=null, value=null}
Output {idOutput=0, name='sovereign', dataType=null, value=null}
Output {idOutput=0, name='iso_2_code', dataType=null, value=null}
Output {idOutput=0, name='entryTerr', dataType=null, value=null}
Output {idOutput=0, name='who_region', dataType=null, value=null}
Output {idOutput=0, name='shape_area', dataType=null, value=null}
Output {idOutput=0, name='who_code', dataType=null, value=null}
Output {idOutput=0, name='shape_len', dataType=null, value=null}
Output {idOutput=0, name='un_code', dataType=null, value=null}
Output {idOutput=0, name='geom', dataType=null, value=null}
Output {idOutput=0, name='iso_3_code', dataType=null, value=null}
Output {idOutput=0, name='lvlid', dataType=null, value=null}
Method {idMethod=0, name='WHO:world_countries_inside_points'}
Output {idOutput=0, name='who_code', dataType=null, value=null}
Output {idOutput=0, name='sovereign', dataType=null, value=null}
Output {idOutput=0, name='lvlid', dataType=null, value=null}
Output{idOutput=0,name=cntry_terr ,dataType=null ,value=null}
Output{idOutput=0,name=un_code ,dataType=null ,value=null}
Output{idOutput=0,name=geom ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=iso_3_code ,dataType=null ,value=null}
Output{idOutput=0,name=iso_2_code ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}

Method{idMethod=0,name=WHO:world_countries}
Output{idOutput=0,name=start_date ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_memb ,dataType=null ,value=null}
Output{idOutput=0,name=trans_oper ,dataType=null ,value=null}
Output{idOutput=0,name=cntry_terr ,dataType=null ,value=null}
Output{idOutput=0,name=who_region ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_engl ,dataType=null ,value=null}
Output{idOutput=0,name=who_status ,dataType=null ,value=null}
Output{idOutput=0,name=who_code ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_alp_1 ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_alpha ,dataType=null ,value=null}
Output{idOutput=0,name=end_date ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_num_c ,dataType=null ,value=null}
Output{idOutput=0,name=labelrank ,dataType=null ,value=null}
Output{idOutput=0,name=sovereign ,dataType=null ,value=null}
Output{idOutput=0,name=un_code ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_code1 ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_alp_2 ,dataType=null ,value=null}
Output{idOutput=0,name=off_reg_al ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_status ,dataType=null ,value=null}
Output{idOutput=0,name=iso_2_code ,dataType=null ,value=null}
Output{idOutput=0,name=id ,dataType=null ,value=null}
Output{idOutput=0,name=the_geom ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_code ,dataType=null ,value=null}
Output{idOutput=0,name=update_date ,dataType=null ,value=null}
Output{idOutput=0,name=iso_3_code ,dataType=null ,value=null}
Output{idOutput=0,name=isdisputed ,dataType=null ,value=null}
Output{idOutput=0,name=terminated ,dataType=null ,value=null}
Output{idOutput=0,name=ctry_short ,dataType=null ,value=null}

Method{idMethod=0,name=WHO:mmr_rds2_tm_vmap1_polyline}
Output{idOutput=0,name=edg_id ,dataType=null ,value=null}
Output{idOutput=0,name=f_code ,dataType=null ,value=null}
Output{idOutput=0,name=tile_id ,dataType=null ,value=null}
Output{idOutput=0,name=wtc_desci ,dataType=null ,value=null}
Output{idOutput=0,name=f_code_des ,dataType=null ,value=null}
Output{idOutput=0,name=acc ,dataType=null ,value=null}
Output{idOutput=0,name=acc_desci ,dataType=null ,value=null}
Output{idOutput=0,name=id ,dataType=null ,value=null}
Output{idOutput=0,name=wtc ,dataType=null ,value=null}
Output{idOutput=0,name=geo_db_oid ,dataType=null ,value=null}
Output{idOutput=0,name=object_id ,dataType=null ,value=null}
Output{idOutput=0,name=shape_length ,dataType=null ,value=null}
Output{idOutput=0,name=geom ,dataType=null ,value=null}

Method{idMethod=0,name=WHO:mm_admin1}
Output{idOutput=0,name=geom ,dataType=null ,value=null}
C. SERVICE1 SYNTACTIC DESCRIPTION

Output {idOutput=0, name=LENGTH, dataType=null, value=null}
Output {idOutput=0, name=NAME1, dataType=null, value=null}
Output {idOutput=0, name=ID, dataType=null, value=null}
Output {idOutput=0, name=PARTS, dataType=null, value=null}
Output {idOutput=0, name=POINTS, dataType=null, value=null}
Output {idOutput=0, name=AREA, dataType=null, value=null}
Output {idOutput=0, name=NAME2, dataType=null, value=null}
Appendix D

Procedure-Oriented Service Model (POSM)

Below we show the POSM ontology.

Listing D.1: POSM ontology

```xml
<?xml version="1.0"?>
<!-- this file is not intended as human-readable; see http://www.wsmo.org/ns/posm/0.1/index.n3 instead -->
<rdf:Property rdf:about="http://www.wsmo.org/ns/posm/0.1#hasOperation">
  <rdfs:domain>
    <rdfs:Class rdf:about="http://www.wsmo.org/ns/posm/0.1#Service"/>
  </rdfs:domain>
  <rdfs:range>
    <rdfs:Class rdf:about="http://www.wsmo.org/ns/posm/0.1#Operation"/>
  </rdfs:range>
</rdf:Property>
<rdf:Property rdf:about="http://www.wsmo.org/ns/posm/0.1#hasInputMessage">
  <rdfs:domain rdf:resource="http://www.wsmo.org/ns/posm/0.1#Operation"/>
  <rdfs:range>
    <rdfs:Class rdf:about="http://www.wsmo.org/ns/posm/0.1#Message"/>
  </rdfs:range>
</rdf:Property>
<rdf:Property rdf:about="http://www.wsmo.org/ns/posm/0.1#hasOutputMessage">
  <rdfs:domain rdf:resource="http://www.wsmo.org/ns/posm/0.1#Operation"/>
  <rdfs:range>
    <rdfs:Class rdf:about="http://www.wsmo.org/ns/posm/0.1#MessagePart"/>
  </rdfs:range>
</rdf:Property>
</rdf:Class>
```
D. PROCEDURE-ORIENTED SERVICE MODEL (POSM)
Appendix E

Glossary of Terms

E.1 Preliminaries Background Technologies

E.1.1 XML

Extensible Markup Language (XML) is a simple, flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML has also played an important role in the exchange of a wide variety of data on the Web and elsewhere.

E.1.2 RDF, RDF schema, OWL

As described in (BHBL09), the Data Web uses RDF to make statements in order to connect arbitrary entities of the world. RDF, specified by W3C, is a Semantic Web technology for describing things (resources) and their interrelations. RDF makes statements about resources in subject-predicate-object form. Such triples can also be represented as a directed graph. Each triple represents a statement about two resources (entities). In RDF, subjects and properties are always resources. Objects can either be a resource or a literal value.

Various serialization formats exist for RDF. The RDF serialization formats generally used are RDF/XML, N3 or Turtle.

E.1.3 SPARQL

The most common query language for RDF is SPARQL, which has similar capabilities as SQL queries. Our research uses SPARQL to query the provided DBpedia entities
to find match between concepts and web services parameters. It became a W3C recommendation in 2008 and is the key query language of the Semantic Web. It provides capabilities to retrieve information of unknown relationships stated in RDF documents. Query results can consist of any number of results either encoded in XML, JSON or RDF triples/graphs.

**E.1.4 Ontologies**

An ontology plays a specific role in computer science and is even the key enabling technology for the Semantic Web. Thomas and GRUBER describe an ontology as an explicit specification of a conceptualization. Conceptualization can be regarded as how we understand the real world. It represent something in the real world with concepts and relations among those. A specification is then an abstract description of those concepts.

**E.1.5 SDO**

SDO is a specification for a programming model that unifies data programming across data source types and provides robust support for common application patterns in a disconnected way [22].

**E.1.6 REST APIs**

REST is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant Web service design model. The term REST was introduced and defined in 2000 by Roy fielding in his doctoral dissertation. A REST API is a library of functions that access through HTTP protocol. APIs are designed for machine consumption and follow the style called REST. Generally, APIs are described HTML page documentation, it is useful to a human developer. Public APIs on the Web are called RESTful Web services. Web APIs enable developers to create new applications leveraging various kinds of content and functionality.
Appendix F

RESTful and WFS Services

Below we show a list of RESTful and WFS services.

Listing F.1: RESTful services

the_formated_address&SuppressError=Yes_or_No
the_street_num_and_name_or_street1$_street2&city=the_city&state=
the_state_or_province&zip=the_ZIP_or_postal_code&country=
the_country&SuppressError=Yes_or_No
the_point_of_interest&city=the_city&state=the_state_or_province&country=
the_country&SuppressError=Yes_or_No
the_city&state=the_state_or_province&country=the_country&SuppressError=
Yes_or_No
the_state_or_province&country=the_country&SuppressError=Yes_or_No
the_ID&country=the_country&SuppressError=Yes_or_No
the_ID&zip=the_ZIP_or_postal_code&country=the_country&SuppressError=
Yes_or_No
the_ID&Longitude=your_longitude&Latitude=your_latitude&Language=
your_preferred_language
http://mlbs.net/nacgeoservicesv4.5/simplexmlapi.asp#Directions
http://mlbs.net/nacgeoservicesv4.5/simplexmlapi.asp#Directions
http://mlbs.net/nacgeoservicesv4.5/simplexmlapi.asp#trafficinfo
http://www.happenr.com/webservices
http://groups.google.com/group/messagemost-api/web/api-documentation
http://api.eventful.com/rest/events/search?app_key=
p4t8BFcLDtCzpdxS&location=Madrid
http://api.eventful.com/rest/venues/search?app_key=
p4t8BFcLDtCzpdxS&location=Madrid
F. RESTFUL AND WFS SERVICES

http://ws.geonames.org/astergdem?lat=50.01&lng=10.2&type=XML
http://ws.geonames.org/cities?north=44.1&south=-9.9&east=-22.4&west=55.2
http://ws.geonames.org/countryCode?lat=47.03&lng=10.2&type=xml
http://ws.geonames.org/countryInfo?country=ES
http://ws.geonames.org/countrySubdivision?lat=47.03&lng=10.2
http://ws.geonames.org/extendedFindNearby?lat=47.3&lng=9
http://ws.geonames.org/findNearbyPlaceName?lat=47.3&lng=9
http://ws.geonames.org/findNearbyPostalCodes?postalcode=28025&country=ES&radius=10
http://ws.geonames.org/findNearbyStreetsOSM?lat=37.451&lng=-122.18
http://ws.geonames.org/findNearestIntersectionOSM?lat=37.451&lng=-122.18
http://ws.geonames.org/ocean?lat=40.78343&lng=-43.96625
http://ws.geonames.org/timezone?lat=47.01&lng=10.2
http://ws.geonames.org/wikipediaBoundingBox?north=44.1&south=-9.9&east=-22.4&west=55.2
http://ipinfo.db.com/ip_query.php?ip=74.125.45.100
http://ipinfo.db.com/ip_query_country.php?ip=74.125.45.100
http://www.43.places.com/service/search_places?api_key=1234&q=madrid
http://en.tixik.com/api/search?api_key=madrid
http://en.tixik.com/localize?lng=46.9989876381546&lat=2.7191162109375
http://trustedplaces.com/api/rest/place/ge?country=spain&city=madrid&key=6adc6c8c5ae40f06734cdd09a0199f1
http://trustedplaces.com/api/rest/place/geo/?lng=-0.126236&lat=51.500152&rad=500&key=6adc6c8c5ae40f06734cdd09a0199f1
http://api.themoviedb.org/2.1/Movie.search/en/xml/5/a126c6ca2a67171b0056faeb083b3b1
/Transformers
http://api.themoviedb.org/2.1/Person.search/en/xml/5/a126c6ca2a67171b0056faeb083b3b1
/brad=pit
http://api.emusic.com/album/search?apiKey=xa572k4ahavg7zftcg82t3np&term=love+songs
http://api.emusic.com/book/search?apiKey=xa572k4ahavg7zftcg82t3np&term=balaxy
http://api.emusic.com/artist/search?apiKey=xa572k4ahavg7zftcg82t3np&term= bob+marley
http://api.emusic.com/label/search?apiKey=xa572k4ahavg7zftcg82t3np&term=orchard
http://api.thedemap.com/search/deals/?l=seattle&q=pizza&a=1&key=
http://api.thedemap.com/search/businesses/?l=seattle&q=pizza&a=1&key=
http://api.thedemap.com/search/businesses/?l=+37.7786−122.4172&q=0.1&quality=1&key=
http://fireeagle.yahoo.net/developer/explorer/0.1/user
http://geocoder.us/member/service/csv/geocode?address=1600+Pennsylvania+Ave%2C+Washington+DC
http://where.yahooapis.com/geocode?location=701+First+Ave,+Sunnyvale,+CA&appid=pIpjm
http://where.yahooapis.com/geocode?location=701+First+Ave,+Sunnyvale,+CA&appid=pIpjm
http://where.yahooapis.com/geocode?name=Yosemite+National+Park&appid=pIpjm
http://TPVtapF2dLkMmvq3iq6rexVQB8YvgH5b3LSzYuosLgYoQweiTTOJ5egmureRI
http://TPVtapF2dLkMmvq3iq6rexVQB8YvgH5b3LSzYuosLgYoQweiTTOJ5egmureRI

212
http://where.yahooapis.com/geocode?woeid=12797523&appid=plpjmvTV34FUDkMmvq3qlexVQB8VygH5b3LSzYyosLgYoQweiiTTQJ5egmueRI
http://where.yahooapis.com/geocode?line1=100+Market+St.&line2=San+Francisco,CA&appid=plpjmvTV34FUDkMmvq3qlexVQB8VygH5b3LSzYyosLgYoQweiiTTQJ5egmueRI
http://where.yahooapis.com/geocode?house=151&street=3rd+St&postal=94103&city=San+Francisco&state=CA&country=US&appid=plpjmvTV34FUDkMmvq3qlexVQB8VygH5b3LSzYyosLgYoQweiiTTQJ5egmueRI
http://where.yahooapis.com/geocode?location=Eiffel+Tower&locale=fr_FR&appid=plpjmvTV34FUDkMmvq3qlexVQB8VygH5b3LSzYyosLgYoQweiiTTQJ5egmueRI
http://where.yahooapis.com/geocode?location=37.787082+-122.400929&flags=R&appid=plpjmvTV34FUDkMmvq3qlexVQB8VygH5b3LSzYyosLgYoQweiiTTQJ5egmueRI
http://api2.citysearch.com/search/locations?what=bakery&where=91101&publisher=acme&api_key=dwak67nmwcua6t3zwqw47ysd
http://api2.citysearch.com/profile/?listing_id=273&publisher=acme&client_ip=123.45.67.89&api_key=dwak67nmwcua6t3zwqw47ysd
http://api2.citysearch.com/reviews/?where=90069&tag_name=restaurant&publisher=acme&api_key=dwak67nmwcua6t3zwqw47ysd
http://www.cyclestreets.net/api/journey.xml?key=ev&start_longitude=0.169687&start_latitude=51.535231&finish_longitude=-0.115356&finish_latitude=51.521935&plan=quietest
http://cambridge.cyclestreets.net/api/nearestpoint.xml?key=yourapikey&longitude=11.17950&latitude=52.205302
http://georeach.apigee.com/neighborhood?key=demo&lat=34.076207993438&lng=-118.397965182076&state=ca
http://local.yahooapis.com/MapsService/V1/trafficData?appid=YdnDemo&street=701+First+Ave&city=Sunnyvale&state=CA

Listing F.2: WFS services

1 Servicio WFS: http://geo.ifip.tuwien.ac.at/geoserver/ows?SERVICE=WFS&REQUEST=GetCapabilities
2 Servicio WFS: http://ide.gijon.es:8080/geoserver/wfs?request=GetCapabilities
3 Servicio WFS: http://www.geostor.arkansas.gov/arggis/services/FEATURESERVICES/GEOSTOR_CULTU/MapServer/WFSServer?SERVICE=WFS&REQUEST=GetCapabilities
4 Servicio WFS: http://gis.srh.noaa.gov/arggis/services/AtStormViewer/MapServer/WFSServer?SERVICE=WFS&REQUEST=GetCapabilities
5 Servicio WFS: http://94.81.205.97/geoserver/ows?SERVICE=WFS&REQUEST=GetCapabilities
7 Servicio WFS: http://picws.harphsea.si/geoserver/wms/reflect?SERVICE=WFS&REQUEST=GetCapabilities
8 Servicio WFS: http://apollo.erdas.com/erdas-apollo/vector/CherokeeVector20067&SERVICE=WFS&REQUEST=GetCapabilities
9 Servicio WFS: http://maps.trimet.org/geoserver/wfs?REQUEST=SERVICE=WFS&REQUEST=GetCapabilities
10 Servicio WFS: http://pr5sit.regione.calabria.it/geoserver/ows?SERVICE=WFS&REQUEST=GetCapabilities
11 Servicio WFS: http://pacoos.coas.oregonstate.edu/wfsconnector/com.esri.wfs,
F. RESTFUL AND WFS SERVICES

23 Esrimap/pacoos_biology_wfs?&SERVICE=CSW&REQUEST=GetCapabilities
24 Servicio WFS: http://geoportal.cm-agueda.pt/geoserver/wms?&SERVICE=WFS&REQUEST=
25 GetCapabilities
26 Servicio WFS: http://geobretagne.fr/geoserver/bzh/wfs?SERVICE=WFS&REQUEST=
27 GetCapabilities
29 GetCapabilities
30 Servicio WFS: http://www.ideaportaligm.gob.ec/50k/wfs?SERVICE=WFS&REQUEST=
31 GetCapabilities
32 Servicio WFS: http://www.mirandalucia.es/dea100/wfs?SERVICE=WFS&REQUEST=
33 GetCapabilities
34 Servicio WFS: http://www.mirandalucia.es/dea100/wfs?&SERVICE=WFS&REQUEST=
35 GetCapabilities
36 Servicio WFS: http://213.160.247.143/webinnsyn/wfs/?REQUEST=GetCapabilities
37 &VM=1.0.0
39 GetCapabilities
40 Servicio WFS: http://crius.nodc.no:8080/geoserver/ows?SERVICE=WFS&REQUEST=
41 GetCapabilities
42 Servicio WFS: http://www.apps.geovista.psu.edu/geoserver/wfs?REQUEST=
43 GetCapabilities&VERSION=1.1.0&SERVICE=WFS
44 Servicio WFS: http://nlwis-snite1.agr.gc.ca/cgi-bin/ogc/ind2001_wms_e?
45 &SERVICE=WFS&REQUEST=GetCapabilities
46 Servicio WFS: http://nlwis-snite1.agr.gc.ca/cgi-bin/ogc/ind2001_wms_l?&
47 SERVICE=WFS&REQUEST=GetCapabilities
49 GetCapabilities
50 Servicio WFS: http://spatialserver.net:8080/geoserver-nijel/wfs?SERVICE=
51 WFS&REQUEST=GetCapabilities
52 Servicio WFS: http://webservices.ionicsoft.com/unData/wfs/UN?&SERVICE=
53 CSW&REQUEST=GetCapabilities
55 WFS&REQUEST=GetCapabilities
56 Servicio WFS: http://imaa.geosdi.org/geoserver/wms?&SERVICE=WFS&REQUEST=
57 GetCapabilities
58 Servicio WFS: http://datum.unigeo.igeograf.unam.mx:8080/geoserver/ows?
59 SERVICE=WFS&REQUEST=GetCapabilities
61 GetCapabilities
62 Servicio WFS: http://datum.unigeo.igeograf.unam.mx:8080/geoserver/ows?
63 service=WFS&request=GetCapabilities
64 Servicio WFS: http://www.pvretano.com/nga/cubeserv.cgi?CONFIG=plugweek
65 &DATASTORE=ts&SERVICE=WFS&REQUEST=GetCapabilities
66 Servicio WFS: http://protezionecivile.regione.campania.it/geoserver/ows
67 ?&SERVICE=WFS&REQUEST=GetCapabilities
68 Servicio WFS: http://server21.to.cnr.it/geoserver/wms?%3C%2Furl&SERVICE=
69 WFS&REQUEST=GetCapabilities
70 Servicio WFS: http://ows.bom.gov.au/cgi-bin/mapserver/users/bomw0473/
71 latest?&SERVICE=WFS&REQUEST=GetCapabilities
72 Servicio WFS: http://www.catais.org/geoserver/wfs?&SERVICE=WFS&REQUEST=
73 GetCapabilities
74 Servicio WFS: http://webgis.regione.sardegna.it/geoserver160/wfs?SERVICE
=WFS&REQUEST=GetCapabilities
Servicio WFS: http://gis.drcog.org/geoserver/wms/?&SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://earth.unibuc.ro/geoserver/ows?SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://www.earth.unibuc.ro/geoserver/wfs/?SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://www.earth.unibuc.ro/geoserver/wfs;?SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://donosti.geodata.es/ccvoc/ows/ccvoc_public?&SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://geoportal.invemar.org.co/cgi-bin/service?&SERVICE=WFS&REQUEST=GetCapabilities
Servicio WFS: http://nikos2.alachuacounty.us:8080/geoserver/ows?&SERVICE=WFS&REQUEST=GetCapabilities
References


REFERENCES


[jYzZkGW06] Shou jian Yu, Jing zhou Zhang, Xiao kun Ge, and Guowen Wu. Semantics based web services discovery. In PDPTA’06, pages 340–345, 2006. 28


[KL05] Eva Klien and Michael Lutz. The role of spatial relations in automating the semantic annotation of geodata. In COST7, pages 133–148, 2005. 24


[LWdHR06] Rob Lemmens, Andreas Wytikzik, Rolf de By, Carlos Granell, Michael Gould, and Peter van Oosterom. Integrating Semantic and Syntactic Descriptions to Chain Geographic Services. IEEE Internet Computing, 10(5):42–52, 2006. 43

218


REFERENCES


Declaration