Diploma Thesis

Knowledge-based search for Earth Observation products

Marius Podwyszynski
podwyszy@fim.uni-passau.de

August 4, 2009

Supervisors

Prof. Dr. Burkhard Freitag

University of Passau
Department of Computer Science and Mathematics
Chair of Information Management

Stephan Kiemle

German Aerospace Center (DLR)
German Remote Sensing Data Center (DFD)

Declaration of authorship

I certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other University.

Passau, August 4, 2009

Marius Podwyszynski

Contents

Chapter 1

Introduction

The German Aerospace Center (DLR)[1] "is Germany's national research center for aeronautics and space"[2]. The remote sensing of the earth via various satellite or airborne sensors is one of the missions of DLR and allows to retrieve interesting information about our environment. This Earth Observation (EO) data is maintained by the German Remote Sensing Data Center (DFD)[3] and supports EO application domain experts in observing, analyzing and interpreting natural and anthropogenic phenomena. Considering, for instance, the water management domain, river flood, regional desertification or vegetative deterioration can be recognized via complex data processing of raw satellite data. At the lowest level, EO data can be retrieved by *optical, radar-based, electro-magnetic spectral, lidar-based or hyper-spectral* sensors. The data heterogeneity is even emphasized by the versatile sensor modes for the measurement of physical properties. While a radar-based sensor determines precise terrain heights, an electro-magnetic spectral sensor captures temperature values of the sea surface or pressure zones in the atmosphere. The heterogeneity of the data

[1] `http://www.dlr.de`

[2] `http://www.dlr.de/en/desktopdefault.aspx/tabid-636/1065_read-1465/`

[3] `http://www.dlr.de/caf/en/desktopdefault.aspx/tabid-2536/3792_read-5686/`

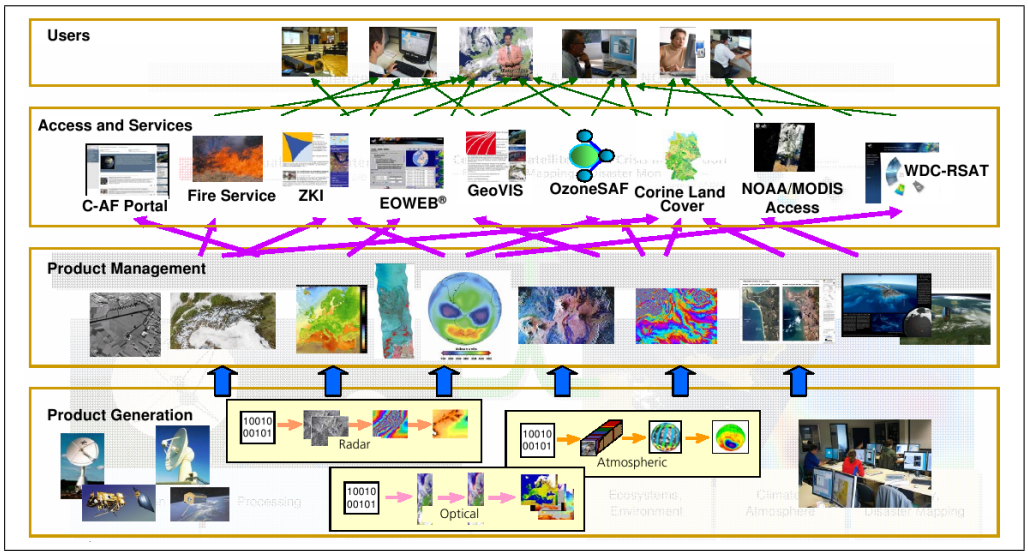

Figure 1.1: Variety of sensor data, product types and application scenarios (source: DLR)

and its applications is illustrated in figure 1.1. Application domain users from the EO domain must be able to access the EO data that is appropriate for them. The EO data can be accessed through distributed services which are maintained by the DLR (*C-AF portal, Fire Service, ZKI, EOWEB, GeoVIS, OzoneSAF, Corine Land Cover, NOAA/MODIS access, WDC-RSAT*). To maintain application independence and to support multiple usage scenarios of these services, application-specific terms and taxonomies are seldomly integrated into them. This fact often hinders application domain users from selecting the appropriate EO data as they have to learn the terminology and structure of a service before they can actually use it. By modelling the diverse semantic interdependencies between sensor characteristics and observed phenomena in so-called ontologies, it can become feasible to access EO data based on application domain terms and taxonomies. Ontologies are an "explicit specification of a conceptualization"[11] and so have the capability to store the domain knowledge of experts from various research fields in a knowledge-based,

human-readable, and machine-processable manner. Thus, an online GUI built from the contents of this ontology can be used to guide an application domain expert through the semantic web of his or her domain knowledge, while hiding the complexity of the sensor interns and the resulting EO data. By navigating through the ontology, the appropriate actions to the underlying legacy systems can be triggered to access the actual EO products which are hosted at the DFD.

This diploma thesis considers both, theoretical and practical aspects of the semantic modelling approach for the EO domain. Firstly, it briefly introduces the EO data management at the DFD (cf. chapter 2) and then continues with the problem analysis based on real-life usecases (cf. chapter 3) used as a starting point to create a common high-level information model of the EO domain. Next, the common high-level information model can be formally specified into an ontology for the EO domain that reuses common terminologies from earth sciences (cf. chapter 5). Furthermore, the user access to the developed ontology is presented in the EOLOsearch web demonstrator (cf. chapter 6). Finally, the feasibility of the semantic modelling approach for the EO domain is evaluated based on the developed ontology and its web demonstrator considering several quality criteria that apply for the EO domain (cf. chapter 7).

Chapter 2

Earth Observation (EO) data-management at the German Remote Sensing Data Center (DFD)

The storage and subsequent query of EO data obtained from a satellite sensor is a non-trivial task. "The data growth rates are challenging, and even more so the increasing diversity of data structures and formats."[17, p. 223]. At the German Remote Sensing Data Center (DFD), EO data is managed by the Data and Information Management System (DIMS) and made publically available for product ordering via the EOWEB® online interface[4]. Both architectures are illustrated in the following sections.

[4] `http://eoweb.dlr.de/`

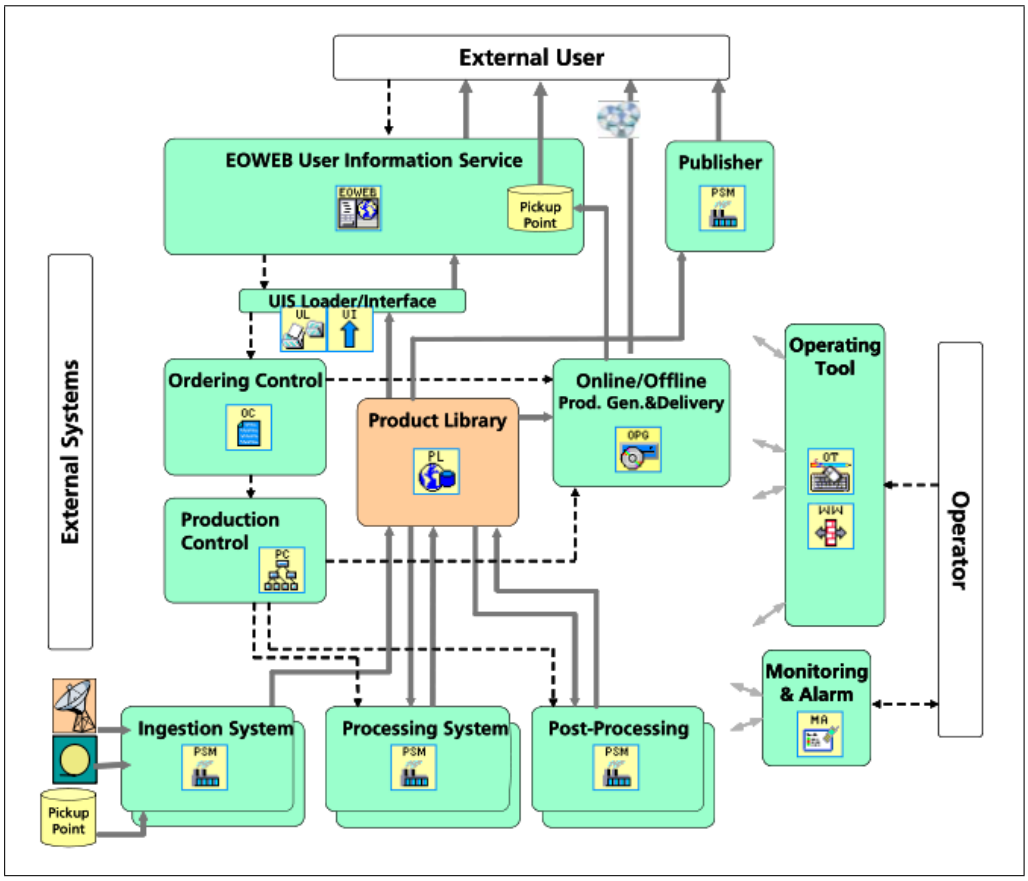

Figure 2.1: Data and Information Management System (DIMS) (source: DLR)

2.1 Data and Information Management System (DIMS)

The Data and Information Management System (DIMS) is a distributed software-system for the production, cataloging, archiving, ordering, management and distribution of Earth Observation (EO) products (see figure 2.1). The order of magnitude of the EO data that is managed by the system is several hundred Tera Bytes. Besides the huge data size, the data that is obtained from a satellite sensor is at the lowest level in binary form, that can only be understood by machines. Thus, complex processing chains subsequently create more valuable, interesting and human-

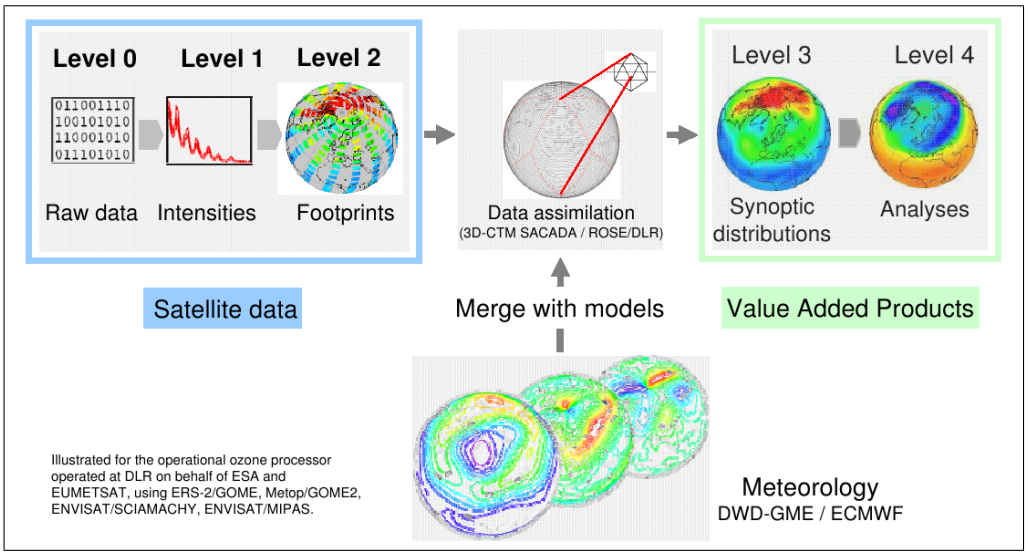

Figure 2.2: Processing chain of EO data for the Meteorology domain (source: DLR)

readable data that can then be interpreted and analyzed by various domain experts (see figure 2.2). The EO data can be accessed via various online interfaces, e.g. EOWEB®. Operators of the DLR use an Operating Tool to develop new data models for emerging EO products and to maintain existing ones. Moreover, external systems may be setup to obtain privileged access to the DIMS Product Library (PL) which is the central point for data access at the DFD. The main responsibility of the DIMS PL is the consistent long-term data archiving of EO products. The "[..] separation of metadata characterizing a product and the original primary data of the product"[16, p.61] allows for the efficient cataloging and searching for primary data by querying its corresponding metadata. "A typical earth observation data product consists of different components like the original primary data (e.g. hierarchical data format files), browse images reduced in resolution and auxiliary data useful e.g. for data interpretation"[16, p.62]. Figure 2.3 illustrates this separation for a sample EO product. The product metadata comprises of mission-based (*mission, sensor, code*),

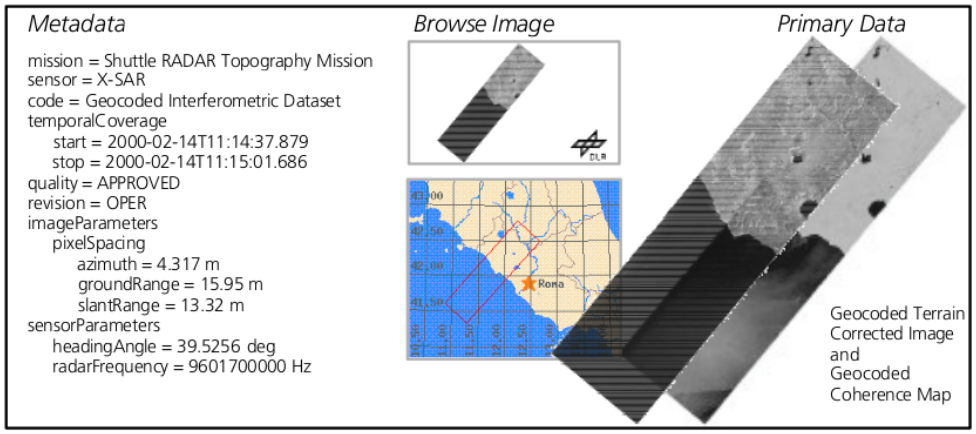

Figure 2.3: Metadata and preview image for primary data obtained from a radar sensor (source: DLR)

temporal (*temporalCoverage*), operational (*quality, revision*), data-based (*imageParameters*), and sensor-based (*sensorParameters*) parameters for the primary data. Any subset of these parameters can be employed for the search for EO products. Moreover, the metadata parameters may vary for different EO product-types substantially.

2.1.1 DataSet

A *DataSet* is the ISO-standardized term for a unique EO product with certain metadata fields which are used for product description and discovery. A *DataSet* is the smallest atomic unit for product retrieval.

2.1.2 Collection

A *Collection* is the ISO-standardized term for a series of *DataSets* which provide some extra metadata fields. Usually, all *DataSets* within a *Collection* are homogeneous and provide the same properties such as provenance, resolution and accuracy. It is also possible to create heterogeneous *Collections* with *DataSets* from various sources, however.

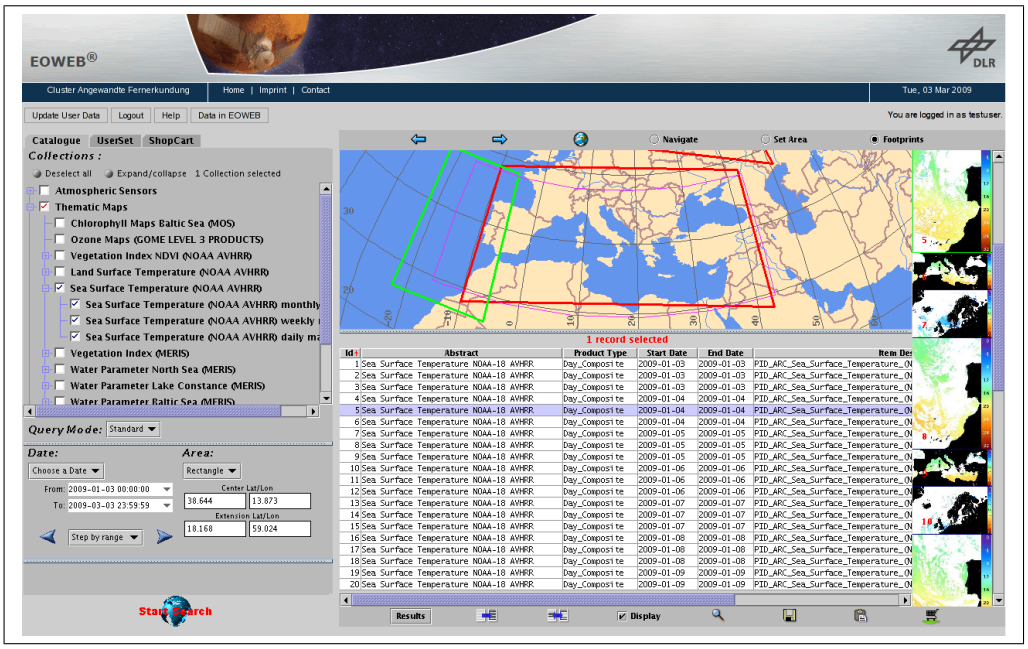

Figure 2.4: EOWEB® online interface (source: `http://eoweb.dlr.de/`)

2.2 EOWEB® Online Interface

Typically, the online EO data access for external users is performed by querying the EOWEB® online interface[5](see figure 2.4). At the DFD, EO products are grouped thematically or mission-based into *Collections* which are structured into hierarchies in the left frame. When browsing the *Collection* trees for the selection of a certain *Collection*, the user must know where to find a certain *Collection* that contains certain EO products. For instance, the *Collection* 'Sea Surface Temperature (NOAA AVHRR) monthly' is contained in the *Collection tree* 'Thematic Maps'. In the lower frame, the *Query Mode* ('Standard' or 'Advanced') can be chosen. The *Date* and *Area* define the more specific *Dataset* properties as temporal and spatial data coverage. The spatial coverage can also be set by clicking and dragging a geometric form in the world map in the right frame. Clicking

[5]`http://eoweb.dlr.de/`

on 'Start Search' at the bottom of the left frame triggers the actual search for the *DataSets* with the appropriate parameters. After a short while, all *DataSets* that match the search parameters are presented at the right frame below the world map in a list containing a subset of each primary data's metadata. In the rightmost frame, the corresponding preview images of the primary data are embedded into the results. Moreover, a detailed listing with all metadata fields can be generated. The actual primary data can be directly purchased online via EOWEB® by clicking on the shopping cart icon at the bottom of the preview frame. A direct download of data via OGC (Open Geospatial Consortium) interface is currently under development.

Chapter 3

Problem statement

This chapter explains the main problem in the focus of this diploma thesis. In the Earth Observation (EO) domain, data images from the earth are obtained by various remote sensing technologies. On the one hand, the data is maintained and published by the EO datacenter operators at the DFD. On the other hand, the data is used by EO application domain experts from the scientific community in order to perform research activities. The problem is the semantic gap which is exactly between these two worlds. Neither a datacenter operator has detailed information about how the data will be used in research nor the application domain expert is aware of which data is relevant for a specific research field. Where does this situation derive from? The main reason for the semantic interoperability of both domains is the fact that they develop independently of each other. Moreover, the focus of both working groups differs totally: most of the employees working in the datacenter are computer specialists whereas the application domain researchers are experts in biology, physics or chemistry. Each group has difficulties in interpreting expert terms from the other group and thus misinterpretations and communication problems arise. The term *the value for an algae bloom index near the Irish west coast* completely differs from the term `ENVISAT.SCIA.L2` which is the name of a collection,

for instance. To illustrate the problem further, two usecases for application domains are presented: the domain of air-pollution monitoring at the World Datacenter for Remote Sensing of the Atmosphere (WDC-RSAT) and the domain of water management within the WISDOM project for water quality management in the Vietnamese Mekong delta.

3.1 EO datacenter domain

The datacenter domain represents the established remote sensing infrastructure at the DFD which is provided by the various satellite and airborne missions. A satellite platform usually carries several sensors. A first distinction of these sensors is made upon the sensor type as being one of the most current types: *optical, radar-based, electro-magnetic spectral, lidar-based, gps-based, or hyper-spectral.* Distinct sensors usually deliver EO data for different physical properties. For instance, *electro-magnetic spectral* sensors measure temperature values or the concentration of a chemical compound, *radar* sensors render a topographic surface structure and *optical* sensors may capture images of land coverage. Moreover, sensors usually vary in their adjustable sensor modes, resolutions and accuracies. Furthermore, according to the satellite's orbit, a sensor is only able to capture EO data from specific geographic regions and temporal coverages. To deal with this complexity, employees of the DFD typically have deep knowledge about the sensor characteristics.

3.2 EO application domain

Various application domains such as earth sciences, environmental monitoring, civil security, and health care depend on EO data as a scientific base for further research, mapping and information service activities. In general, each application domain is interested in different phenomena and

physical properties that can be measured by various EO sensors. However, the application domain users have one thing in common: the relevant EO data that is needed for their activities is stored, processed, and maintained by the EO datacenter. Each application domain provides its specific application terminologies and taxonomies which are rarely used by datacenter operators to describe and structure the EO data due to multiple usage scenarios of the EO data. For instance, maps measuring the seasurface temperature can be used in oceanology or in climatic sciences. From the application user's viewpoint, the lack of the common application domain terminology and taxonomy hinders the search process for EO data. Therefore, application domain users must learn how the EO datacenter structures and names their relevant *Collections*. The following two use cases, which are researched at the DLR, illustrate the different application objectives and data requirements. These are employed in this thesis to develop a common high level information model.

3.2.1 Use case 1: Air-pollution monitoring

The DLR "hosts and operates the World Data Center for Remote Sensing of the Atmosphere (WDC-RSAT)[6] [..]. Primary focus of WDC-RSAT is to offer scientists and the general public free and simplified access [..] to a continuously growing collection of atmosphere-related satellite-based data sets and services"[7]. For instance, the "Integrated Air Quality platform"[8] provides information about air pollutant concentrations of ozone, nitrogen dioxide and particulate matter (see figure 3.1). The chemical compounds are measured by *electro-magnetic spectral* sensors. High-level EO data is created out of the raw data by sophisticated processing systems at the DLR and used in various research areas. The graphic products are of "high interest for the

[6] `http://wdc.dlr.de`

[7] `http://wdc.dlr.de/about/index.php`

[8] `http://wdc.dlr.de/data_products/projects/promote/IAQ/`

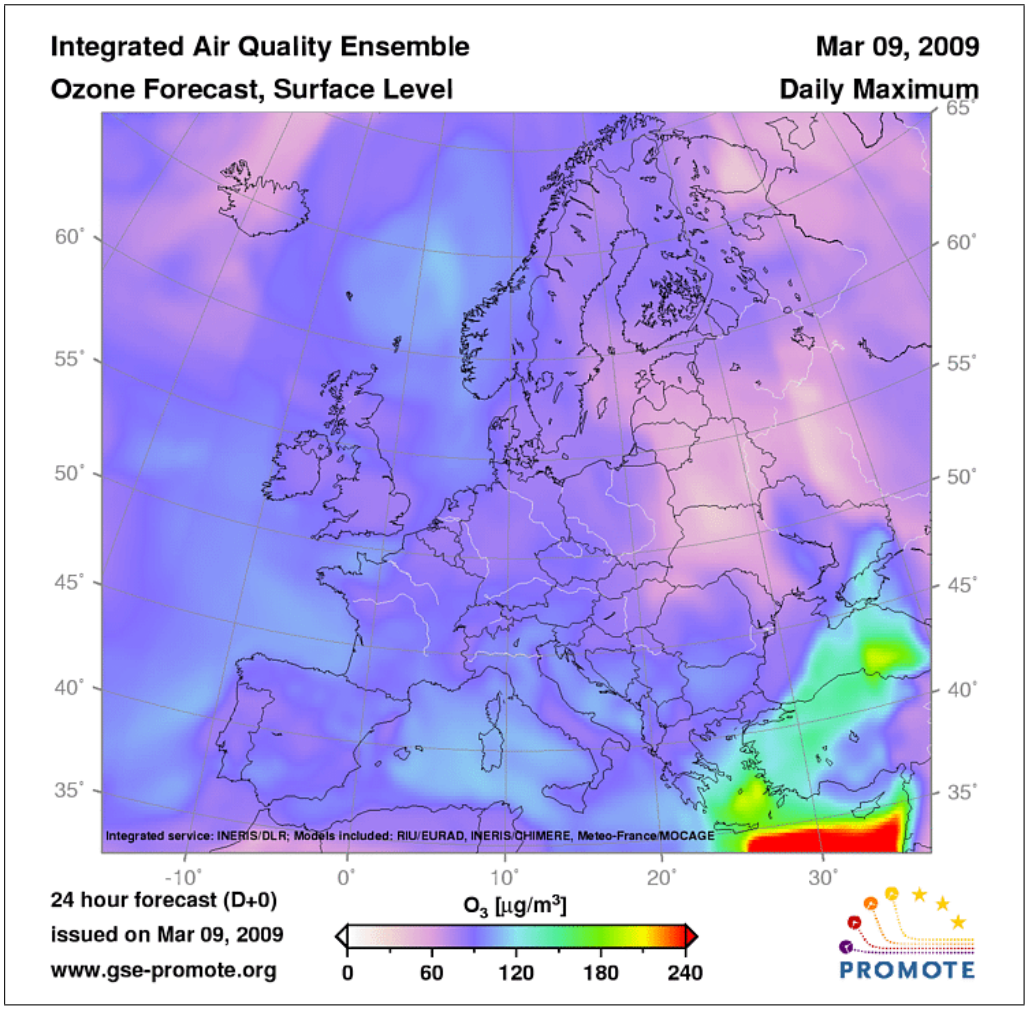

Figure 3.1: Integrated Air Quality Ensemble forecast for Europe (source: DLR)

authorities in charge of air quality management"[9]. The "Bavaria pilot"[10] project, for example, correlates meteorological and air quality EO data with medical data to reveal possible causes of negative health effects. Application users working at the WDC-RSAT, need EO data measuring chemical compounds such as *O3, CO2, NO2, SO2* in the various atmospheric layers *Stratosphere, Troposphere, Mesosphere, Thermosphere* for their daily activities. An excerpt of the relevant *Collections* for the WDC-RSAT domain are the following: `ENVISAT.AATSR, ENVISAT.GOMOS.L2, ENVISAT.MERIS.L1, ENVISAT.MIPAS.AUX, METOP.GOME2.TC.L1B`, and `Vertical-Column-Density`. The names are encoded by the EO datacenter whereas its relevance can hardly be derived from them.

3.2.2 Use case 2: WISDOM - Water management in the Mekong Delta

The Water related Information System for the Sustainable Development of the Mekong Delta Vietnam (WISDOM)[11] project has been started in 2005 due to the high population growth, climatic change and agricultural importance of the Mekong Delta in Vietnam in order to manage the available water resources. "It is the goal of WISDOM to jointly (Vietnamese and German partners) design and implement an Information System for the Mekong Delta, containing information from the fields of hydrology, sociology, information technology and earth observation. The integration of such data will enable the end-user of the system to perform analysis on very specific questions; and thus will supply the end-user with a tool supporting regional planning activities."[12]. Flood monitoring, prediction and damage potential,

[9] `http://wdc.dlr.de/data_products/promote/IAQ/iaqp_product_specification.pdf`

[10] source WDC

[11] `http://www.wisdom.caf.dlr.de`

[12] `http://www.wisdom.caf.dlr.de/intro/objectives_en.html`

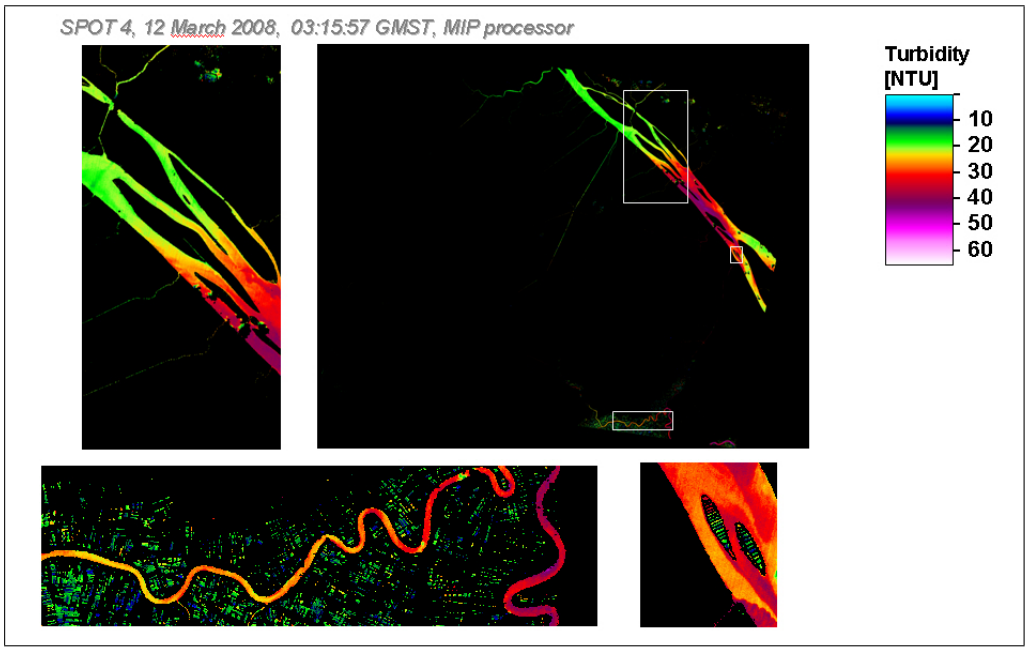

Figure 3.2: Turbidity in [NTU] derived from SPOT4 2008 (source: DLR)

the monitoring of water quality, pollution and sediment concentration or the monitoring of the population growth are all among these activities. Many of the mentioned activities can be realized by obtaining the appropriate EO data that is archived at the DFD. For instance, flood monitoring and damage potential can be calculated with the delta of two *radar* images, one that was taken just before the flood and one of the current situation. Water sediment concentration has causes on the water turbidity and can be visualized by processing low level EO data that was initially recorded by an *electro-magnetic spectral* or *hyper-spectral* sensor with the spectral band for vegetation observations into high level EO data. The images in figure 3.2 have been created by "algorithms based on physical inversion schemes that derive biophysical parameters from the measured radiance signal at the sensor"[13]. The application domain user is interested in EO data measuring

[13] `http://www.wisdom.caf.dlr.de/results/remote/2009-02-16_Water_Quality_en.html`

vegetative deterioration, flood or *water pollution* for the input parameters of the mentioned processing algorithms. Some relevant *Collections* for this application domain are: `AQUA.MODIS`, `GLOBE.DEM`, `NOAA.AVHRR.NDVI`, `TSX-1.SAR.Imaging`, and `TSX-1.SAR.L1b-Stripmap`. Similar to usecase 1, their relevance for the application domain can not be derived from the naming of the *Collections*.

3.3 Semantic gap between the datacenter domain and the application domain

An application domain expert always wants to employ the best available EO data, that is, the data that best describes the observations from the research field he or she is working on. In the datacenter domain, a special physical observation is recorded by a specific sensor attached to a satellite. Consequently, there seems to exist a strong interrelation between the datacenter domain and the application domains based on the physical observation. However, from the application domain expert's point of view, detailed a priori knowledge about the datacenter domain is required to select the appropriate EO *Collections* which meet the expert's requirements. Conversely, from the datacenter operator's point of view, it is difficult to relate the available EO *Collections* to application domains without querying the corresponding researchers in advance. Hence, the datacenter domain and the application domains consider two facets of the same problem that is identified as the *semantic gap between the datacenter domain and the application domain*. This problem occurs every time when an expert crosses his or her competency boundary by implying an inter-domain relationship that requires expertise in the targeting domain as well. Both, an application domain expert working at the DLR as well as an external user, will have similar difficulties in learning the relation between the

sensor characteristics and the EO data appropriate for their research fields. From the viewpoint of a DFD operator working in the datacenter domain, similar competency problems arise when externalizing *Collections* for certain application domains. The multiple usage scenarios for a *Collection* hinder the development of an universal *Collection* hierarchy and naming that is appropriate and understandable by all application domains. In addition, the application domains and the datacenter domain are developing with different velocities. It is quite difficult for application domain users to stay up-to-date about the current configuration and capabilities at the datacenter. Furthermore, datacenter operators cannot be informed about every development of the various application domains which exist in science. The lack of a formalized communication model can be seen as a possible reason for these informational assymetries.

A solution approach, with both theoretical and practical perspectives, will be discussed in the following chapters 5 and 6.

Chapter 4

Semantic modelling

Speaking of the "Semantic Web"[6], people usually have different interpretations in mind. One might think of a social bookmarking website like *delicious*[14], others might have a friend networking platform like *facebook*[15] or the *amazon* online shop[16] in mind. What do all of these platforms have in common? It is their usage of semantic modelling for knowledge representation. "The Semantic Web standards have been created [..] as a medium in which people can collaborate on models"[2, p. 15]. But why is such collaboration necessary? The main benefit of a semantic model is that it can provide "[..] a framework [..] for representing and organizing commonality and variability of viewpoints [..]"[2, p. 26]. Moreover, "models help people communicate"[2, p. 15] and build an infrastructure of common knowledge which is semantically enriched by the modelling community. So the Semantic Web improves the communication among people with diverse knowledge domains in a systematic way. But not only people can communicate more efficiently with their knowledge modelled into knowledge bases. By specifying this knowledge in formal representation languages

[14] `http://delicious.com`
[15] `http://www.facebook.com`
[16] `http://www.amazon.com`

such as Resource Description Framework (RDF), RDF Schema (RDFS), and Web Ontology Language (OWL), the Semantic Web enables even more intelligent machine-to-machine interaction based on human semantics of a specific domain of interest.

4.1 Information representation

In information representation, several modelling levels enrich the semantic modelling of an area of interest, continuously. At the lowest level, a vocabulary of terms is defined that best describes the considered concepts. "A vocabulary is a collection of unambiguously defined terms used in communication."[13, p.99] It represents a commonality of terms that is agreed upon by different stakeholders and forms the basis for any deeper semantic modelling of the area of interest. When the vocabulary of terms is created, a taxonomy can be constructed "[..] in which terms are organized in a hierarchical manner."[13, p.99] The basic relations between the terms that are employed in this way are specialization and generalization. In addition, an ontology can be defined upon the vocabulary and taxonomy to "[..] define concepts and the relationships between them [..]"[13, p.99]. In contrast to taxononomies or thesauri, ontologies do not impose limitations on the relations that can be modelled between concepts.

4.2 Languages for information representation

The previous section introduced various levels of information representation possibilities in semantic modelling. This section concentrates on the description languages developed by the World Wide Web Consortium (W3C)[17] for obtaining a standardized and interoperable scheme for the formalization of the information representations (see figure 4.1). The

[17] `http://www.w3.org`

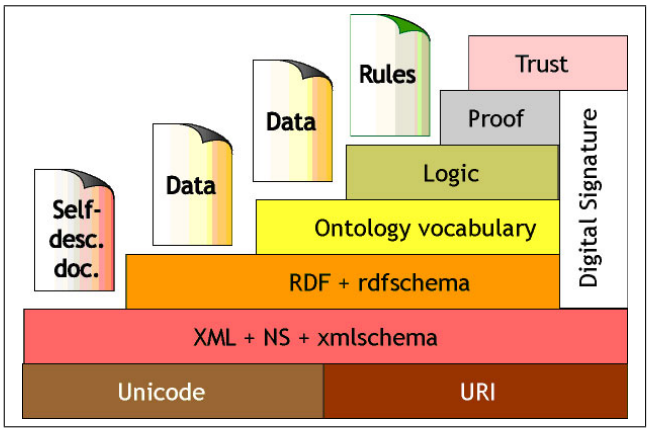

Figure 4.1: The Semantic Web Stack (source: `http://www.semantic-conference.com`

syntactic interoperability is introduced with support for *Unicode* encodings and *URIs* at the lowest layer. The next layer contains the *eXtensible Markup Language (XML)*[21] with *namespaces* and *xmlschema* which defines XML datatypes. On top of XML, the *Resource Description Framework (RDF)*, a domain independent language for representing data, is placed as a first semantic layer, because "[..] any intended semantics are outside the realm of the XML specification"[9, p. 13]. Furthermore, *RDF Schema (RDFS)* "defines the vocabulary used in RDF data models"[4, p. 66]. On top of RDF and RDFS, *ontology vocabularies* and their *logic* reside. The Web Ontology Language (OWL) is the "proposed standard for Web ontologies"[4, p. 152]. One layer above, semantic *proof* can be achieved by applying logical rules. Description Logic (DL) reasoners implement this layer. Finally, *Semantic Trust* is located at the top of the Semantic Web Stack and represents the ideal of the Semantic Web. The mentioned description languages XML, RDF, RDFS, and OWL will be outlined in the following sections.

4.2.1 URI

Unified Resource Identifiers (URI) are "[..] a standardized way of naming resources [..]"[13, p. 70] that "[..] provide a unique name for items contained in a statement across the entire Internet."[13, p. 11]. Thus, the URI `http://www.dlr.de/ontologies/EOLO.owl#GPSCollection` can be used to universally address this resource. Therefore, URIs "provide a foundation for data-sharing infrastructures"[13, p.70].

4.2.2 Namespaces

Namespaces are part of URIs and describe the scope of the uniqueness of a resource. Considering the namespace `xmlns:rdf=‘‘http://www.w3.org/1999/02/22-rdf-syntax-ns#’’`, the resource `rdf:type` specifies a schema for type inheritance. Every resource which is referred by this URI must be interpreted as type inheritance. A different resource *type* with a different meaning could coexist only within a differing namespace, i.e. `other:type`.

4.2.3 Extensible Markup Language (XML)

The eXtensible Markup Language (XML) is a text-based description language that "can provide an effective solution to the syntax problem for data sharing."[13, p.67] The interoperability at the syntactic level only shifts the problem to a higher level, however. Specifically, "XML does not provide any means of talking about the semantics (meaning) of data."[4, p.65] Therefore, several other description languages like RDF, RDFS, and OWL have been defined to formally represent the data semantics.

4.2.4 Resource Description Framework (RDF)

The Resource Description Framework (RDF) "is a W3C recommendation for the notation of metadata on the World Wide Web"[7, p.197] that is domain-

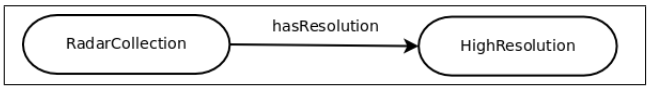

Figure 4.2: RDF graph

independent and allows for the definition of binary relationships [4, p. 109]. "RDF is designed to provide a basic object-attribute-value model for Web data"[7, p. 200]. Such a triple is called a RDF statement. For instance, an object *O* has an attribute *A* which takes a specific value *V*. In RDF, a value *V* can also play the role of an object *O'* in another statement. RDF also provides the possibility that a RDF statement itself can be object or value of another RDF statement which is referred to as "reification"[2, p. 49]. The graph in figure 4.2 and its RDF/XML syntax in listing 4.1 illustrate a triple with the object `RadarCollection`, the attribute `hasResolution` and the value `HighResolution`.

```
1 <rdf:Description rdf:about="RadarCollection">
2         <hasResolution rdf:resource="HighResolution"/>
3 </rdf:Description>
```

Listing 4.1: RDF/XML syntax

4.2.5 Resource Description Framework Schema (RDFS)

```
1 <rdfs:Class rdf:about="Collection"/>
2
3 <rdfs:Class rdf:about="RadarCollection">
4         <rdfs:subClassOf rdf:resource="#Collection"/>
5 </rdfs:Class>
```

Listing 4.2: A `RadarCollection` is a subclass of a `Collection`

RDF Schema takes a further step into a deeper semantic modelling as "[..] the only difference between RDF Schema expressions and 'normal' RDF

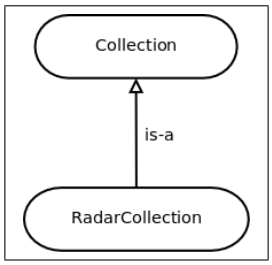

Figure 4.3: Graphical representation of a sample type hierarchy for Class `Collection`

expressions is that in RDF Schema an agreement is made on the semantics of certain terms and thus on the interpretation of certain statements"[7, p.200]. With RDFS, many new semantic concepts like classes, subclasses, subproperties, domain and range restrictions help to enrich the semantics of the model [9, cf. p.15]. A sample type hierarchy that can be modelled in RDFS for the class `Collection` is illustrated in listing 4.2 and in figure 4.3. It can be observed, that the RDFS syntax conforms to the RDF syntax. However, "RDF Schema is not rich enough to express inconsistencies."[4, p. 228]

4.2.6 Web Ontology Language (OWL)

RDFS has introduced the very powerful modelling primitives *class, subclass, subproperty, domain and range restrictions*, and *individuals* of a *class*[4, cf. p. 112]. However, some situations cannot be expressed with RDFS and require a further abstraction layer. "OWL is the World Wide Web Consortium (W3C) recommended ontology language for the Semantic Web, and exploits many of the strengths of Description Logics, including well defined semantics and practical reasoning techniques"[15, p. 459]. OWL enables the modelling of the "local scope of properties [..], disjointness of classes [..], boolean combinations of classes [..], cardinality restrictions [..] (and) special (mathematical) characteristics of properties"[4, p.111]. The

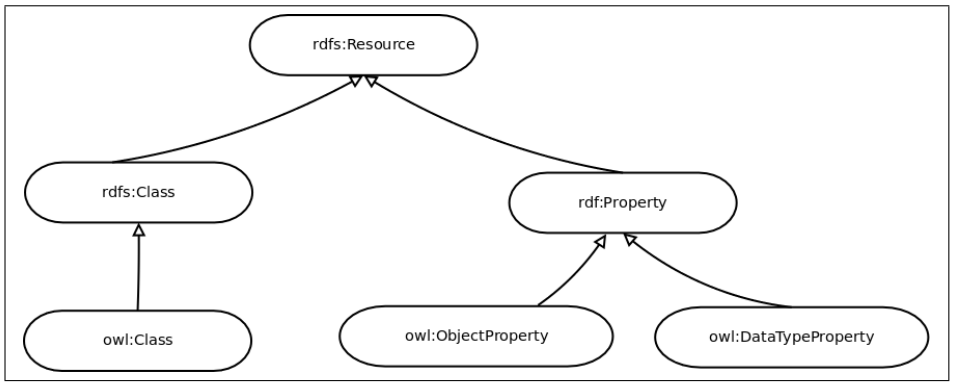

Figure 4.4: OWL primitives

main OWL modelling primitives are `owl:Class, owl:DataTypeProperty` and `owl:ObjectProperty` (see figure 4.4). OWL is divided into the sublanguages OWL-Lite, OWL-DL and OWL-Full with different expressive power and compatibility with RDF [15, p. 467]. All OWL sublanguages use RDF for their syntax. For the sake of completeness, the OWL sublanguages will be shortly described.

4.2.6.1 OWL-Full

OWL-Full is the most expressive sublanguage as no restrictions on the usage of the OWL primitives are imposed through the language. It "allows to combine these primitives in arbitrary ways with RDF and RDF Schema"[4, p. 112]. The high expressiveness and the compatibility with RDF has a major drawback: OWL-Full is undecidable and thus unsuitable for efficient reasoning[4, p. 113].

4.2.6.2 OWL-DL

OWL-DL restricts the usage of the OWL and RDF primitives to "regain computational efficiency"[4, p. 113]. The sublanguage is based on a well-studied Description Logic which is explained in subsection 4.2.7.

4.2.6.3 OWL-Lite

OWL-Lite sets even more restrictions on the usage of modelling primitives in favor of users and tool builders. However, this approach leads to less expressive modelling capabilities[4, p. 113].

4.2.7 Description Logics (DL)

Description Logics (DL) are "[..] a formalism to represent knowledge [..]"[18, p.1]. "A characteristic feature of Description Logics is their ability to represent other kinds of relationships that can hold between concepts beyond IS-A relationships."[18, p.5] Furthermore, Description Logics describe "knowledge representation (KR)"[5, p. 47] systems that provide "facilities to set up knowledge bases, to reason about their content, and to manipulate them."[5, p.50] A knowledge base typically comprises two parts: the "TBox and the ABox"[5, p. 50] which refer to the "terminology"[5, p. 50] and the "assertions"[5, p. 50] of the domain of interest, respectively. However, a "DL system not only stores terminologies and assertions, but also offers services that reason about them."[5, p. 51] Such reasoning can be employed to infer new knowledge about a domain of interest.

4.3 Inferencing by Reasoning

Information systems like databases typically store the actual facts about the area of interest. [19, cf. p. 91] Knowledge bases go a step further by archiving "[..] certain pieces of knowledge such as rules[..]"[19, p. 91] which are used to infer implicit knowledge about the area of interest. "In the context of the Semantic Web, inferencing simply means that given some stated information, we can determine other, related information that we can also consider as if it had been stated"[2, p. 80].

4.3.1 From information to knowledge

Information is rapidly growing and the explicit formalization of its attributes and relationships is a very time consuming task, especially for large databases systems. According to [19, p. 3], information is referred to as "data equipped with meaning". Beyond this, knowledge is determined as the "whole body of data and information that people bring to bear to practical use in action, in order to carry out tasks and create new information." A less abstract definition of knowledge is "[..] complex information, typically telling us something about other information."[19, p. 86] Mapping these definitions to the EO domain, *Collections* can be interpreted as information which explicitly define certain attributes. Connecting this information with other information is the basis for high-level knowledge, for instance the capability of monitoring a natural or anthropogenic phenomenon.

4.3.2 Knowledge models

Besides the difference between information and knowledge, "[..] one can see a clear distinction between intensional knowledge, or general knowledge about the problem domain and extensional knowledge, which is specific to a particular problem."[18, p.13]. While the "TBox contains intensional knowledge in the form of a terminology [..] the ABox contains extensional knowledge - also called assertional knowledge [..]"[18, p.14]. Furthermore, it is possible to obtain other knowledge, than the previously stated one. Such implicit knowledge can be identified by establishing and applying a set of rules that compose the knowledge facts. This is the main task in obtaining knowledge from an information system and is referred to as reasoning by inferencing. The rules which have generated the new knowledge can be stored persistently in a knowledge base system. So the new knowledge can be inferred repeatedly and adapt to the dynamic evolution of the knowledge base. Inferred knowledge is not defined explicitly, but is generated after

evaluating the TBox and ABox knowledge models and its relations, and thus reflects their logical consequences. Inferred knowledge for the TBox is acquired by "determining subsumption (which) is the problem of checking whether the concept denoted by D (the subsumer) is considered more general than the one denoted by C (the subsumee)"[18, p. 9-10] for the expression $C \sqsubseteq D$. Reasoning for the knowledge contained in the ABox "verifies whether a given individual is an instance of (belongs to) a specific concept."[18, p. 16]

Chapter 5

Solution statement - bridging the semantic gap in the EO domain

The semantic gap in the EO domain addressed by this diploma thesis has been defined in chapter 3. This chapter covers the steps that are taken for bridging this gap. The following sections concern the design of a high-level information model which is the base for a formalized ontology for the EO domain. In the development of this ontology, the usage of existing concepts from thesauri, taxonomies and ontologies for the earth sciences provide common terminologies which are as well plausible and effectively useful for the EO domain.

5.1 Knowledge acquisition

In the first phase, both obvious and tacit knowledge of the EO domain have to be acquired from several information resources. In the scope of this diploma thesis, mainly EO-related specifications, existing earth science knowledge bases, and discussion sessions with DFD employees have been

used to create and evolve a common high-level information model for the EO domain. More comprehensive knowledge acquisition techniques, such as "Competency Questions"[12] are discussed in chapter 7.4.1, but were not applied in this diploma thesis. Knowledge acquisition is an iterative process establishing valuable input for the development of a high-level information model that describes the EO domain.

5.2 Reusing existing conceptualizations

Knowledge models help people to communicate, and formalized models accelerate the knowledge exchange and reuse. Therefore, it is reasonable to reuse existing concepts defined by expert groups and application communities around the world in order to improve the collaboration instead of reinventing the wheel again and again. While the granularity of the knowledge models is in certain knowledge areas very fine-grained, other areas are eventually modelled too coarse, yet. However, a coarse model can be taken as a good starting point that can be refined by domain experts. There are several concept hierarchies for the earth sciences, aerospace science and geographic places from various stakeholders containing relevant concepts and terminologies for the EO domain. Whether a concept from these information sources should be reused or not depends on its semantics and on the relevance for the application domain. In order to improve readability and to avoid ambiguous concept names between the independent hierarchies, each reused concept hierarchy is identified by a unique namespace prefix that references its actual namespace.

5.2.1 Compatibility issues

An ontology provides a default namespace given by its base URI. For instance, the namespace for the resources in the ontology

located at `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl` is `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl` if not otherwise stated in the ontology. When reusing existing concepts from other ontologies or even refining them, it must be carefully evaluated whether the refined concepts should be re-integrated into the initial ontologies or loosely coupled by references in a self-standing ontology. In the first case, the refined concepts must be accepted by the model stakeholders with consensus about the relevance and quality of the new concepts. Many existing concept hierarchies are based on standardizations and common viewpoints on specific topics and are agreed upon by a wide research community. Thus, the re-integration of a refinement must be accepted by the community that created the original concept hierarchy in the first place. Generally, this approach should only be taken if the refinements have considerable importance for third-party research activities, as well, and if there is enough project time to await the acceptance of the inclusion by the concept stakeholders. In the latter case, the concepts are imported into a new ontology and one can define a namespace prefix abbreviation for the imported ontology namespace, e.g. `atmoWind` for the ontology `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl`. For references to a resource `DesertWind` within the referenced ontology, one can formulate the URI as `atmoWind:DesertWind` instead of `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl#DesertWind`. Refinements can be placed in relation to the imported concept `atmoWind:DesertWind`. However, this local modelling flexibility implies eventual synchronizations between both ontologies, the imported one and the newly created one as refinements in the ontology could possibly intersect with a refinement that was created at a later moment in the imported ontology so that a merge of both ontologies becomes necessary. In this diploma thesis the second approach is taken due to pragmatic and

temporal reasons. The resources are imported from the relevant ontologies and referenced via unique namespace prefixes defined in the ontology.

5.2.2 An authoritative ontology for earth science: Semantic Web for Earth and Environmental Terminology (SWEET)

The Semantic Web for Earth and Environmental Terminology (SWEET)[18] ontology developed by the NASA is a large ontology for conceptualizations in the earth sciences. The ontology models several hundreds of OWL classes, properties, and individuals to represent several natural or anthropogenic phenomena, processes and objects that exist on earth. For instance, the URI `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl` provides a comprehensive ontology for atmospheric winds which are observed in meteorology. A small excerpt of the `Wind` concept is illustrated in figure 5.1. The current SWEET ontology 2.0 Beta, which has been used in this diploma thesis, covers the following areas: astronomy, atmosphere, biology, chemistry, data storage, geography, geology, heliology, human sciences, hydrology, mathematics, oceanology, physics, scientific systems, as well as space and time sciences. The current version is taken as a starting point for the development of an ontology for the EO domain. The structure of the predefined SWEET concepts has influenced the development of models for the EO datacenter operators, the EO application domain experts, and the modelled bridge between both worlds. The main motivation for the reuse of some concepts from the SWEET ontology lies in the common terminology and semantics for earth sciences which are clearly defined and accepted among various research communities that contribute and agree upon the modelled concepts in the SWEET ontology. However, some classes and properties are modelled too coarse in the SWEET

[18] `http://sweet.jpl.nasa.gov`, last accessed 31 July, 2009

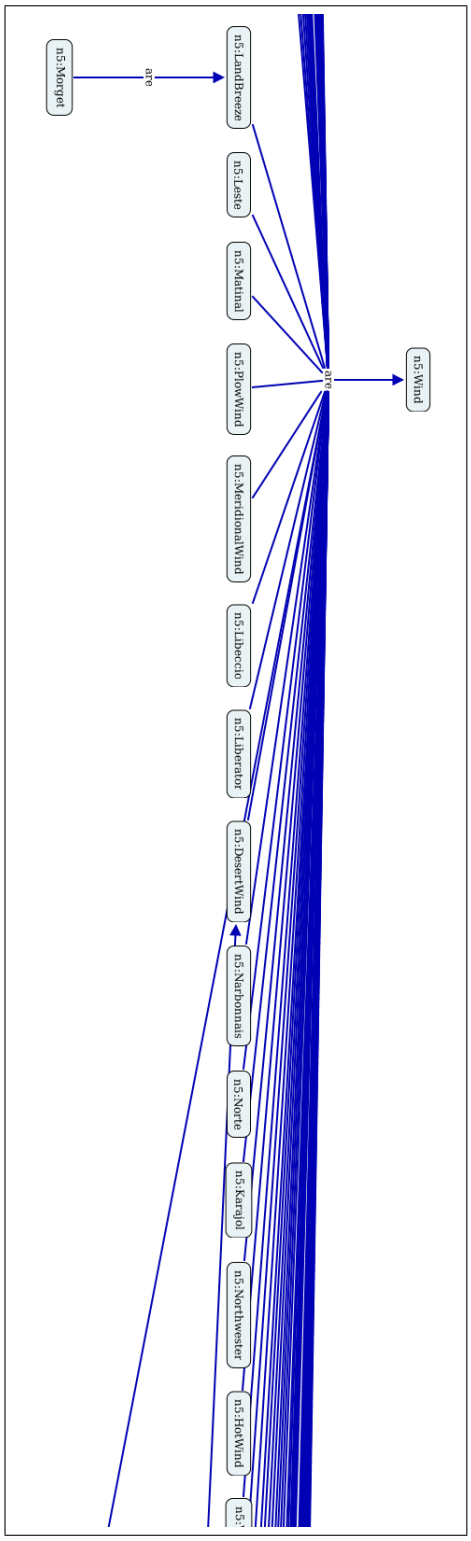

Figure 5.1: Excerpt of the concept hierarchy from the SWEET ontology `http://sweet.jpl.nasa.gov/2.0/atmoWind.owl`

ontology to solve the problem of this thesis out of the box. Therefore, the ontology for the EO domain refines the concepts from SWEET by defining new properties between existing SWEET concepts. For example, the processing level for collections is represented by the SWEET concept `http://sweet.jpl.nasa.gov/2.0/data.owl#ProcessingLevel`, with its abbreviated term `data:ProcessingLevel`, but the semantics that a collection has a specific processing level is not contained in the SWEET ontology, yet. Thus, the property `hasProcessingLevel`, defining the relationship between a collection and its processing level formally must be modelled into the EO domain ontology explicitly in order to represent this circumstance.

5.2.3 Obtaining geographical information from gazetteers

There are several gazetteers for geographical information that provide online access to their data, for instance, the Getty Thesaurus of Geographic Names® (TGN) Online[19] and GeoNames.[20] While the TGN is a "structured vocabulary that can be used to improve access to information about art, architecture, and material culture"[21] and contains approximately 1,115,000 names and other information about places around the world, GeoNames is a "geographical database (that) covers all countries and contains over eight million placenames that are available for download free of charge."[22] Figure 5.2 illustrates the geographic information contained in GeoNames for the city of Passau, Germany. Initially, it seemed quite reasonable to reuse such comprehensive geographic information in the EO ontology to integrate geographic locations from existing sources. However,

[19] Getty website, `http://www.getty.edu/research/conducting_research/vocabularies/tgn/`, last accessed: 06/06/2009

[20] GeoNames website, `http://www.geonames.org/`, last accessed: 06/06/2009

[21] Getty website, `http://www.getty.edu/research/conducting_research/vocabularies/tgn/about.html`, last accessed: June 6, 2009

[22] GeoNames website, `http://www.geonames.org/`, last accessed: 06/06/2009

"many records in TGN include coordinates [..] (which) are approximate and are intended for reference only."[23]. Considering GeoNames, the gazetteer strictly encodes all available places relative to the World Geodetic System 84 (WGS84) coordinate reference system which serves also as base for the Global Positioning System (GPS). There are several other reference systems more suitable for cartography at specific geographic regions. For instance, images of small islands will be encoded in a more precise coordinate reference system than images of continents. Moreover, more accurate global navigation satellite systems like Galileo will use a more precise coordinate reference system than GPS does. Thus, any geographic information contained in GeoNames will have to be recalculated relative to the desired coordinate reference system other than WGS84. Furthermore, in many cases the occurrence of observed phenomena is not bound to a specific geographic place, but is spanned over a flexible area of geographical locations. Consequently, the gazetteers' information serves as a good reference, but some extra effort will be needed to cover the exact geographic location of a phenomenon. Regardless where the geographic information derives from one question remains: Should such detailed geographic information be modelled into the ontology? Geographic operations such as intersection, union or area computation can be performed by Geo Information System (GIS) databases more efficiently than with Semantic Web technologies using ontologies. Therefore, the explicit modelling of geographical locations into the ontology seems not to be a reasonable approach and instead an abstraction from the concrete geographical information is modelled as a geographical extent which defines the spatial extent in two dimensions and has no geographical location on earth associated to it (cf. sections 5.4.1.5 and 7.2). For instance, the geographical information describing the concrete city *Munich* can be

[23]Getty website, `http://www.getty.edu/research/conducting_research/vocabularies/tgn/about.html`, last accessed: 06/06/2009

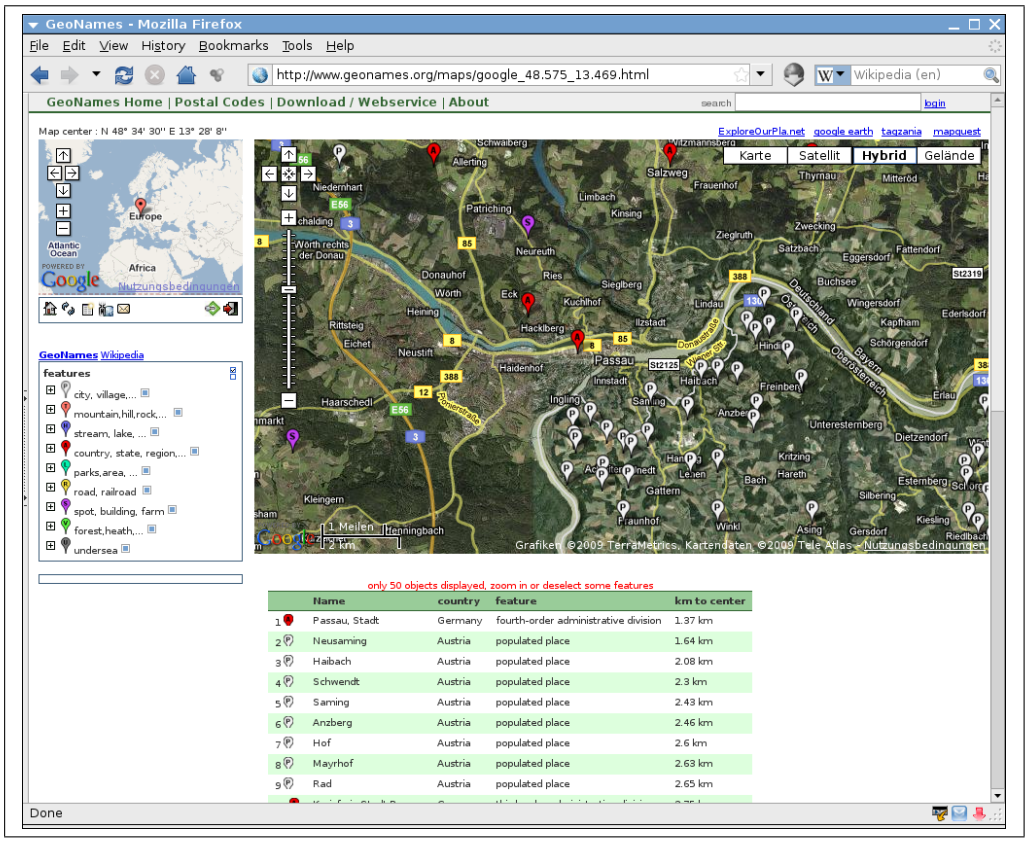

Figure 5.2: Geographic information about the city Passau in GeoNames

abstracted to the geographical extents *Metropolitan* and *Regional* whereas the concrete geographical information for the country *Germany* is abstracted to the geographical extents *Regional* and *Administrative*.

5.3 High-level information model for the EO domain

The high-level information model illustrates an abstract yet describing view of the EO world. It connects the two worlds of the EO application domain experts and the EO datacenter operators into a common model and is depicted in figure 5.3. It serves as a starting point for more formalized

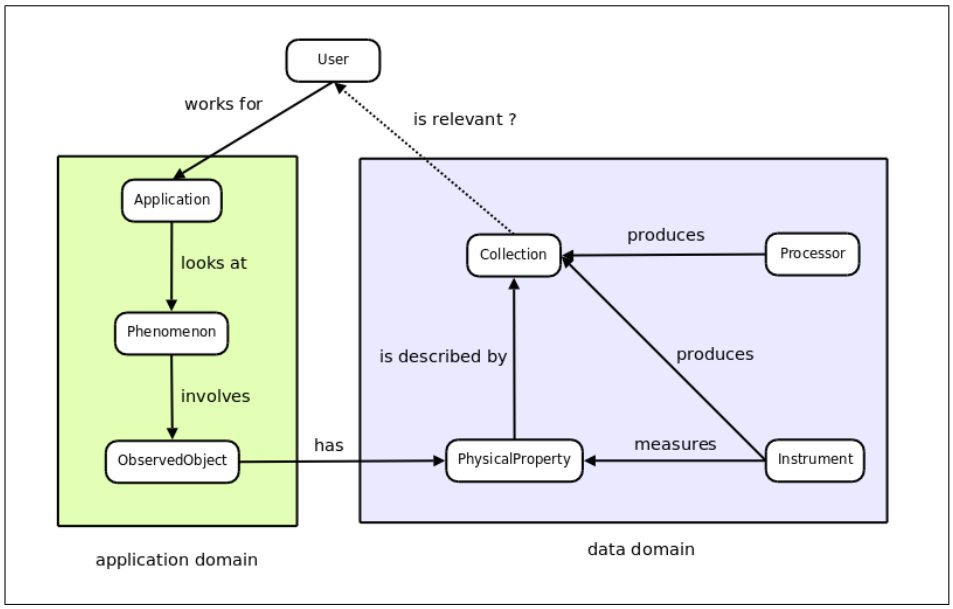

Figure 5.3: High-level information model for the EO domain

domain modelling in section 5.4. The information model illustrates the central knowledge facets and their relationships that suggest ways to integrate application domains with application data. Starting from the top, a *User* works for an *Application* (e.g. water management, crisis information). Each application looks at one or several *Phenomena* (e.g. hurricane, flood, ozone hole). Such phenomena involve *ObservedObjects* which define the objects where the phenomenon is observed (e.g. sea surface, upper atmosphere, land surface). Such objects have certain *PhysicalProperties* (e.g. pressure, temperature, spectral albedo) that can be measured by *Instruments* (e.g. electro-magnetic spectrometer, radar, lidar). Moreover, the physical properties are described by *Collections* (e.g. ENVISAT.SCIA.L2, TSX-1.SAR.L1B) which are produced either by the *instruments* or by *Processors* which create higher-level collections by performing geographic corrections and spatial and temporal intersections of several collections. Any model that is developed further on, must answer the question about the relevance of a collection for a user.

5.4 Earth Observation Lightweight Ontology (EOLO)

The high-level information model from the previous section is formalized into the *Earth Observation Lightweight Ontology (EOLO)* and encoded into the Web Ontology Language (OWL). Its building blocks are *OWL classes*, *object properties* or *datatype properties*, *OWL individuals*, and *axioms*. OWL classes represent the diverse concepts of the EO domain and are structured as separate taxonomies. OWL individuals are instances of classes with specific characteristics. The collection `ENVISAT.SCIA.L2`, for instance, is modelled as an individual of the class `EMSpectroMeterCollection`. It can be specified that collections which are obtained from a specific sensortype with a spatial and temporal extent have the ability to measure a specific physical property at an observable object. These definitions pave the path for knowledge inferences about the relevance of collections for the observation of a specific phenomenon which is researched by various application domain experts. Meaningful combinations of the mentioned concepts describe the domain of interest in both ways, syntactically and semantically. Individuals take part in the relations between classes which are defined as so-called *properties* which cross-link the orthogonal concept hierarchies. Property restrictions like *domain* and *range* characterize the property more precisely by defining which classes are the preimage and image objects for the property. Properties can be *Object Properties* or *Datatype Properties*. While the first ones have a RDFS class in their range restriction, the latter ones provide XML Schema Document (XSD) datatype definitions for their range. For instance, the property `hasEOWEBurl` that relates a `data:Collection` with its link in EOWEB, has the *domain* class `data:Collection` and the *range* XSD datatype `xsd:string`. Contrary to this, the property `hasResolution` defines the relation between a `data:Collection` and an

abstracted form of a spatial `Resolution` (drawn as edge from the *domain* class `data:Collection` to the *range* class `Resolution` in figure 5.5). Such *domain* and *range* restrictions are used in reasoning to infer class membership based on subsumption. For instance, when an OWL class fills the property `hasResolution`, the reasoner can infer that the type of this class is `Resolution`. Besides these qualitative restrictions, properties can also have quantitative multiplicity restrictions. Continuing the example, it can be defined that the number of individuals from the *range* class that take part in the relation is exactly 1, i.e. the `hasResolution` property is functional and has exactly 1 `Resolution` individual associated to it. Furthermore, condition axioms that can be declared for *partial* ($\sqsubseteq$) or *defined* ($\equiv$) classes with boolean conditions are the base for class-based reasoning. Partial classes impose *necessary* conditions on individuals, i.e. if an individual has the type of the partial class, it must satisfy necessarily the class condition. The semantics of partial classes can be interpreted as an implication from class membership to class condition satisfiability. In contrast to this, a defined class imposes *necessary and sufficient* conditions on individuals by adding the condition to the partial class condition, that it is sufficient to infer that an individual has the type of a class if it satisfies the class conditions. Thus, the semantics for defined classes are interpreted as an implication in both directions, i.e. an equivalence between class membership and class condition satisfiability. The difference between partial and defined classes is illustrated in figure 5.4. It must be stated that some parts of the model are more relevant for the EO datacenter operators while other parts are tailored to EO application domain experts. In addition, there is a common part that is relevant for both groups (see figure 5.5). The following sections contain detailed explanations of the modelled classes.

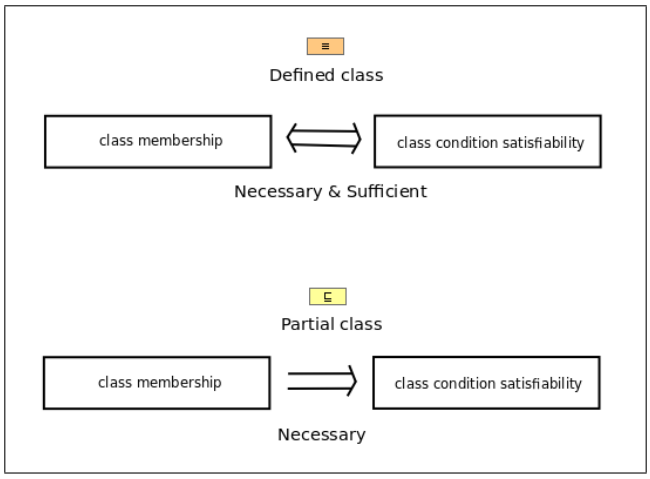

Figure 5.4: Difference between partial and defined classes

5.4.1 Modelling the world of the EO datacenter domain

The central class in the information model for the EO datacenter domain is the `data:Collection`[24] class with its properties that have the range classes `Resolution`, `Quality`, `data:ProcessingLevel`[25], `GeoExtent`, `time:TemporalExtent`[26], and `measuresRelation`. The properties describe class characteristics with their relevant range classes and are drawn inside the blue ellipse in figure 5.5.

5.4.1.1 OWL class `data:Collection`

A `data:Collection` class represents a container for homogeneous data that provides similar characteristics. A rough distinction is performed by separating `SensorRelatedCollection`s from `SensorUnrelatedCollection`s to represent the various provenance of the

[24]The concept Collection and its abbreviated namespace data are included from `http://sweet.jpl.nasa.gov/2.0/data.owl`

[25]The concept ProcessingLevel and its abbreviated namespace data are included from `http://sweet.jpl.nasa.gov/2.0/data.owl`

[26]The concept TemporalExtent and its abbreviated namespace time are included from `http://sweet.jpl.nasa.gov/2.0/time.owl`

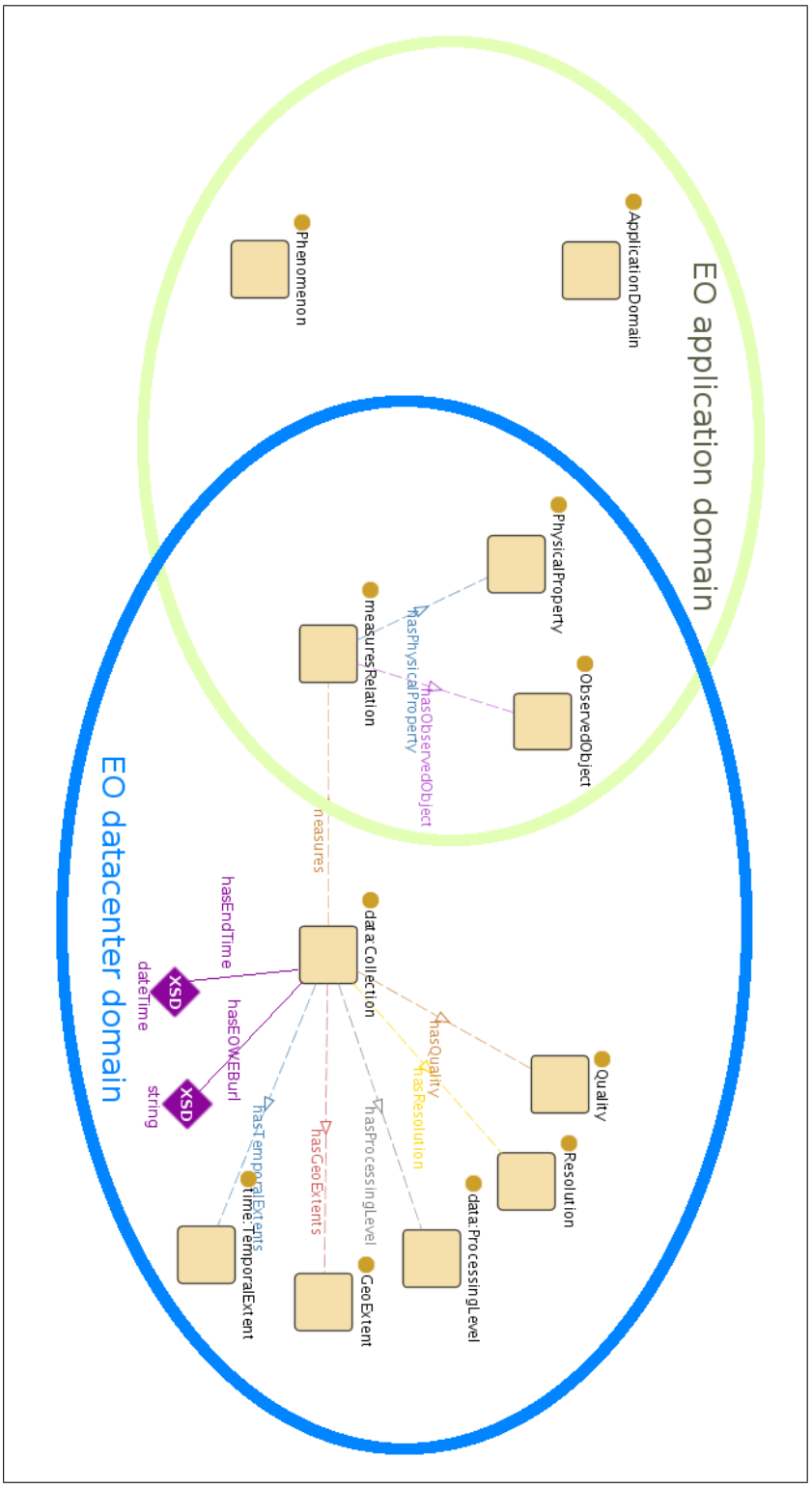

Figure 5.5: Relevance of the EOLO ontology for the datacenter domain and the application domain

data:Collections. The class has the object properties hasResolution, hasQuality, hasProcessingLevel, hasGeoExtent, hasTemporalExtent, and measures with the domain class data:Collection and their associated range classes Resolution, Quality, data:ProcessingLevel, GeoExtent, time:TemporalExtent, and measuresRelation. The datatype property hasEOWEBurl is modelled for this class with the range XSD datatype xsd:string. These structural relations are common to all data:Collections and so concrete collections are modelled as OWL individuals of the appropriate data:Collection subclass. The concrete collection ENVISAT.SCIA.L1, for example, is an OWL individual of the OWL class EMSpectroMeterCollection, which is a subclass of data:Collection and thus inherits all of its properties. This schema information enables an OWL individual belonging to any subclass to participate in the properties of the superclasses. Each of the mentioned properties has the data:Collection class as its *domain* class, whereas the *range* varies. Figure 5.6 illustrates the complete taxonomy of the data:Collection class. The number of actual OWL individuals representing real collections is displayed in parenthesis after the class name. The focus has been put on the EMSpectroMeterCollection which contains the four data:Collections ENVISAT.MERIS.L1, ENVISAT.SCIA.L0, ENVISAT.SCIA.L1, and METOP.GOME2.TC.L1B.

5.4.1.2 OWL class Resolution and Object Property hasResolution

The Resolution represents the spatial resolution of a data:Collection and takes one of the discrete values *LowResolution*, *MediumResolution*, and *HighResolution*. The values are encoded as OWL individuals. For each of the OWL individuals, OWL classes have been modelled which are structured into a taxonomy (illustrated in figure 5.7) to support subsumption based reasoning for the Resolution class. For instance, when

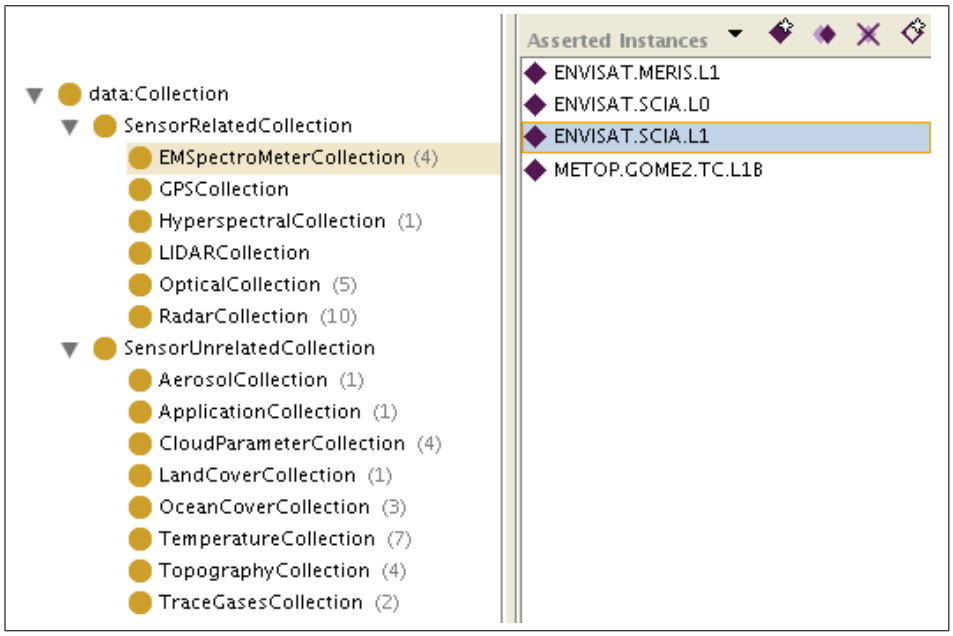

Figure 5.6: Taxonomy of OWL class `data:Collections`

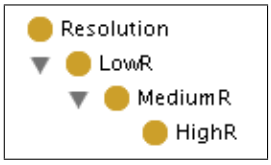

Figure 5.7: Taxonomy of OWL class `Resolution`

`LowResolution` is chosen, then `MediumResolution` and `HighResolution` are chosen implicitly, because the taxonomy structure defines that `LowR` is-a `MediumR` is-a `HighR`. `Resolution` is the *range* class of the `hasResolution` property with the *domain* `data:Collection`. The property has multiplicity 1:1, i.e. at most one `Resolution` can fill the property `hasResolution` for a `data:Collection`.

5.4.1.3 OWL class `Quality` and Object Property `hasQuality`

The `Quality` can take one of the discrete levels *LowQuality*, *MediumQuality*, and *HighQuality*. A quality level is encoded as an OWL individual.

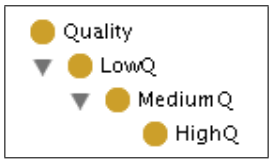

Figure 5.8: Taxonomy of OWL class `Quality`

The levels result from the characterization of `data:Collection`s based on specific criteria, for instance systematic deviation or spatial distortions. While the usage of few discrete values is a pragmatic approach for this diploma thesis, productive environments should evaluate whether it is reasonable to provide more levels with a more comprehensive structure. The corresponding property `hasQuality` has the domain `data:Collection` and the range `Quality`. The multiplicity of the property is limited to 1:1, i.e. at most one `Quality` can be assigned to a `data:Collection` via this property. Similar to the `Resolution`, the discrete `Quality` levels are modelled into a taxonomy described in figure 5.8. Reasoners can infer that `HighQuality` is equally relevant, if `MediumQuality` has been selected, because `HighQ` is-a `MediumQ`.

5.4.1.4 OWL class `data:ProcessingLevel` and Object Property `hasProcessingLevel`

The `data:ProcessingLevel` class represents the processing level of the data contained in a `data:Collection`. While data in raw binary format that is retrieved from the satellite is regarded to be in the lowest processing level (*L0*) and only machine-processable, higher-level data provides higher semantics and is human-readable. In subsequent steps, the data is radiometrically and geometrically calibrated (*L1*), geocorrected and georeferenced (*L1B*), assigned to geophysical parameters that have been measured by the sensor (*L2*), composited with other data based on temporal and/or spatial intersections (*L3*), and integrated as timeseries

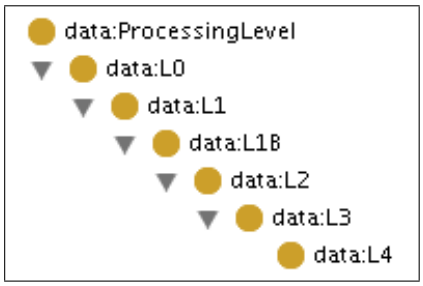

Figure 5.9: Taxonomy of OWL class `data:ProcessingLevel`

of data from the lower levels (*L4*). The mentioned levels are encoded as discrete OWL individuals. A `data:Collection` can have the property `hasProcessingLevel` with at most one `data:ProcessingLevel` as its range class, that is, the multiplicity is 1:1. The taxonomy for the OWL class `data:ProcessingLevel` class is illustrated in figure 5.9. Again, as the taxonomy is defined in the way that `data:L1B` is-a `data:L1` is-a `data:L0`, reasoning infers that if `L1B` is requested, `L1` and `L0` are applicable as well.

5.4.1.5 OWL class `GeoExtent` and Object Property `hasGeoExtents`

The `GeoExtent` class describes an abstract scale-based spatial extent that is not geographically-located, that is, only its dimensions are of importance and the position on the earth is not relevant. The named extents are *Global*, *Continental*, *Regional*, *Administrative*, *Metropolitan*, and *Maritime* which are encoded as OWL individuals. This modelling approach has several reasons. First, an alternative to this dynamic modelling would face the static modelling of every geographic area, city, and region. Then, it would be infeasible to model the whole world in this manner. One could argue, however, that third-party topologies or information gazetteers (cf. section 5.2.3) could be reused to integrate their data into the ontology. Indeed, this could be quite effective if the observations that are taken by the application domain experts take place at known locations which are modelled in the topologies. In general, the area of interest will not be bound by a city

extent or a country's borderline. Instead, the experts will use a user-defined spatial polygon describing the dimensions of the area of interest. Consider the city of Munich and the 100 km region strip at the coastline of the river Isar. This issue has been heavily discussed within the HMA forum for *Ontology based information discovery* with the conclusion that very fine-grained geographical information should not be included into the ontology. [27] Another interesting feature of `GeoExtent`s is the intuitive inclusion of smaller `GeoExtent`s into bigger ones which is not always the desired behaviour. Section 7.2 describes this problem in more detail and provides dynamic solution approaches for this issue. The property `hasGeoExtents` has the domain class `data:Collection` and the range class `GeoExtent`. The multiplicity of the property is 1:N, that is, 1 `data:Collection` can have several `eo:GeoExtent`s with the `hasGeoExtents` property. Air pollution images of a city like *Munich* have the `eo:GeoExtent`s *Metropolitan*, *Administrative*, or *Regional*. The `GeoExtent`s *Global*, *Continental*, and *Maritime* are not applicable.

5.4.1.6 OWL class `time:TemporalExtent` and Object Property `hasTemporalExtents`

The `time:TemporalExtent` class represents time as intervals. A `time:TemporalExtent` has no specific date associated with it as it merely distincts between a discrete number of interval aggregations, namely *Annual*, *Monthly*, *Weekly*, *Daily*, *Hourly*, and *Instant*. The possible aggregations are encoded as OWL individuals. The abstract modelling with `time:TemporalExtent`s eases the modelling for the knowledge engineer at the EO datacenter. Similar to `GeoExtent`, a dynamic matching

[27] `http://wiki.services.eoportal.org/tiki-view_forum_thread.php?topics_offset=1&forumId=2&comments_parentId=128` and `http://wiki.services.eoportal.org/tiki-view_forum_thread.php?forumId=2&comments_parentId=230&comments_per_page=1&thread_style=commentStyle_threaded`, last accessed: July, 9th, 2009

between concrete intervals and `time:TemporalExtent`s is developed and discussed in section 7.2. The property `hasTemporalExtents` with range class `data:Collection` and domain class `time:TemporalExtent` has the multiplicity 1:N, as one `data:Collection` can be described by several `time:TemporalExtent`s.

5.4.1.7 Datatype Property `hasEOWEBurl`

This datatype property represents the URL of a `data:Collection` in the EOWEB frontend that is deployed at the DLR. The EOWEB URL is used to pass parameters about the `data:Collection` name and eventually other EOWEB control parameters to the EOWEB applet. For instance, the EOWEB URL `https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&Service=TSX-1.SAR.Non-Imaging&QueryMode=Advanced&autoSearch=no` is stored for the `data:Collection` individual `TSX-1.SAR.Non-Imaging`. By storing the complete URL for each `data:Collection` allows for the reference to `data:Collection`s which are configured at diverse EOWEB instances. This property has the *domain* class `data:Collection` and as *range* the XSD datatype `xsd:string`.

5.4.1.8 Datatype Property `hasEndTime`

This datatype property represents the date when a `data:Collection` stopped getting current data due to the termination of the underlying satellite or airborne missions. Therefore, the property is used to model the currentness of a `data:Collection` that is only relevant for filtering. The `hasEndTime` property has the *domain* class `data:Collection` and the *range* XSD datatype `xsd:dateTime`.

5.4.2 Modelling the bridge between EO datacenter domain and EO application domain

In the EOLO ontology, the two domains EO datacenter and EO application are bridged via semantic links which are created based on a common concept of both worlds, namely the measurement features of physical properties for observable planetary objects. Figure 5.5 illustrates the classes which bridge the world of EO datacenter operators and EO application domain experts inside the intersection of both ellipses.

5.4.2.1 OWL class `ObservedObject` and Object Property `hasObservedObject`

The class `ObservedObject` represents objects on earth that can be observed (e.g. upper atmosphere, sea surface, land surface). `ObservedObject`s are modelled as OWL classes that are taken from the SWEET2.0 Beta ontologies for the compliance to a controlled terminology for earth sciences (cf. 5.2.2). The current structure of the `ObservedObject` class is illustrated in figure 5.10. The property `hasObservedObject` has the domain class `measuresRelation` and the range class `ObservedObject`.

5.4.2.2 OWL class `PhysicalProperty` and Object Property `hasPhysicalProperty`

The class `PhysicalProperty` defines the physical properties that can be measured by various instruments (e.g. temperature, pressure, spectral albedo). The properties are encoded as OWL classes. Similar to the `ObservedObject` the classes are derived from the SWEET2.0 Beta ontologies. The hierarchy that has been modelled in the scope of this diploma thesis is shown in figure 5.11. The property `hasPhysicalProperty` has the domain class `measuresRelation` and the range class `PhysicalProperty`.

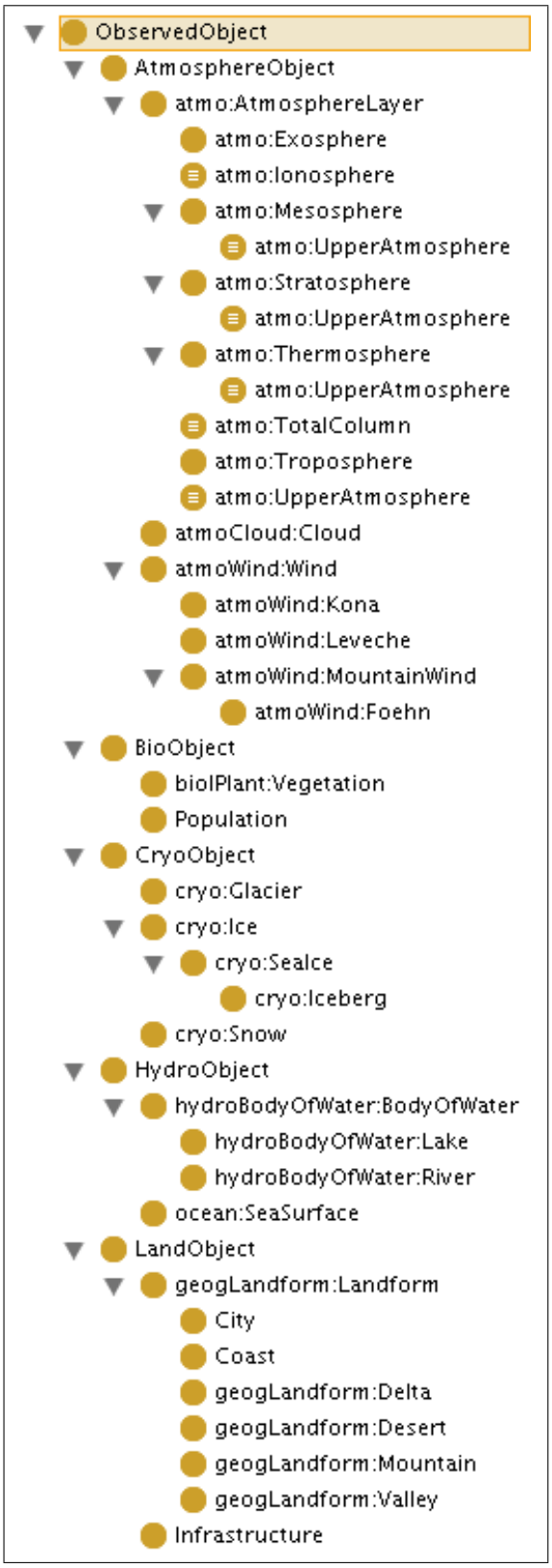

Figure 5.10: Taxonomy of OWL class `ObservedObject`

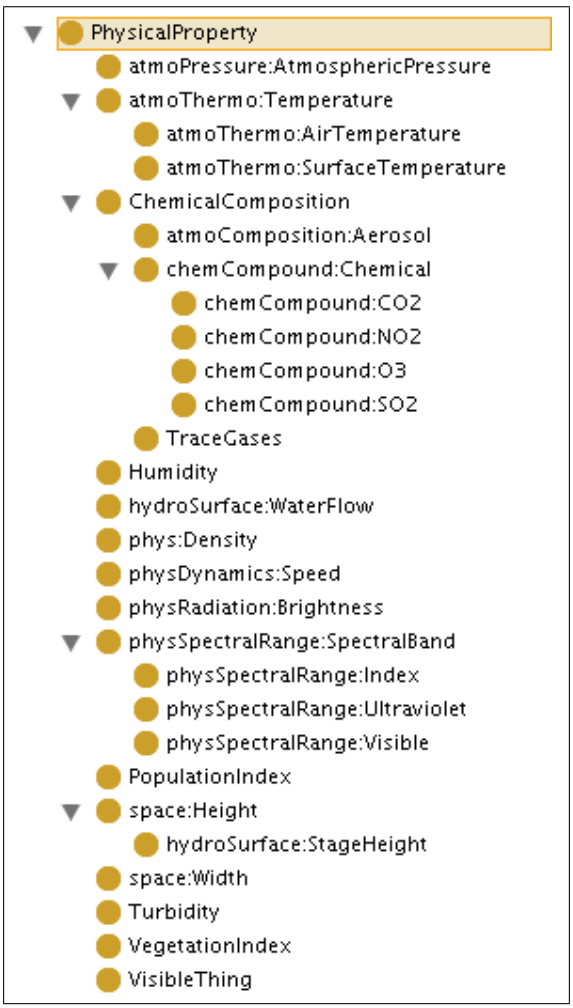

Figure 5.11: Taxonomy of OWL class `PhysicalProperty`

5.4.2.3 OWL class `measuresRelation` and Object Property `measures`

The class `measuresRelation` is a class representing the "n-ary relation"[28] of the property `measures` between the classes `data:Collection`, `ObservedObject` and `PhysicalProperty`. It has the range class `data:Collection` and the domain class `measuresRelation` with the multiplicity 1:N, i.e. one `data:Collection` can have several `measuresRelation`s with the property `measures`. This object property differs from the other object properties of the `data:Collection` class. While the previously mentioned properties are modelled per OWL individual, the link to the measuresRelation is defined as a *necessary* condition axiom for the subclass of `data:Collection`. Therefore, all OWL individuals with the type of the subclass must satisfy the condition axiom. The reason for such modelling is the fact that these measurements are all common for a specific collection type and the axiom-based definition can be used for reasoning tasks in the ontology. For instance, the `EMSpectroMeterCollection` can measure *aerosols in the atmosphere*, the *speed of winds*, the *height of clouds*, and the *temperature of the sea*. The mentioned aggregations of the classes `ObservedObject` and `PhysicalProperty` are defined by the class `measuresRelation`. Figure 5.12 illustrates its class taxonomy.

5.4.3 Modelling the world of EO application domains

For this thesis, only a basic modelling approach of the EO application domain has been taken. However, the work that has been performed can be taken as a base for extended modelling and customizing. The main classes for EO application domain experts are `ApplicationDomain`, `Phenomenon`, and `measuresRelation`. The relevant classes for EO application domain

[28] Defining N-ary Relations on the Semantic Web, `http://www.w3.org/TR/swbp-n-aryRelations/`, last accessed: July, 9th, 2009

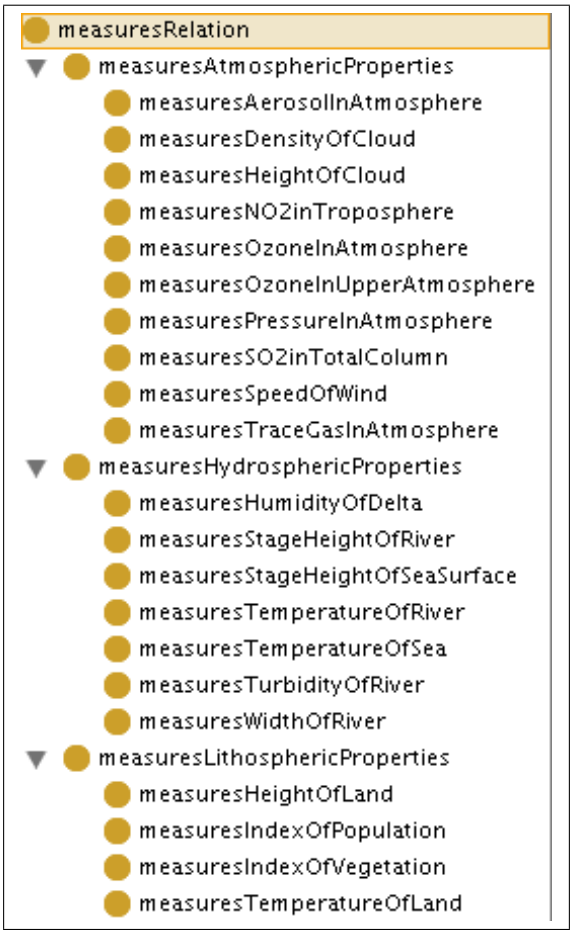

Figure 5.12: Taxonomy of OWL class `measuresRelation`

experts are depicted in figure 5.5 inside the green circle.

5.4.3.1 OWL class `Phenomenon`

The `Phenomenon` class represents the entry point for application domain experts for defining and querying natural or anthropogenic phenomena. Besides the `measuresRelation` class, this class is the base for qualitative Description Logics (DL) reasoning by defining axioms in subclasses of `Phenomenon` with conditions which are necessary and sufficient. Thus, the subclasses of `Phenomenon` are *defined* classes, i.e. it is necessary that any OWL individual which is a member of such an OWL class must satisfy its condition and it is sufficient for an OWL individual to satisfy the condition of the OWL class to be a member of it (cf. section 5.4). The axioms for the `Phenomenon` subclasses are defined based on the `measuresRelation` class. For instance, the `Phenomenon atmoComposition:AirPollution`[29] defines the condition in figure 5.14 as necessary and sufficient. It is important to notice that there is no static property between the `Phenomenon` and the `measuresRelation`. Instead, the links between the two classes are established dynamically, based on reasoning mechanisms to infer OWL class membership of OWL individuals of the OWL class `data:Collection` to the OWL class `Phenomenon`. Figure 5.13 illustrates the connection between `Phenomenon` and `measuresRelation` that is dynamically established by DL reasoning with the solid red line between the classes. The axiom in figure 5.14 classifies all `data:Collection`s based on the condition to have at least one (boolean $\vee$) of the subclasses of `measuresRelation` in their `measures` property. Individuals satisfying this condition are then inferred to be members of the `atmoComposition:AirPollution` subclass. The illustrated axiom defines only boolean disjunctions ($\vee$), but, in general also other

[29] The class `atmoComposition:AirPollution` is taken from the URI `http://sweet.jpl.nasa.gov/2.0/atmoComposition.owl#AirPollution`

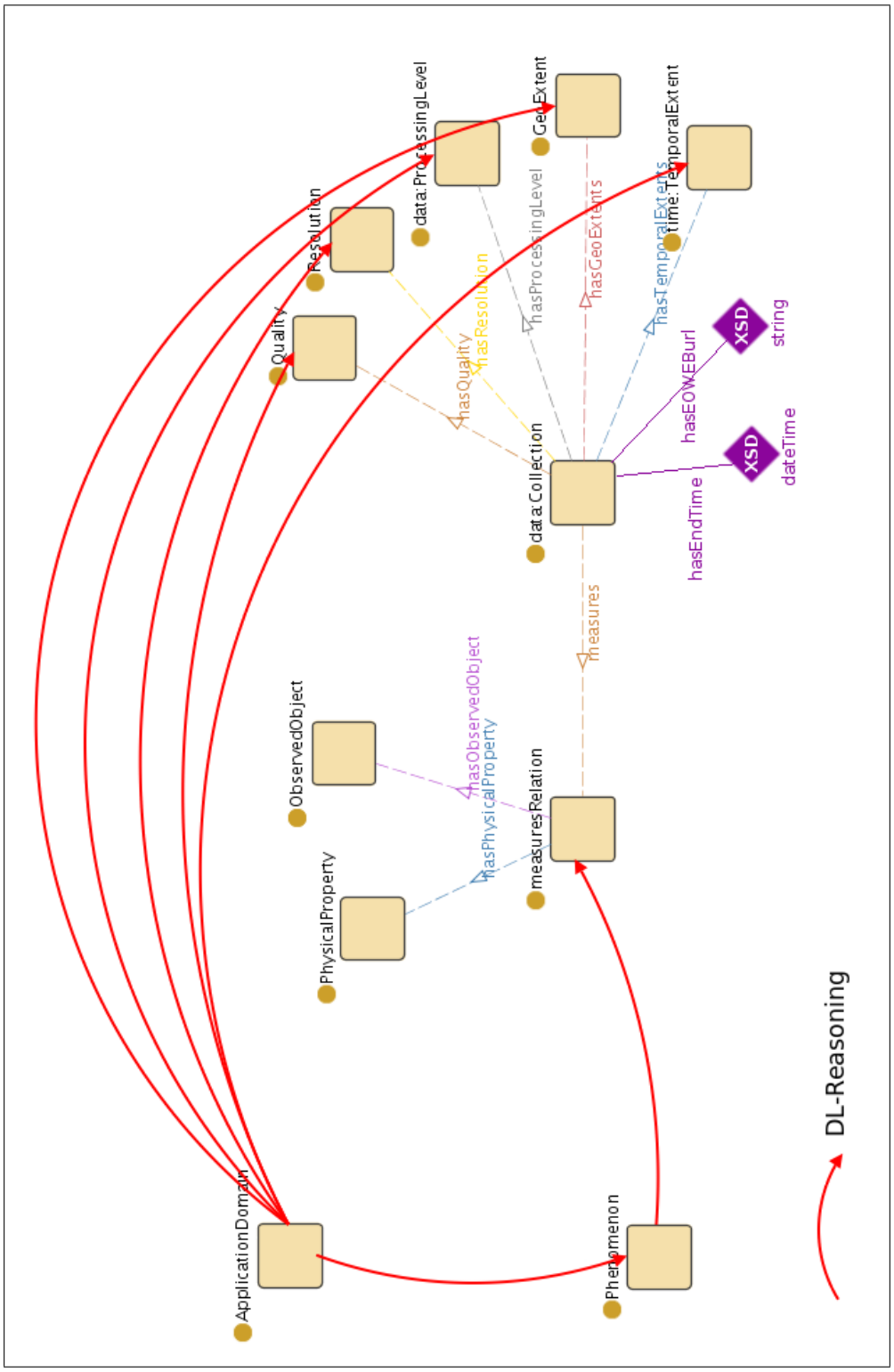

Figure 5.13: Description Logics (DL) reasoning applied to classes for EO application domain experts to obtain `data:Collections` from the EO datacenter

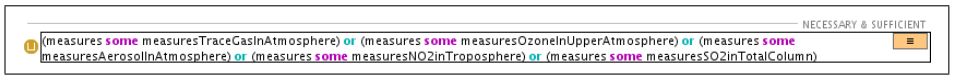

Figure 5.14: Necessary and sufficient (defined) definitions for class `atmoComposition:AirPollution`

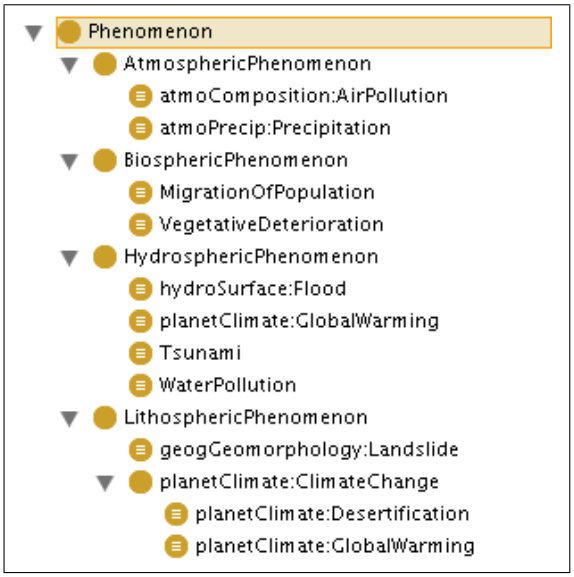

Figure 5.15: Taxonomy of OWL class `Phenomenon`

boolean operations ($\wedge$, $\neg$), value ($\ni$) or cardinality restrictions ($\geq$, $\leq$) can be used to strengthen the axiom and influence the inference results. Besides this qualitative axioms no other restrictions are modelled into the `Phenomenon` class. These restrictions are moved upwards to the next abstraction layer `ApplicationDomain`, which is explained in the next section. The currently modelled class taxonomy of the `Phenomenon` class is illustrated in figure 5.15.

5.4.3.2 OWL class `ApplicationDomain`

The class `ApplicationDomain` defines axioms that include several phenomena and restrictions on the `data:Collection` properties. Therefore, it serves as a shortcut to a manual selection of one or

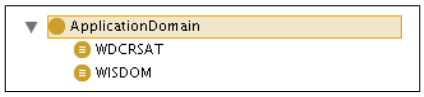

Figure 5.16: Taxonomy of OWL class `ApplicationDomain`

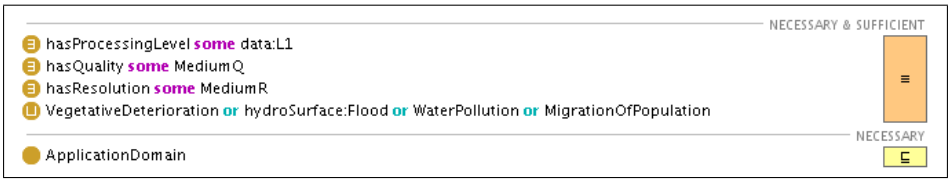

Figure 5.17: Axioms for `ApplicationDomain` `WISDOM`

several `Phenomena` and restrictions on the `data:Collection` properties `measuresRelations`, `GeoExtent`, `time:TemporalExtent`, `Resolution`, `Quality`, and `data:ProcessingLevel`. The links are established by DL reasoning (see solid red lines in figure 5.13). The restrictions can be modelled on any subset of the mentioned classes to represent the application domain. Currently, only the two application domains `WDC-RSAT` and `WISDOM` have been defined (cf. sections 3.2.1 and 3.2.2). The class taxonomy for the OWL class `ApplicationDomain` is shown in figure 5.16. A sample set of axioms for the `WISDOM` application domain can be found in figure 5.17. The axioms state that the `data:ProcessingLevel` must have the level `data:L1`, the `Quality` must be `MediumQ`, the `Resolution` must be `MediumR` and the possible `Phenomena` are defined by the axiom `VegetativeDeterioration or hydroSurface:Flood or WaterPollution or MigrationOfPopulation`. The reasoner uses these axioms to infer the membership of `data:Collections` to the `WISDOM` `ApplicationDomain` because they satisfy the class conditions of it.

Chapter 6

Implementation report

In the previous chapter, the EOLO ontology has been formally defined. This chapter describes the implementation details of the EOLOsearch web demonstrator that implements the user-based access to the modelled EOLO ontology. Moreover, use cases are explained for EO users who use the knowledge contained in the ontology and EO knowledge engineers who maintain this knowledge within the ontology. Furthermore, the deployment of the demonstrator at the DFD is presented.

6.1 Architecture

The EOLOsearch web demonstrator is written in the languages Java, SQL, AJAX, and CSS and implements the *Model-View-Controller* pattern. It can be deployed as a dynamic web application into a Java application server. The server side is implemented as a `HTTPServlet`, the client side as a Java Server Page (JSP) with AJAX scripts for the dynamic result integration. An architectural overview of the developed system is illustrated in figure 6.1. The knowledge base for the EO domain is kept inside the EOLO ontology. The ontology can be accessed by loading the Protégé project file EOLO.pprj with the Protégé-OWL ontology modelling tool.

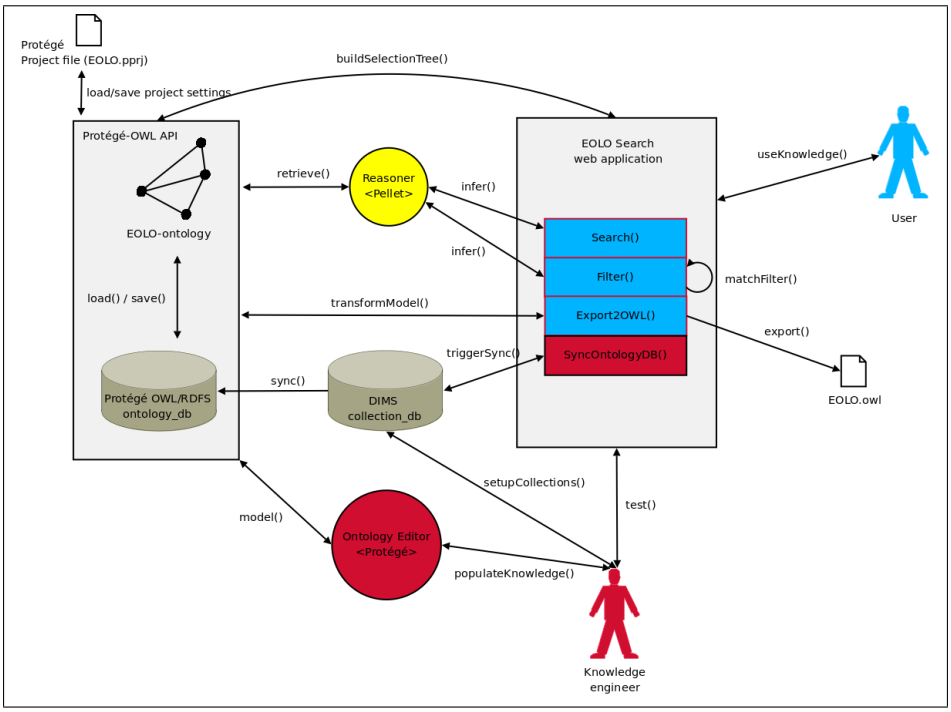

Figure 6.1: Architecture overview

The ontology is stored persistently in a database, the so-called *ontologyDB*. The Protégé-OWL API provides a generic reasoner interface called `edu.stanford.smi.protegex.owl.inference.reasoner.ProtegeReasoner`[30] that is implemented by various concrete reasoner adapters. The architecture has been tested with the Pellet reasoner which is delivered with the Protégé-OWL libraries. The reasoner interface provides methods for accessing the ontology and performing reasoning tasks like classification, subsumption, and satisfiability on it. Moreover, the API is employed to transform the high-level ontology model from the database to a textual representation in RDF/XML format. Exporting the ontology into a common syntax is especially important for collaborative ontology developments as the ontology can be imported into other ontologies and modelling tools. The

[30] `http://protege.stanford.edu/protege/3.4/docs/api/owl/edu/stanford/smi/protegex/owl/inference/reasoner/ProtegeReasoner.html`

ontologyDB can be synchronized with the so-called *collectionDB* which is used by various operating tools and the EOWEB system for the retrieval of `Collection` metadata. The architecture has two usergroups with different aims, the common EO user and the EO knowledge engineer. While the common EO user only use the system to perform search queries, to apply filters, or to export the ontology to an OWL file in the EOLOsearch web demonstrator (blue), the EO knowledge engineer additionally can synchronize the *ontologyDB* and has access to the *collectionDB* and the modelling tools (red). The detailed workflow for EO knowledge engineers is explained in sections 6.3 and 6.4.

6.1.1 Controller

The `EOLOServlet` class serves as the controller of the web application that dispatches the various user requests to the correct model classes. The servlet contains a `HashMap<String, Action>` so that the user-defined action can be invoked dynamically by parameter passing within the URL. The code for the `doGet()` method that is called when the application URL `http://zebra:8080/EOLOSearch` is invoked to handle the request and response dispatch is displayed in figure 6.2.

The package `controller.action` contains the *actions* which implement the web interfaces for the business logic of the application. They are invoked by the controller uniformly as each `controller.action.Action` implements an abstract method called `doAction()` of its abstract super class `controller.action.Action`. *Actions* delegate their requests to the appropriate classes in the model. The implemented *actions* are `controller.action.Export2OWLAction`, `controller.action.FilterCollectionsAction`, `controller.action.RetrieveSelectionTreeAction`, `controller.action.SearchCollectionsAction`,

```
protected void doGet(
        HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

    // read the "action" parameter from the request and check what to do
    Object actionObj = request.getParameter("action");
    if (actionObj != null) {
        String action = (String) actionObj;

        // get the action to perform from the actions HashMap!
        Action a = this.actions.get(action);
        if (a != null) {
            // invoke the action
            HashMap<String, CollectionMetadataBean> collections =
                a.doAction(request, response);

            if (collections != null) {
                // if the action returned any collections,
                // export them to the client!
                a.exportCollections(request, response, collections);
            }
        }
    }
}
```

Figure 6.2: `doGet()` method of `EOLOServlet`

`controller.action.SyncOntologyAction` and the `controller.action.PerformanceTestAction`. The different actions are illustrated in figure 6.3 All *actions* are described in more detail in the use cases of the developed application in sections 6.2, 6.3 and 6.4.

6.1.2 Model

The model of the demonstrator contains the business logic that is called by the *actions*. It comprises of the functionality to retrieve a selection tree that consists of `Phenomena` and `ApplicationDomains` and is used to generate the GUI. In addition, it handles `data:Collection` request and retrieval, as well as applying filters to search results of `data:Collection`s. Furthermore, it synchronizes between the *collectionDB* and the *ontologyDB* and it exports the EOLO ontology into a common ontology syntax. Finally, random tests measure the performance of the synchronization of and the reasoning for `data:Collection`s. All business logic is distributed into the classes in the

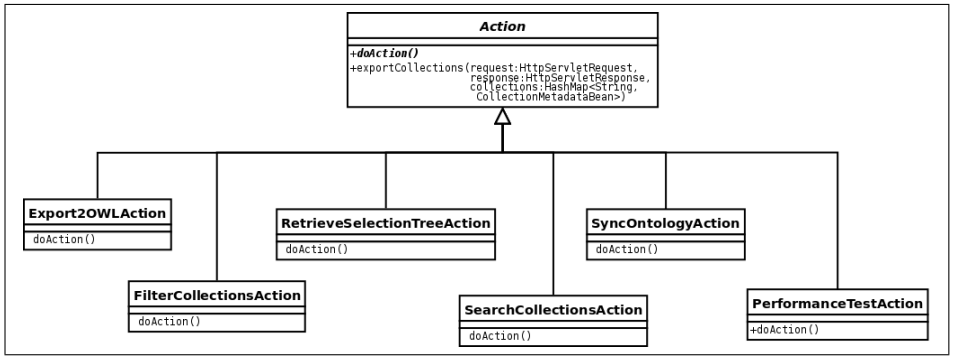

Figure 6.3: Abstract and concrete actions

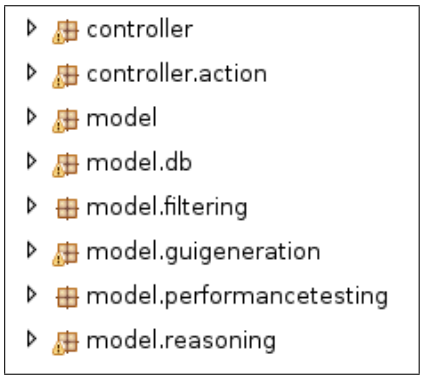

Figure 6.4: Package structure of the EOLOsearch web demonstrator

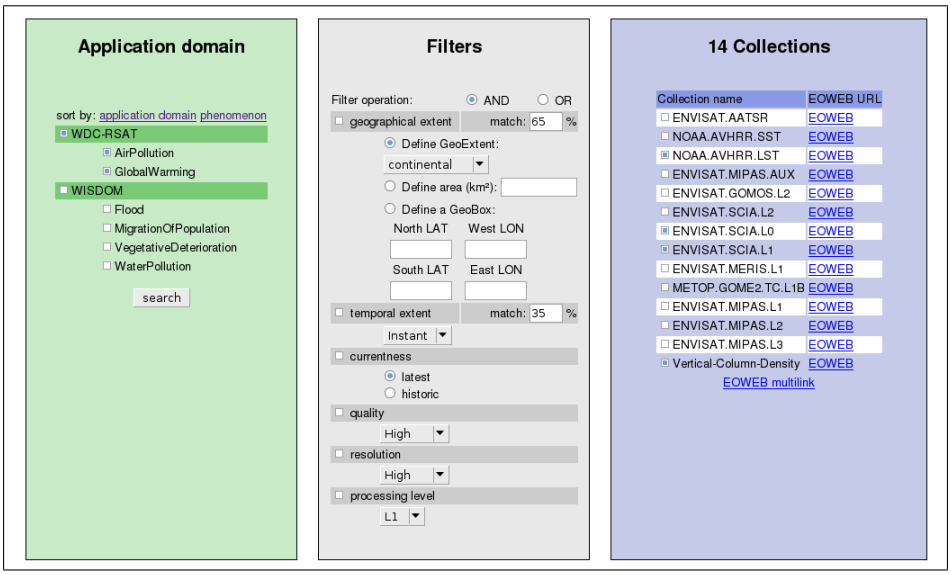

Figure 6.5: EOLO Search web application screenshot

package `model` and its subpackages (see figure 6.4).

6.1.3 View

The GUI of the web application is splitted into the three regions *Application domain / Phenomenon* for the search selection, *Filters* for search parameters, and *Collections* for the search results, depicted in figure 6.5. The GUI interfaces with the model asynchronously through AJAX by adjusting the `HTTP GET` parameters in the `URL` that is sent to the `EOLOServlet`. When the user changes the selection of application domains or phenomena and hits the search button or when the filter settings are changed a new request is sent to the servlet containing the current selections. When the result is available, the GUI is updated by parsing the resulting XML data.

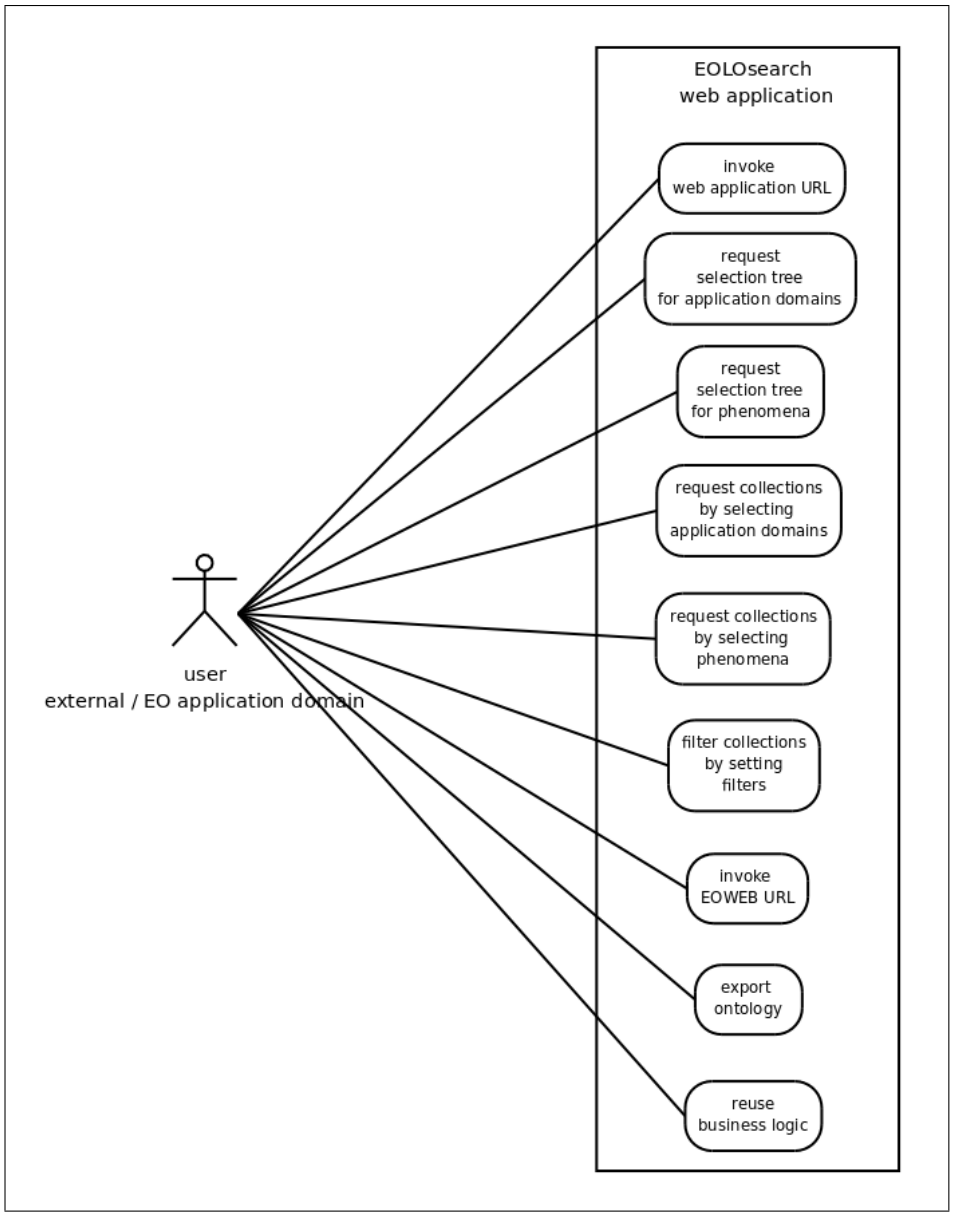

Figure 6.6: Use cases for EO application domain and external users

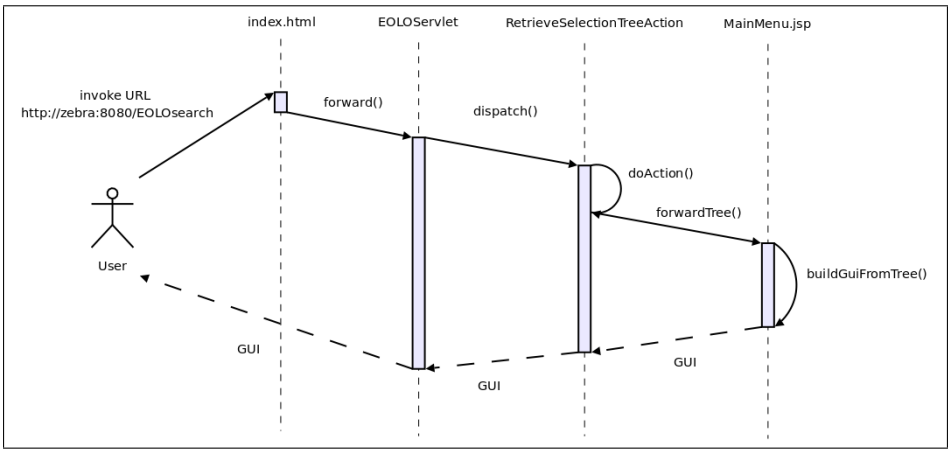

Figure 6.7: Request/response sequence for retrieving the selection tree and generating a GUI from it

6.2 Use cases for EO users

EO users are referred to as users that have an interest in retrieving EO data from the web demonstrator. They can be application domain experts who have some a priori knowledge about the EO domain or external users without any EO background. The use cases for EO users are illustrated in figure 6.6. EO users *use the knowledge* contained in the EOLO ontology to retrieve and filter `data:Collection`s for their requested `Phenomena` or `ApplicationDomain`s. Moreover, they can click directly on the links for the `data:Collection`s in the EOWEB system. By providing the ability to export the ontology and to reuse the developed business logic for `data:Collection` retrieval through HTTP and XML, platform-independent applications can emerge.

6.2.1 Invoking the web application URL

First of all, the user invokes the URL of the EOLOsearch web demonstrator `http://zebra:8080/EOLOsearch`, demonstrated in figure 6.7. When the URL is invoked from the web browser, the request is forwarded

to the static `index.html` page which again forwards the request to the `EOLOServlet` by invoking `http://zebra:8080/EOLOsearch/servlet?action=retrieveSelectionTree&treeType=appTree`. The business logic that is started then, is explained in detail in the next subsection.

6.2.2 Requesting the selectionTree for `ApplicationDomains`

When the URL `http://zebra:8080/EOLOsearch/servlet?action=retrieveSelectionTree&treeType=appTree` is requested, the request is dispatched to the `controller.action.RetrieveSelectionTreeAction` which retrieves the selection tree for `ApplicationDomain`s from the ontology and forwards the tree to the dynamic `MainMenu.jsp` page. Then, a GUI with `ApplicationDomain`s can be generated from it and displayed to the user (see figure 6.8). Each `ApplicationDomain` can have multiple `Phenomena` associated to it through its axioms. Application domains are displayed in the headings of the selection tree (see 6.8, 1). The selection of an application domain includes the selection on all modelled criteria to the other concepts `Phenomenon`, `Resolution`, `Quality`, `data:ProcessingLevel`, `GeoExtent`, and `TemporalExtent`, implicitly. Under the application domain, several phenomena are listed which can be selected or deselected (6.8, 2). The de-/selection of a phenomenon resets all criteria for the application domain and allows for manual filtering of the mentioned concepts in a next step. The selection tree can be switched from application domain-based to phenomenon-based by clicking the appropriate link at the top of the selection tree (6.8, 4).

6.2.3 Requesting the selectionTree for `Phenomena`

Similar to the retrieval of the selection tree for `ApplicationDomain`s, the retrieval of the selection tree for `Phenomena` is requested by invoking the URL `http://zebra:8080/EOLOsearch/servlet?action=`

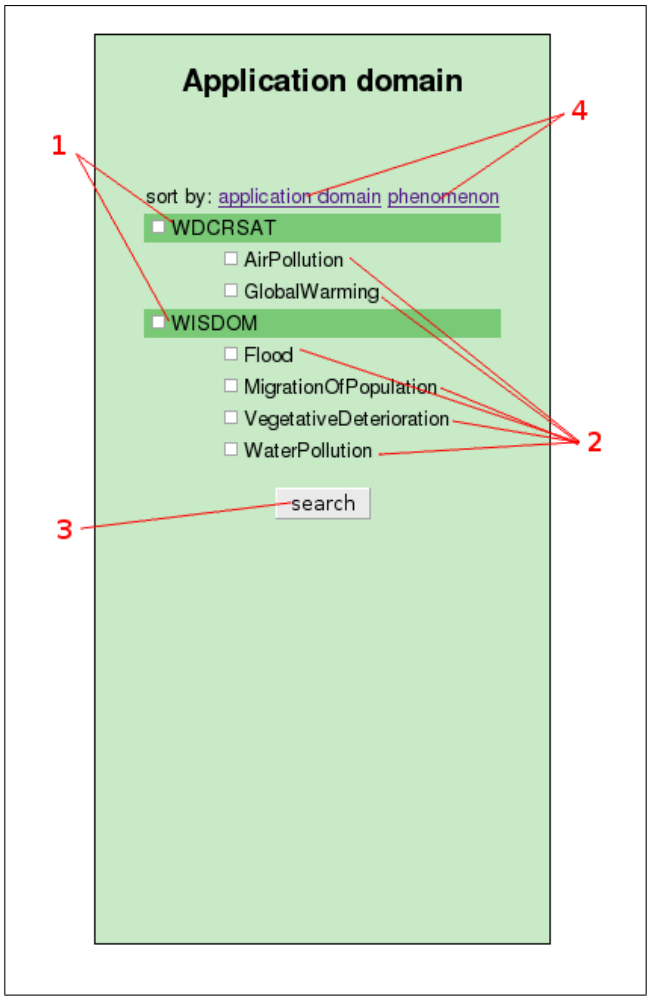

Figure 6.8: Generated selection tree for application domains

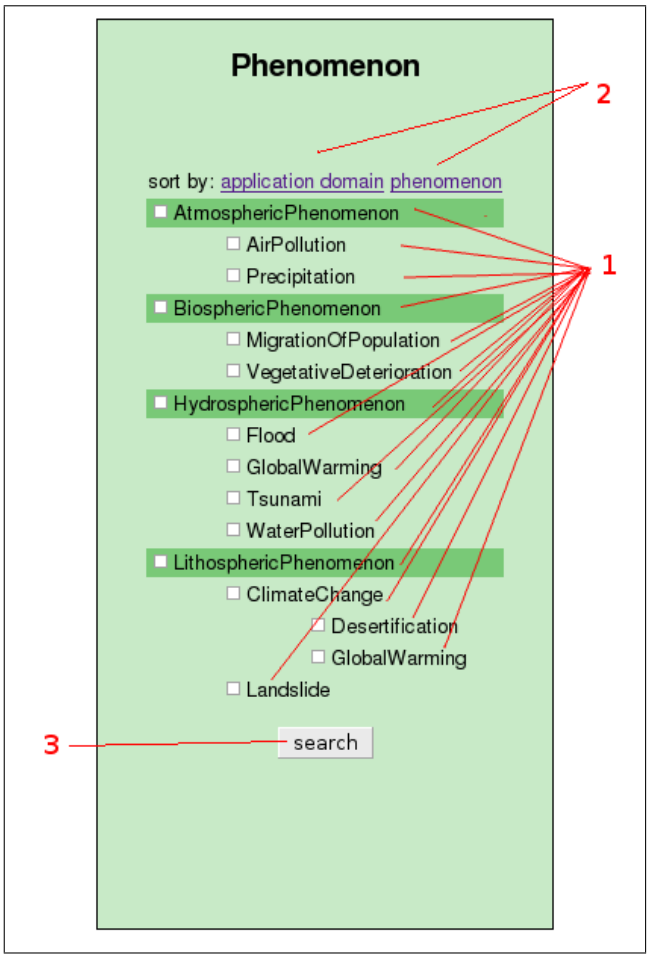

Figure 6.9: Generated selection tree for phenomena

`retrieveSelectionTree&treeType=phenomenonTree`. The phenomenon-based selection displays the available phenomena which are structured topologically under the main categories `AtmosphericPhenomenon`, `BiosphericPhenomenon`, `HydrosphericPhenomenon`, and `LithosphericPhenomenon` (see figure 6.9, 1). Similar to the application domain-based selection mode, the selection tree is generated from the ontology class `Phenomenon` (cf. section 5.4.3.1) and the user may narrow the search results by applying several filters, which are explained in the next section. In addition, the links at the top for switching the selection

tree type and the search button (see figure 6.9, 2 and 3) work similar to the application domain-based mode.

6.2.4 Requesting `data:Collections` for selected `ApplicationDomains`

From the selection tree for `ApplicationDomains`, the user can select and request the `ApplicationDomains` of interest by clicking the *search* button (6.8, 3). The `data:Collections` which are relevant for a specific `ApplicationDomain` are not hardcoded into the ontology. Instead, the relevance is inferred dynamically by reasoning over the axioms of the selected `ApplicationDomains`. Application domain-based selection does impose filter settings on the concepts `Phenomenon`, `Resolution`, `Quality`, `data:ProcessingLevel`, `GeoExtent`, and `TemporalExtent` as such information is axiomatically encoded in the ontology for the `ApplicationDomain` class. Considering the sample `ApplicationDomain WISDOM`, the provided axioms (see figure 5.17) are used to infer the membership of the `data:Collection` individuals `AQUA.MODIS, GLOBE.DEM, NOAA.AVHRR.NDVI, TSX-1.SAR.Imaging, TSX-1.SAR.L1b-High-Resolution-Spotlight, TSX-1.SAR.L1b-ScanSAR, TSX-1.SAR.L1b-Spotlight, TSX-1.SAR.L1b-Stripmap, TSX-1.SAR.Non-Imaging,` to the application domain in a dynamic manner. The request URL for this selection is `http://zebra:8080/EOLOsearch/servlet?action=searchCollections&concepts=http://www.dlr.de/ontologies/EOLO.owl!WISDOM`. The reasoner inspects the semantic axioms and creates the dynamic links between the classes `ApplicationDomain` and `data:Collection`. By inferring the membership dynamically, new `data:Collections` with similar capabilities on the filter concepts are classified with the same logical axiom without the need for axiomatic

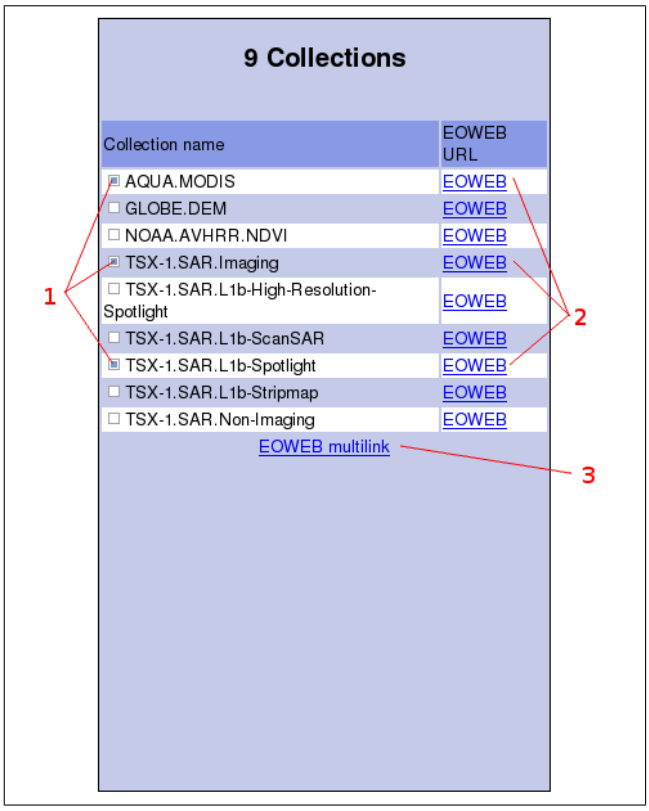

Figure 6.10: Search results for the `ApplicationDomain WISDOM` obtained by reasoning

modifications.

The search results which are obtained by reasoning for the `ApplicationDomain WISDOM` are illustrated in figure 6.10. The result list consists of the name of the collection and the URL in the EOWEB system (see figure 6.10, 2). In the EOLO Search web application, the search for collections is dispatched to the `controller.action.SearchCollectionsAction`. Figure 6.11 displays the sequence diagram for performing a search for `data:Collections`. The `doAction()` method invokes the `calcInferredIndividuals()` method of the `model.reasoning.ReasonerInvoker` class. This class handles the communication to the configured reasoner and to the knowledge base

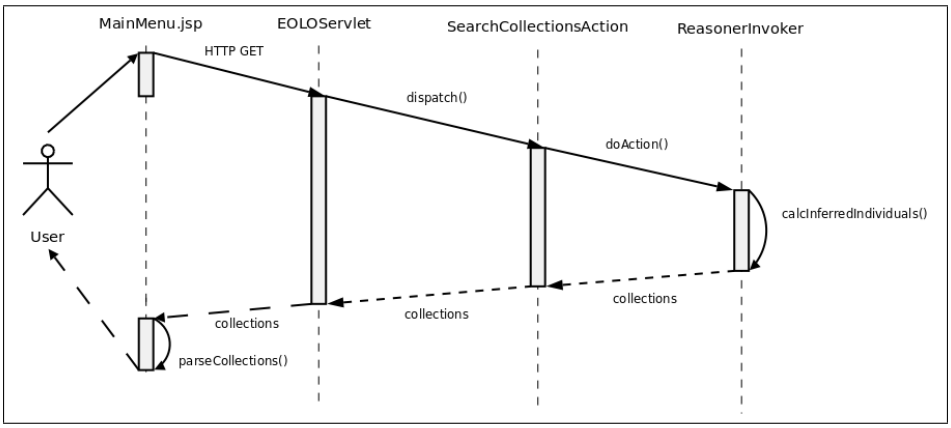

Figure 6.11: Request/response sequence for searching for `data:Collection`s

containing the EOLO ontology. The resulting `data:Collection`s are sent to the client and can be parsed into the GUI to display the search results.

6.2.5 Requesting `data:Collection`s for selected `Phenomena`

In the selection tree for `Phenomena` the user can select the `Phenomena` of interest. Phenomenon-based reasoning infers the membership of `data:Collection`s to `Phenomena`. However, there are some slight differences to application domain-based reasoning. Phenomenon-based search does not impose any filters on the concepts `Phenomenon`, `Resolution`, `Quality`, `data:ProcessingLevel`, `GeoExtent`, and `TemporalExtent` as such information is not encoded in any form into the ontology for the `Phenomenon` class. Instead, the user can set filters on these properties in a next step. Firstly, `Phenomena` do not include other phenomena for reasoning in order to limit the reasoning results and the modelling complexity in the ontology. Secondly, the `Phenomenon` uses only the `measuresRelation` class in its axioms. The reason for this is the fact that the restrictions on other concepts such as `Quality`, `Resolution`, `ProcessingLevel`, `GeoExtent`, and `TemporalExtent` cannot be generalized at the phenomenon level as different application domains could be interested

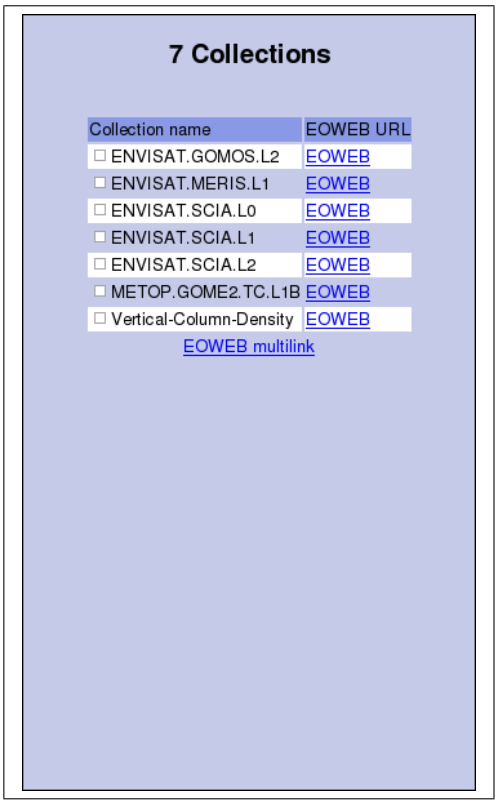

Figure 6.12: Search results for the `Phenomenon atmoComposition:AirPollution` obtained by reasoning

in the same phenomenon with different restrictions on these concepts. One could argue to model these `Phenomenon` variants as OWL individuals instead of the modelling as OWL classes. However, such modelling is contraproductive since it is not possible to define the class-based axioms required for reasoning for OWL individuals. Furthermore, the aggregation of the concepts `Phenomenon` and *ApplicationDomain* should be performed at a higher abstraction layer, namely the `ApplicationDomain` OWL class to ensure better maintainability of the ontology. A sample axiom for the `Phenomenon atmoComposition:AirPollution` is illustrated in figure 5.14. The resulting `data:Collection`s for the selection of the `Phenomenon atmoComposition:AirPollution` are `ENVISAT.GOMOS.L2`,

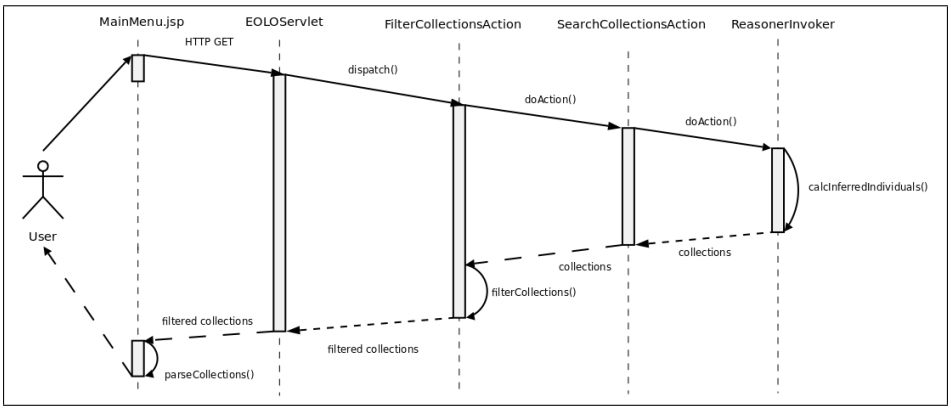

Figure 6.13: Request/response sequence for filtering `data:Collections`

`ENVISAT.MERIS.L1`, `ENVISAT.SCIA.L0`, `ENVISAT.SCIA.L1`, `ENVISAT.SCIA.L2`, `METOP.GOME2.TC.L1B`, `Vertical-Column-Density` (see figure 6.12). The request URL for this selection is `http://zebra:8080/EOLOsearch/servlet?action=filterCollections&concepts=http://sweet.jpl.nasa.gov/2.0/atmoComposition.owl!AirPollution`. Figure 6.13 displays the sequence diagram for performing a filtering of `data:Collections`. First, the filter request is dispatched to the `controller.action.FilterCollectionsAction`. Then, its `doAction()` method forwards the request to the `controller.action.SearchCollectionsAction` to start reasoning tasks for the selected `Phenomena`. The reasoning results are subsequently filtered by the business logic of the `controller.action.FilterCollectionsAction`. Finally, the `data:Collection`s that matched the filters are returned to the client and displayed in the GUI.

6.2.6 Filtering `data:Collections`

When performing phenomenon-based reasoning, it is possible to adjust the filter settings for the search results. The reason for this, is the fact

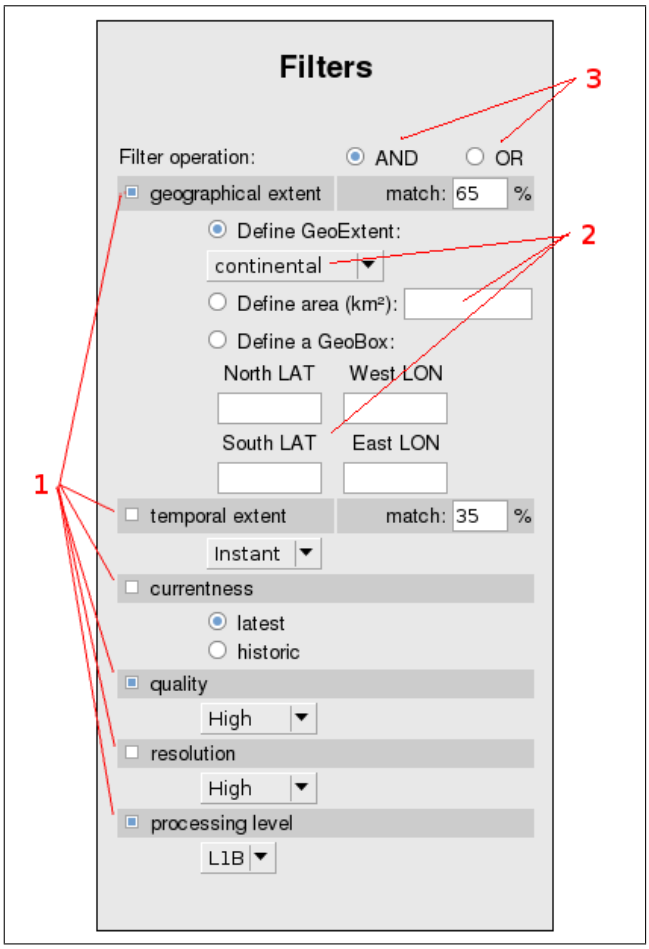

Figure 6.14: Filters for the current selection

that a `Phenomenon` does not impose any restrictions on the filter concepts via class axioms enabling the user to freely create the restrictions online. The implemented filters refer to the concepts `GeoExtent`, `TemporalExtent`, `Resolution`, `data:ProcessingLevel`, `Quality`. The filters can be activated or deactivated by ticking the filter checkbox (see figure 6.14, 1). All selected filters can be logically combined with *AND* and *OR* (see figure 6.14, 3). A user-convenient filtering has been implemented which is started automatically after changing any options or after typing in values and hitting the return button in input fields by AJAX scripts.

6.2.6.1 Filter geographical extent

The *filter geographical extent* provides the possibility to filter the `data:Collection`s with a numeric filter based on the percentual spatial inclusion of its `GeoExtent` within all possible `GeoExtent`s. It contains three different filtering input modes, *GeoExtent*, *area*, and *GeoBox*. The *Define GeoExtent* filter is a simple textual selection with the options *Global*, *Continental*, *Maritime*, *Regional*, *Administrative*, and *Metropolitan* defining their areas in square kilometers, implicitly. The area input field allows for manual input of the area of the GeoExtent. The GeoBox can be used to provide a user-defined bounding box (see figure 6.14, 2). The coordinates *North LAT*, *South LAT*, *East LON*, *West LON* are used to create a `com.informix.geodetic.types.GeoBox` SQL object for the *Informix Geodetic Datablade Module*[31] which can calculate its area with the procedure `area(GeoObject)`. Figure 6.15 shows the Java code that creates such an Informix-compliant `com.informix.geodetic.types.GeoBox` object from the user inputs by employing the geodetic libraries from the Geodetic Datablade module. Moreover, the coordinates are passed to the generation of the EOWEB link which creates a rectangular selection on the world map in the EOWEB applet. The user can define the required match percentage for the geographical extent by adjusting the value. 100% means an exact match, 50% means that the results can have at most twice the size, 200% means that the results can have at most half of the size of the area of the GeoExtent. For instance, the result set R_{ge} for the selection of the `GeoExtent` *Regional* and the match percentage of 200% is defined as follows:

[31] `http://www-01.ibm.com/software/data/informix/blades/geodetic/`, last accessed: August 3, 2009

```
public static GeoBox createGeoBox(
        double swLat, double swLon, double neLat, double neLon) {

    GeoBox go = null;

    try {
        // spatial reference system identifier
        short srid = 0;

        // Southwest-coordinates
        GeoCoords sw_coords = new GeoCoords(swLat, swLon, srid);
        GeoPoint sw = new GeoPoint(sw_coords, new GeoAltRange(),
                new GeoTimeRange());

        // Northeast-coordinates
        GeoCoords ne_coords = new GeoCoords(neLat, neLon, srid);
        GeoPoint ne = new GeoPoint(ne_coords, new GeoAltRange(),
                new GeoTimeRange());

        // create a new GeoObject
        go = new GeoBox(sw, ne);

    } catch (OutOfRange e) {
        Logger.error(e.getMessage(), e);

    } catch (GeoException e) {
        Logger.error(e.getMessage(), e);
    }

    return go;
}
```

Figure 6.15: Method `createGeoBox()` from `GeoBoxFactory` for the instantiation of Informix-compliant GeoBoxes

$$R_{ge}(Regional, 2.0) = \{ge_i \mid i \in \{Global, Continental, Maritime, Regional, Administrative, Metropolitan\}, \frac{area(Regional)}{area(ge_i)} \geq 2.0\}.$$

The area is calculated in square kilometers. `data:Collection`s that have a `GeoExtent` that is in the result set R_{ge} satisfy this filter.

6.2.6.2 Filter temporal extent

The *filter temporal extent* allows the filtering of the `data:Collection`s with a numeric filter based on the percentual temporal inclusion of its `TemporalExtent` within all possible `TemporalExtent`s. The filter options are: *Annual*, *Monthly*, *Weekly*, *Daily*, *Hourly*, and *Instant*. Similar to the geographical extent, the user can define the match percentage of the selected option. For instance, the result set R_{te} for the selection of the `TemporalExtent` *Daily* and the match percentage of 87% is defined as follows:

$$R_{te}(Daily, 0.87) = \{te_i \mid i \in \{Annual, Monthly, Weekly, Daily, Hourly, Instant\}, \frac{duration(Daily)}{duration(te_i)} \geq 0.87\}.$$

The duration is calculated in hours. `data:Collection`s that have a `TemporalExtent` that is in the result set R_{te} satisfy this filter.

6.2.6.3 Filter currentness

The *filter currentness* provides the ability to filter `data:Collection`s by its currentness with the options *historic* and *latest*. The options refer to whether the satellite or airborne mission that provides the actual data is

still running and delivering data, or already has been ended and no recent data is available. This filter can only be matched exactly as its possible values are mutually exclusive.

6.2.6.4 Filter quality

The *filter quality* filters a `data:Collection` by its `Quality` and consists of the three distinct options *HighQuality, MediumQuality* and *HighQuality*. This filter represents the fact that some `data:Collection`s have a well-known systematic or numerical error or uncertainty, for instance a deviation of 1% in their data with respect to other validation measurements. There are more qualitative `data:Collection`s with smaller error or a higher accuracy. The filter is matched according to the taxonomy of the OWL class `Quality` (see figure 5.8). Data from all `Quality` levels is found if the selection is *LowQuality*, data with *MediumQuality* or *HighQuality* is found if the selection is *MediumQuality*, and only *HighQuality* data will be found if the selection is *HighQuality*.

6.2.6.5 Filter resolution

The *filter resolution* filters a `data:Collection` by its `Resolution`. The possible options are *HighResolution, MediumResolution*, and *LowResolution*. For the prototype implementation in the scope of this diploma thesis, only these three subjective values have been implemented to provide a limited but functional support for this filter concept. The filter is matched similar to the quality filter according to the taxonomy of the OWL class `Resolution` (see figure 5.7). Data from all `Resolution` levels is found, if the selection is *LowResolution*, data with *MediumResolution* or *HighResolution* is found if the selection is *MediumResolution*, and only *HighResolution* data will be found if the selection is *HighResolution*.

6.2.6.6 Filter processing level

The *filter processing level* filters a `data:Collection` by its `data:ProcessingLevel`. The possible options are *L0, L1, L1B, L2, L3* or *L4*. The higher the `data:ProcessingLevel` of the `data:Collection`, the more processing with the data has taken place. The filter is matched according to the taxonomy of the OWL class `data:ProcessingLevel` (see figure 5.9). Data from all `data:ProcessingLevel`s is found if the selection is *L0*, data with the levels *L1, L1B, L2, L3* or *L4* is found for the selection of *L1*, data with *L1B, L2, L3, L4* is found if the selection is *L1B* and so on. The selection of the level *L4* delivers only *L4* data.

6.2.7 Invoking EOWEB `URL`s

The search results contain the URLs of each `data:Collection` in the EOWEB system (see figure 6.10, 2). The links are generated from the information contained in the ontology and from the filter settings applied by the user. As such, the EOLOsearch web demonstrator serves as a *frontend for a frontend*, as the links can be clicked directly from the results list and are employed to control the EOWEB applet. For instance, the coordinates typed in for the *filter geographical extent* (cf. section 6.2.6.1) in the filter mode *GeoBox* are reused for the selection of a bounding box in EOWEB by generating an EOWEB link, that passes the coordinates as arguments to the EOWEB applet. The results can be selected or deselected (see figure 6.10, 1). If a collection is selected, the EOWEB multilink (see 6.10, 3) is re-generated to include the selected collection in the multilink. Thus, the multilink can be used to select several collections simultaneously in the EOWEB applet.

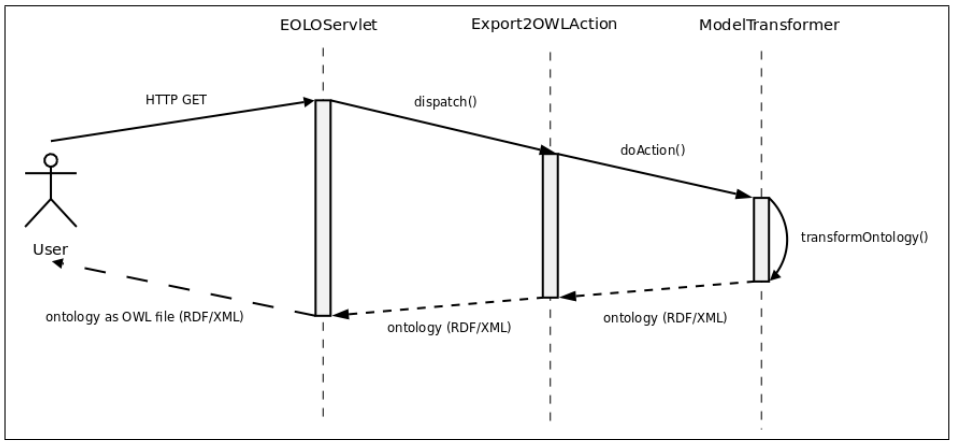

Figure 6.16: Request/response sequence for exporting the ontology

6.2.8 Exporting the EOLO ontology

Exporting the EOLO ontology is essential for collaborative developments or usage of the ontology. The ontology can be exported into an OWL ontology file in RDF/XML syntax which is widely accepted by most ontology editors. Figure 6.16 illustrates that the request `http://zebra:8080/EOLOsearch/servlet?action=export2OWL` is dispatched by the `EOLOServlet` to the `controller.action.Export2OWLAction`. Then, the ontology model retrieval from the *ontologyDB* and its conversion into the RDF/XML syntax is started by the `model.db.ModelTransformer`. These tasks are performed by the usage of the Protégé-OWL API that creates a high-level object-oriented ontology representation from the low-level RDF statements of the ontology. Finally, the ontology is returned to the user by creating a download link.

6.2.9 Reusing business logic for `data:Collection` retrieval

The business logic of the model can be used in two ways, either by referencing the Java project and delegating to the existing business logic in the Java classes, or by communicating with the `EOLOServlet` through `HTTP`

GET parameters in the URL. In the following, the second approach will be explained, as it is more platform-independent and requires less prerequisites. However, information needed for direct references inside other Java code can be obtained from the JavaDoc of this application. The EOLOServlet is controlled through HTTP GET parameters in the invocation URL. Thus, the invocation of the URL `http://zebra:8080/EOLOsearch/servlet?action=searchCollections&concepts=http://www.dlr.de/ontologies/EOLO.owl!WISDOM` instructs the controller to dispatch the request to the `controller.action.SearchCollectionsAction` that searches for `data:Collections` for the `ApplicationDomain` referenced by the URI `http://www.dlr.de/ontologies/EOLO.owl#WISDOM` that identifies the `ApplicationDomain` in the EOLO ontology.[32] The result of this invocation is an XML document containing a list of `collections`. Each `collection` has a `label` that is displayed, an `uri` for unique identification in the ontology, and the associated `eoweburl` that links to the EOWEB system. Figure 6.17 illustrates the resulting XML data for the above request that contains nine `collection` elements. Similarly, to filter the `data:Collections` for the `Phenomena Flood` and `MigrationOfPopulation`, a `HighResolution`, the `data:ProcessingLevel L1` and the filter operation `AND`, the following URL must be invoked: `http://zebra:8080/EOLOsearch/servlet?action=filterCollections&concepts=http://sweet.jpl.nasa.gov/2.0/hydroSurface.owl!Flood;http://www.dlr.de/ontologies/EOLO.owl!MigrationOfPopulation&filter_resolution=true&filter_processing_level=true&resolution=HighResolution&processingLevel=L1&filter_op=0`. The request is dispatched to the `controller.action.FilterCollectionsAction` which

[32] The URL that is sent to the servlet contains an exclamation mark "!" instead of the hash sign "#" in the `ApplicationDomain URI`, because the hash sigh "#" cannot be sent in HTTP GET parameters to the used application server. On the server side, the "!" is replaced by a "#", to conform to the URIs of the concepts in the EOLO ontology.

```
<collections>
  <collection>
    <label>AQUA.MODIS</label>
    <uri>http://www.dlr.de/ontologies/EOLO.owl!AQUA.MODIS</uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=AQUA.MODIS&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>GLOBE.DEM</label>
    <uri>http://www.dlr.de/ontologies/EOLO.owl!GLOBE.DEM</uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=GLOBE.DEM&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>NOAA.AVHRR.NDVI</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!NOAA.AVHRR.NDVI
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=NOAA.AVHRR.NDVI&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.Imaging</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.Imaging
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.Imaging&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.L1b-High-Resolution-Spotlight</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.L1b-High-Resolution-Spotlight
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.L1b-High-Resolution-Spotlight&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.L1b-ScanSAR</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.L1b-ScanSAR
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.L1b-ScanSAR&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.L1b-Spotlight</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.L1b-Spotlight
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.L1b-Spotlight&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.L1b-Stripmap</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.L1b-Stripmap
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.L1b-Stripmap&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.Non-Imaging</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.Non-Imaging
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.Non-Imaging&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
</collections>
```


Figure 6.17: `data:Collections` returned as XML data for the invocation of the URL `http://zebra:8080/EOLOsearch/servlet?action=searchCollections&concepts=http://www.dlr.de/ontologies/EOLO.owl!WISDOM`

```
<collections>
  <collection>
    <label>NOAA.AVHRR.NDVI</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!NOAA.AVHRR.NDVI
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=NOAA.AVHRR.NDVI&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
  <collection>
    <label>TSX-1.SAR.L1b-High-Resolution-Spotlight</label>
    <uri>
      http://www.dlr.de/ontologies/EOLO.owl!TSX-1.SAR.L1b-High-Resolution-Spotlight
    </uri>
    <eoweburl>
      https://centaurus.caf.dlr.de:8443/eoweb-ng/template/default/welcome/entryPage.vm?AppletTab=Catalogue&
      Service=TSX-1.SAR.L1b-High-Resolution-Spotlight&QueryMode=Advanced&autoSearch=no
    </eoweburl>
  </collection>
</collections>
```


Figure 6.18: `data:Collections` returned as XML data for the invocation of the URL `http://zebra:8080/EOLOsearch/servlet?action=filterCollections&concepts=http://sweet.jpl.nasa.gov/2.0/hydroSurface.owl!Flood;http://www.dlr.de/ontologies/EOLO.owl!MigrationOfPopulation&filter_resolution=true&filter_processing_level=true&resolution=HighResolution&processingLevel=L1&filter_op=0`

applies the filter logic. The resulting two `data:Collections` are listed in figure 6.18. By the usage of `HTTP GET` invocation for the requests and `XML` data for the response, various applications can interface with the business logic of the developed model in a platform-independent way. The remaining implemented `Actions` can be controlled through `HTTP GET` parameters, as well, as is described in detail in appendix B.

6.3 Use cases for knowledge engineers in the EO datacenter domain

New satellite or airborne missions that produce new `data:Collections` emerge continuously in the EO domain. These `data:Collections` must be configured and maintained by the EO datacenter so that they can be used in existing applications and systems by various EO users. The maintenance includes the formalization and integration of the EO domain knowledge into

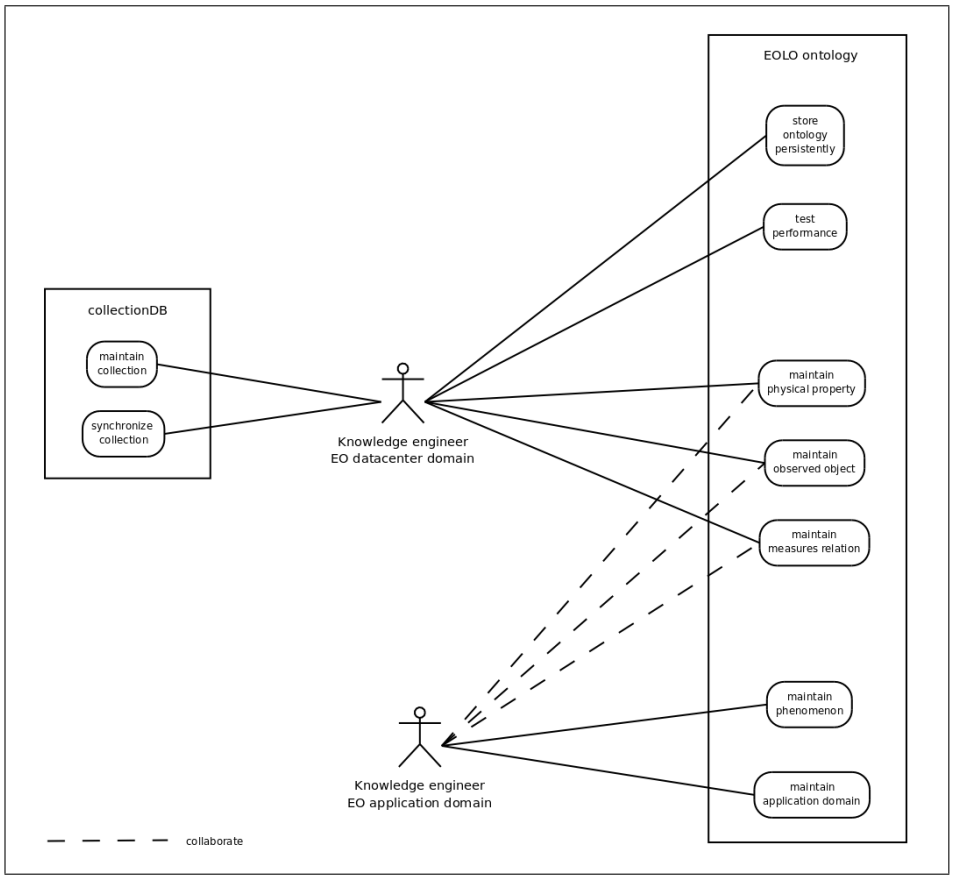

Figure 6.19: Use cases for knowledge engineers in the EO datacenter domain and the EO application domain

the ontology with respect to the newly created `data:Collections`. The OWL classes and the corresponding properties which have relevance for the knowledge engineer in the EO datacenter domain are illustrated in the blue ellipse in figure 5.5. Furthermore, figure 6.19 lists the use cases of knowledge engineers for the EO datacenter domain and the EO application domain.

6.3.1 Storing the EOLO ontology persistently in a DBMS

The EOLO ontology consists of RDF statements that can be stored persistently in the *ontologyDB* database by applying various techniques. The simplest form is to store the RDF triples "[..] in a relational database

with a three-column schema"[1, p. 386] containing the *subject, property,* and *object* of each RDF triple. However, queries are then "[..] potentially very slow to execute [..] since there is only one single RDF table, and almost all interesting queries involve many self-joins over this table."[1, p. 386] A better approach that implements "[..] a fully vertically partitioned database on property value"[1, p. 387] creates "a two-column table for each unique property in the RDF dataset where the first column contains subjects that define the property and the second column contains the object values for those subjects."[1, p. 387] Combined with the usage of a column-oriented DBMS such approach has "superior scaling properties"[1, p. 387] in comparison to a row-based DBMS. For the EOLO Search web application, the Protégé-OWL API, a high level API for ontologies has been used to handle the low level access to the RDF statements and their persistent storage in the *ontologyDB*. Persistence is achieved by storing "the entire contents of the knowledge base [..] in a single table [..]"[33]. The usage of the Protégé-OWL JDBC backend is a pragmatic solution for this diploma thesis that should be evaluated in production environments due to performance drawbacks when using only a single database table.

In order to store a file-based ontology persistently in a relational DBMS with the Protégé-OWL JDBC backend[34], the ontology file must be loaded into the Protégé-OWL ontology editor (see 6.20). Then, the ontology must be converted into an "OWL / RDF Database" project by selecting the menu item "Convert project to format..." from the "File" menu and setting the database credentials (see figure 6.21). After clicking the "OK" button, the ontology is stored into the specified database and the settings are saved to

[33] `http://protege.stanford.edu/doc/design/jdbc_backend.html`, last accessed: July, 27, 2009

[34] Detailed instructions on the usage of the database backend functionality in Protégé can be found in the Protégé-Wiki at `http://protegewiki.stanford.edu/index.php/Working_with_the_Database_Backend_in_OWL`

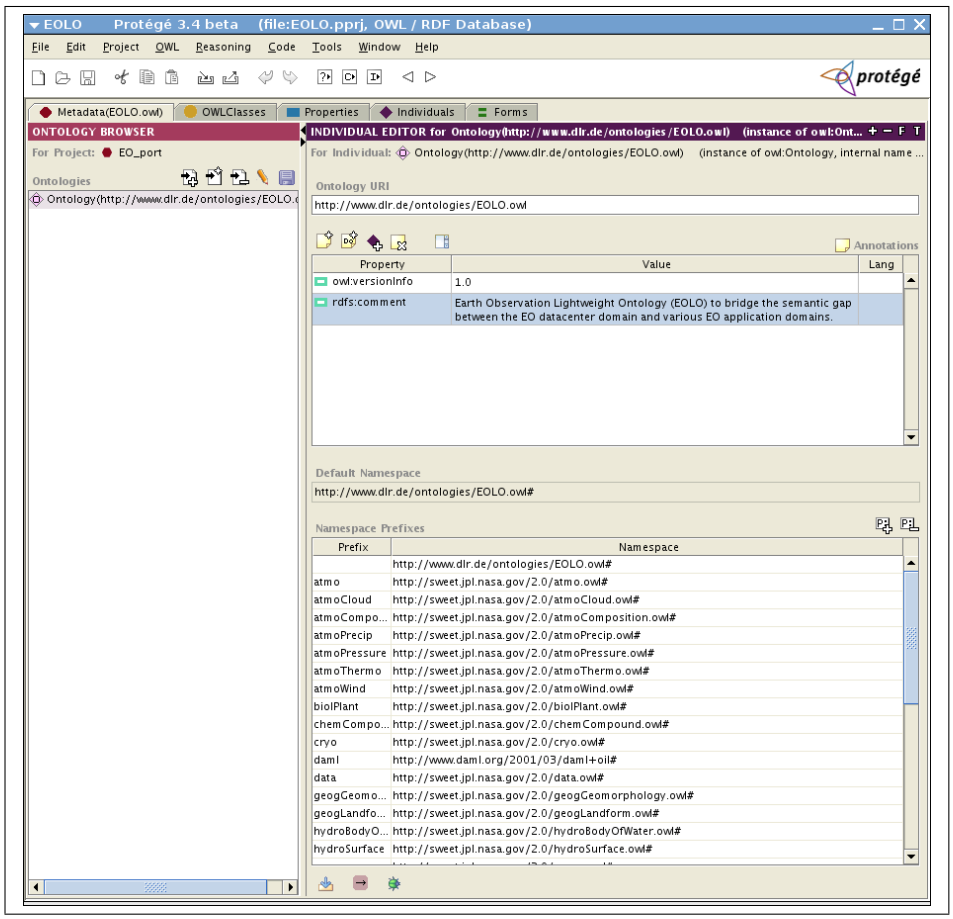

Figure 6.20: Opening the EOLO.pprj file with the Protégé-OWL ontology editor

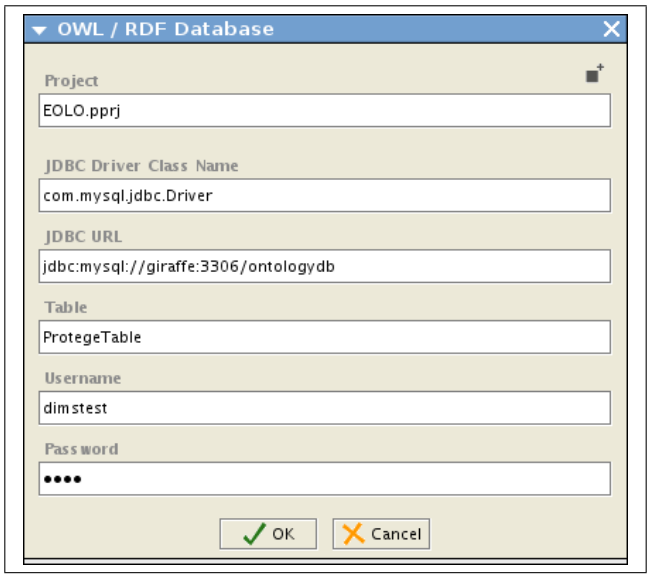

Figure 6.21: Storing a file-based ontology persistently in a DBMS with the Protégé-OWL ontology editor

a Protégé project file (.pprj). For the EOLO ontology in the *ontologyDB*, the EOLO.pprj file defines these settings and is contained in the package `model.db`. To port the ontology to another database, the menu item "Save Project As..." from the "File" menu must be clicked and the new database credentials have to be inserted (see figure 6.21).

6.3.2 Maintaining `data:Collections`

The maintenance of `data:Collections` in the EOLO ontology can be performed in two steps that assure consistency between the *collectionDB* and the *ontologyDB*. The maintenance is started by configuring the `data:Collection` and its properties in the *collectionDB* with a SQL client or an existing operating tool. The operator can set the `type` of the `data:Collection` (subclasses of `data:Collection`, its `Resolution`, `Quality`, `data:ProcessingLevel`, `GeoExtents`, `TemporalExtents`, `EOWEBurl`, and its `endTime` (represents the fact if the mission that produces

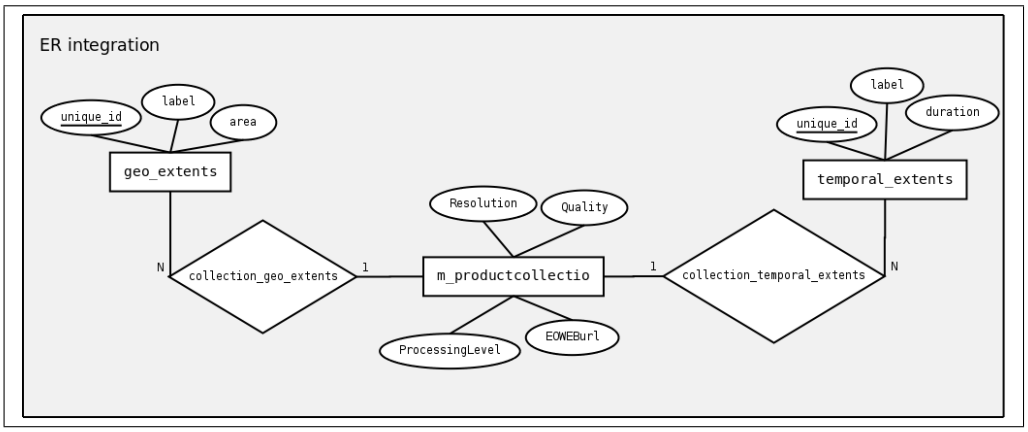

Figure 6.22: Extended ER model for the *collectionDB*

EO data for this `data:Collection` has been terminated, already). Next, the synchronization between the *collectionDB* and the *ontologyDB* must be started that transfers the `data:Collection` configuration into the ontology (cf. section 6.3.3). After the successful synchronization, the `data:Collection` is available for reasoning tasks in the ontology.

6.3.3 Synchronizing `data:Collections` from the *collectionDB* to the *ontologyDB*

Collection metadata from the *collectionDB* is used by EO datacenter operators primarily in various operating tools and legacy systems to query information about *Collections*. The EOLO ontology uses this metadata information, as well and adds some supplementary information to the existing definitions into the *ontologyDB*. However, the newly created meta information can also be useful for datacenter operators in the near future. Therefore, the *collectionDB* has been extended by integrating attributes, entity types, and relation types into the ER model of the *collectionDB*. The implemented extensions of the ER model are illustrated in figure 6.22. The OWL classes `Resolution`, `Quality`, and `data:ProcessingLevel` and the *EOWEBurl* have been added as SQL attributes to the existing database

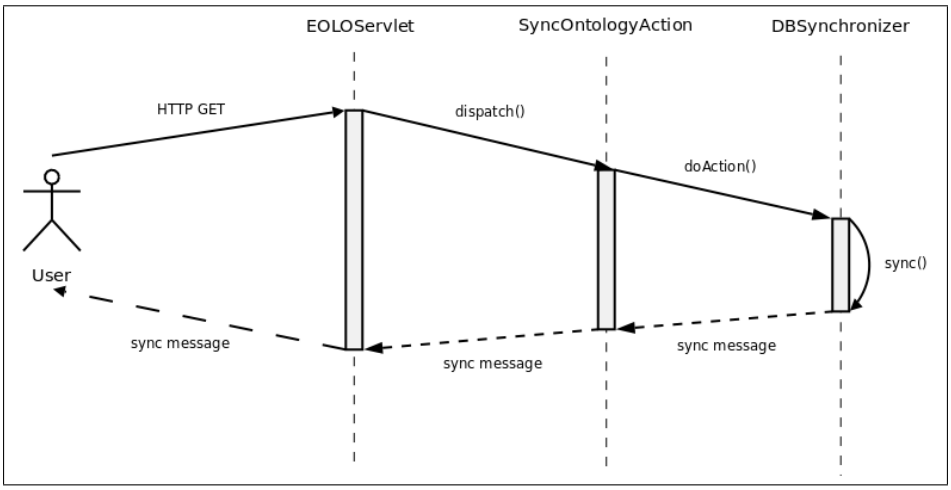

Figure 6.23: Request/response sequence for synchronizing the *collectionDB* with the *ontologyDB*

entity type *m_productcollectio* as their corresponding OWL properties only have 1:1 multiplicity (cf. sections 5.4.1.2, 5.4.1.3, 5.4.1.4, and 5.4.1.7). The classes `GeoExtent` and `time:TemporalExtent` have been modelled as entity types with corresponding relation types modelling the 1:N multiplicity between `data:Collection` and `GeoExtent` and `time:TemporalExtent`, respectively (cf. sections 5.4.1.5 and 5.4.1.6). The newly created entity types contain an attribute `unique_id` which is used to integrate the entity types into the metamodel of the DIMS Product Library. In order to keep the existing systems running with the *collectionDB* and benefitting from the supplementary knowledge contained in the *ontologyDB*, it has become essential to implement a synchronization tool between the *collectionDB* and the *ontologyDB*, so that the `data:Collection`s are consistent in both, the collectionDB and the ontologyDB. The alternative to this approach would be to re-model the schema of the *collectionDB* to cover all OWL classes, properties, OWL individuals, and axioms which are contained in the *ontologyDB*. Such tasks would not only make the *ontologyDB* useless, but one would loose the ability to perform OWL

reasoning and the schema flexibility of OWL ontologies. As described in the previous section, the statements of the EOLO ontology are mapped persistently to a database table in the *ontologyDB*. Thus, the implementation of the synchronization tool concentrated on the generation of statements for the *ontologyDB* that describe the attributes which are modelled in the *collectionDB*. The synchronization tool can be started by invoking the URL `http://zebra:8080/EOLOsearch/servlet?action=syncOntology` (see figure 6.23). The servlet dispatches the request to the `controller.action.SyncOntologyAction` which delegates to the `model.db.DBSynchronizer` class. There, the synchronization between the *collectionDB* and *ontologyDB* is performed on a SQL-basis. Finally, the resulting synchronization message about a successful or unsuccessful synchronization is returned to the client.

6.3.4 Testing the performance of synchronization and reasoning

As the ontology evolves with an increasing number of contained knowledge, performance measurements can assist the knowledge engineer in the development. The implemented performance tests simulate the prospective insertion of `data:Collection`s into the *collectionDB*, their synchronization into the *ontologyDB* and the reasoning about them. The randomly created `data:Collections` are deleted from the *collectionDB* and the *ontologyDB* after the tests have been performed. Furthermore, the reasoning has been started on a subset of five fixed phenomena, that simulates the user's selection. The tests can be started by invoking the URL and setting the `maxCollections` parameter `http://zebra:8080/EOLOsearch/servlet?action=testPerformance&maxCollections=150` (see figure 6.24). The request is dispatched by the `EOLOServlet` to the `controller.action.PerformanceTestAction` class which delegates

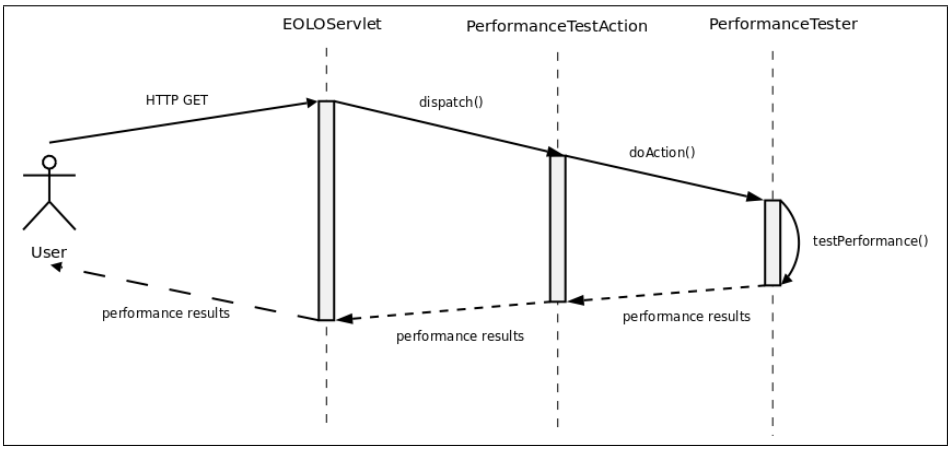

Figure 6.24: Request/response sequence for testing the Performance of the synchronization and reasoning tasks

to the `model.performancetesting.PerformanceTester`. There, the method `testPerformance()` is called that launches three iterative tests for a maximum of 150 random `data:Collection`s, starting with 50 and increasing by 50 in each iteration (i.e. 50, 100, 150). At the end of the tests, the resulting times to perform synchronization and reasoning tasks are presented in a table.

6.3.5 Maintaining `PhysicalProperties` and `ObservedObjects`

The previous sections handled the `data:Collection` maintenance that is performed in the *collectionDB* and subsequently synchronized to the *ontologyDB*. The following sections describe changes in the ontology model itself to reflect a change in `data:Collection` measurement capabilities and an improved understanding of the EO domain. The OWL classes `PhysicalProperties` and `ObservedObject`s are taxonomies of imported OWL classes from the SWEET ontologies (see figures 5.11 and 5.10). These have been selected from the large amount of available classes in the SWEET ontology to describe the characteristics of the sensors which deliver EO data for the `data:Collection`s. Thus, new sensor

characteristics entail adaptions in both classes. For optimal collaborative effects and reuse of common terminology and semantics, it should be evaluated if a new physical property or observed object is already available in an authoritative ontology such as SWEET. Large ontologies can be browsed with an ontology visualization tool (for example the CMAP Tools knowledge modeling kit[35] to find existing definitions more easily (see figure 6.25). The workflow for adding new `PhysicalProperties` and `ObservedObjects` is very similar and is demonstrated by adding the concept `Cirrostratus` from the `http://sweet.jpl.nasa.gov/2.0/atmoCloud.owl` SWEET ontology as subclass of the `ObservedObject` class in the EOLO ontology. Firstly, the EOLO is loaded into the Protégé-OWL ontology editor by opening the file EOLO.pprj. Next, the SWEET ontology that contains the new concept is imported into the EOLO ontology by creating a new entry for its namespace prefix in the "Metadata tab" (see selected entry in figure 6.26). Then, the observed object is added as a subclass of the `ObservedObject` class by selecting its desired location in the existing taxonomy in the "OWLClasses" tab (here `atmoCloud:Cloud`), clicking on the "create subclass icon" (see red rectangle in figure 6.27) and assigning a name to it. The name for the sample concept is set to `http://sweet.jpl.nasa.gov/2.0/atmoCloud.owl#Cirrostratus` and consists of two parts: the namespace of its originating ontology and its actual terminological name separated by the `#` hash sign. `PhysicalProperties` can be added to the EOLO ontology analogously with the only difference that new concepts are added as subclasses of `PhysicalProperty`. Furthermore, the modelled `ObservedObject`s and `PhysicalProperties` can be moved into new subclasses for a better maintainability. However, this must be performed with respect to `measuresRelation`s which aggregate these classes and are used for reasoning tasks.

[35] `http://cmap.ihmc.us`

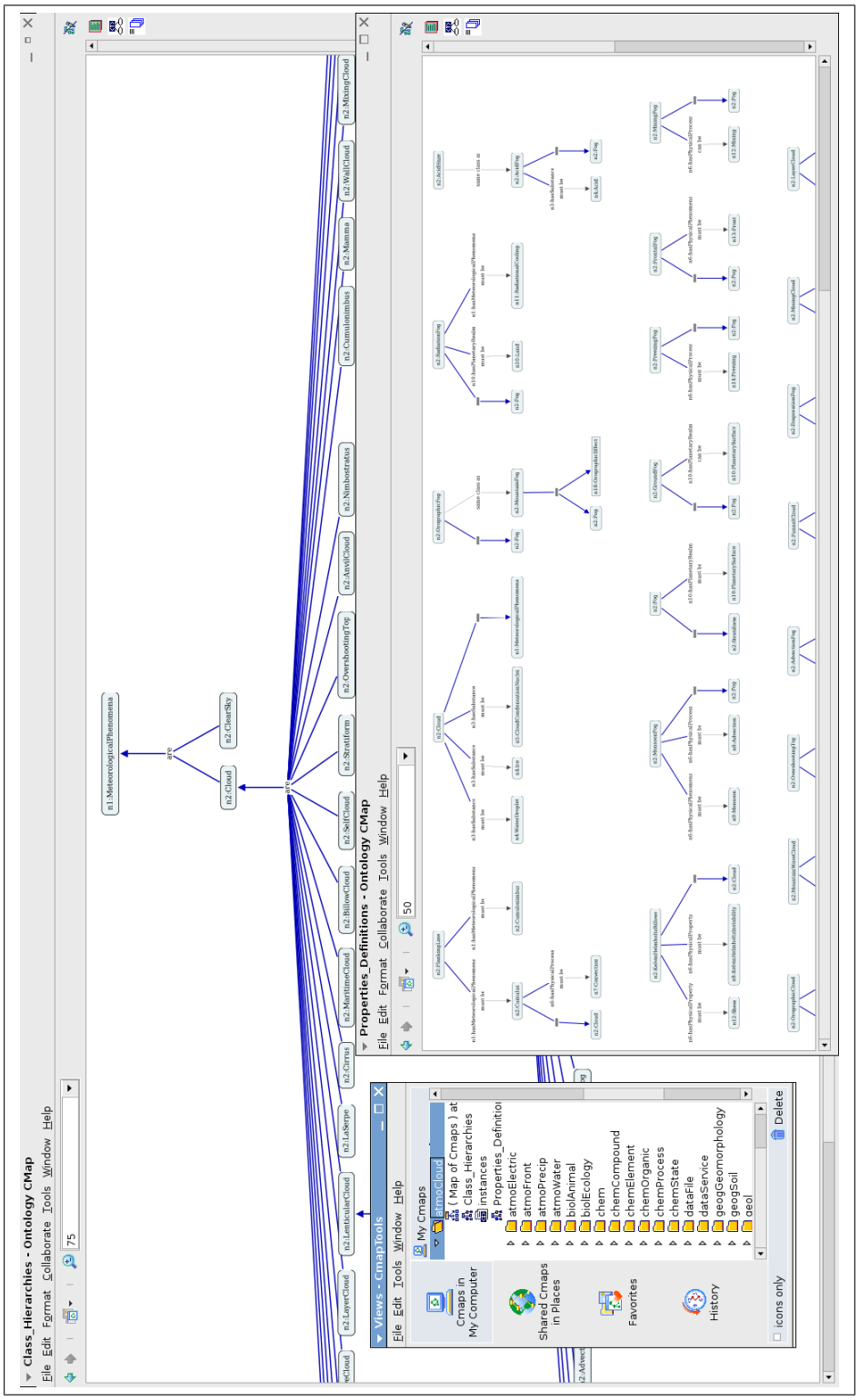

Figure 6.25: Visualizing the SWEET ontology `http://sweet.jpl.nasa.gov/2.0/atmoCloud.owl` in the CMAP Tools knowledge modeling kit

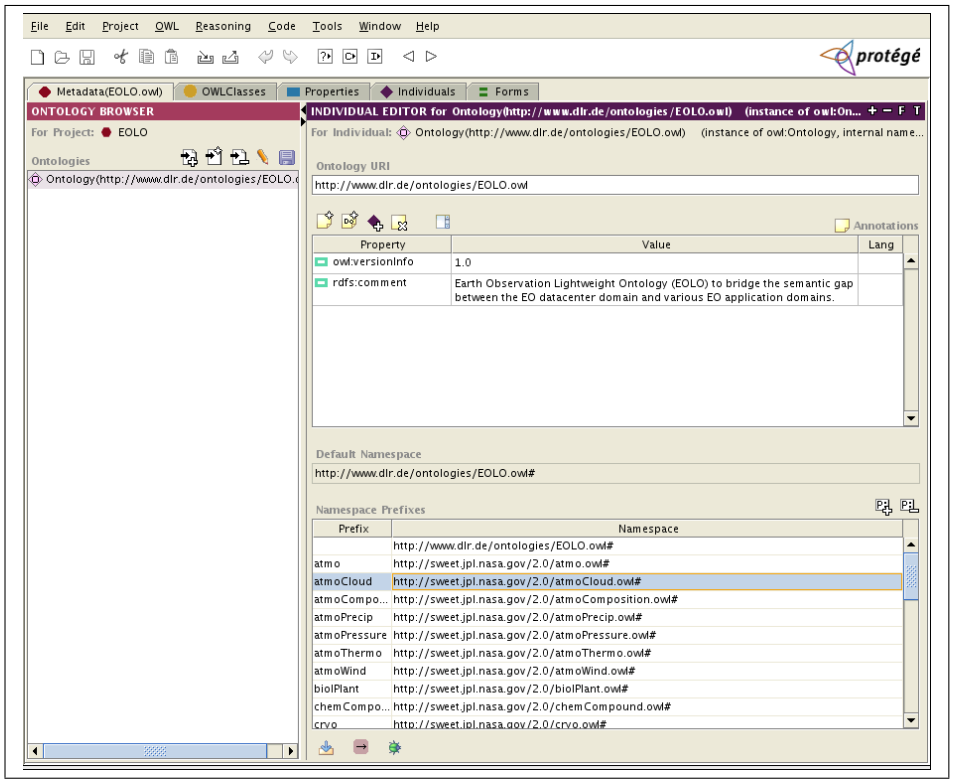

Figure 6.26: Importing the SWEET ontology `http://sweet.jpl.nasa.gov/2.0/atmoCloud.owl` into the EOLO ontology

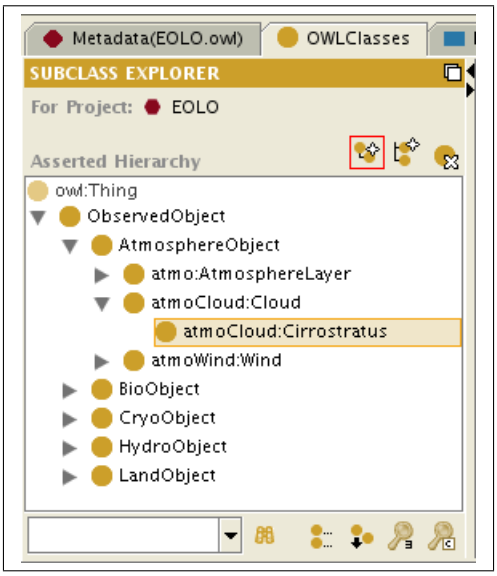

Figure 6.27: Adding the OWL class `atmoCloud:Cirrostratus` to the OWL class `ObservedObject`

6.3.6 Maintaining `measuresRelations`

The current `measuresRelations` cover several reasonable combinations of the classes `ObservedObject` and `PhysicalProperty`. This aggregation class is strongly connected to the features and capabilities of the sensor attached to a satellite or airplane. Therefore, sensor-based updates of its measurement capabilities can be modelled into the `measuresRelation` class. New `measuresRelations` can be modelled as OWL classes under one of the `measuresRelations` `measuresAtmosphericProperties`, `measuresHydrosphericProperties`, or `measuresLithosphericProperties`. Moreover, a new subclass can be created with the Protégé-OWL ontology editor. After creating the OWL class, its *necessary* conditions have to be set by defining axioms which describe the aggregation between `ObservedObject` and `PhysicalProperty`. *Necessary* in this context means, that it is necessary for an OWL individual that has the type `measuresTemperatureOfRiver` to satisfy the

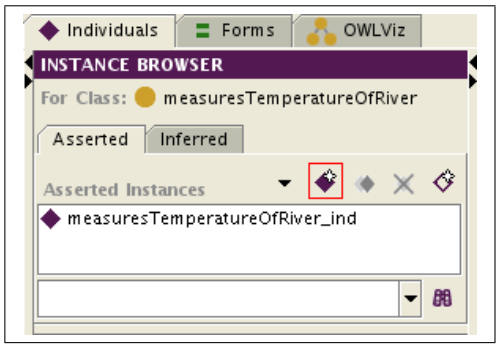

Figure 6.28: Added an OWL individual `measuresTemperatureOfRiver_ind` as instance of OWL class `measuresTemperatureOfRiver`

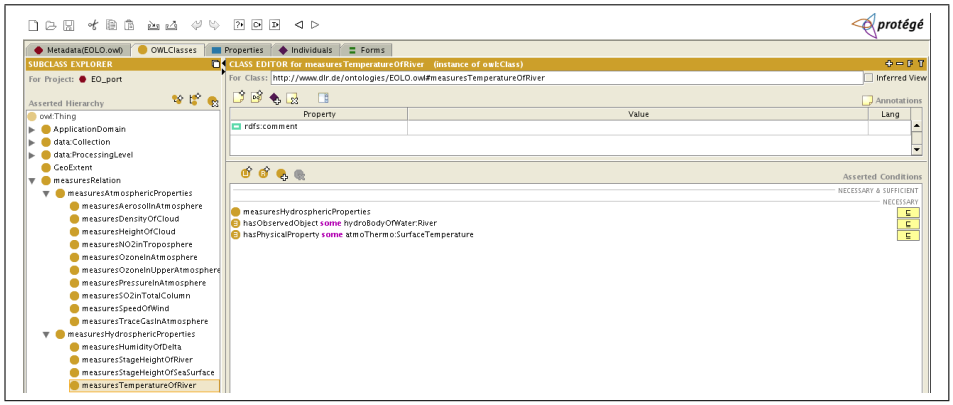

Figure 6.29: Modelling *necessary* condition axioms for the OWL class `measuresTemperatureOfRiver`

conditions defined by the axioms. Thus, the axioms `hasObservedObject some hydroBodyOfWater:River` $\wedge$ `hasPhysicalProperty some atmoThermo:SurfaceTemperature` formalize the semantics of the `measuresTemperatureOfRiver` aggregation illustrated in figure 6.29. The namespaces `hydroBodyOfWater` and `atmoThermo` refer to the imported SWEET ontologies. The last step when integrating new `measuresRelation`s, is to insert an OWL individual as an instance of the newly created class which is needed for reasoning tasks by clicking the "create instance icon" in the "Individuals" tab (see red rectangle in figure 6.28).

6.4 Use cases for knowledge engineers in the EO application domain

The modelling of an application domain starts with the phenomenon of interest (see green ellipse in figure 5.5). While some anthropogenic or natural phenomena are occuring repeatedly, others cannot be predicted or foreseen. Therefore, the adaption to new phenomena is an important step in the ontology maintenance. An application domain concentrates on one or several phenomena and may impose certain requirements on them which can be modelled through *logical axioms* for the OWL classes `Phenomenon` and `ApplicationDomain`.

6.4.1 Maintaining `Phenomena`

The integration of a new `Phenomenon` is performed by creating an OWL class under one of the classes `AtmosphericPhenomenon`, `BiosphericPhenomenon`, `HydrosphericPhenomenon`, and `LithospericPhenomenon`. To the newly created class a *necessary & sufficient* condition axiom must be added (cf. section 5.4) which describes the desired measurement requirements for this phenomenon. Considering the `HydrosphericPhenomenon planetClimate:GlobalWarming`, the following axioms are defined: `(measures some measuresTemperatureOfLand)` $\vee$ `(measures some measuresTemperatureOfRiver)` $\vee$ `(measures some measuresTemperatureOfSea)`. The modelling of `Phenomena` is illustrated in figure 6.30.

6.4.2 Maintaining `ApplicationDomains`

An application domain can be integrated into the EOLO ontology by creating an OWL class under the main class and defining *necessary & sufficient* condition axioms declaring which `Phenomena`

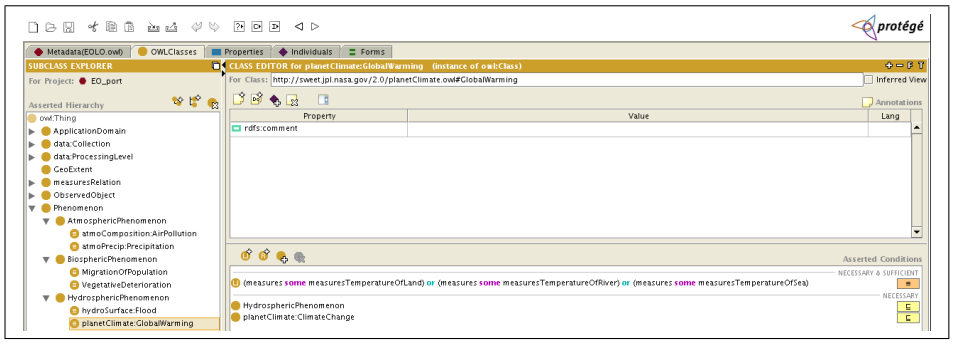

Figure 6.30: Modelling *necessary and sufficient* condition axioms for the OWL class `planetClimate:GlobalWarming`

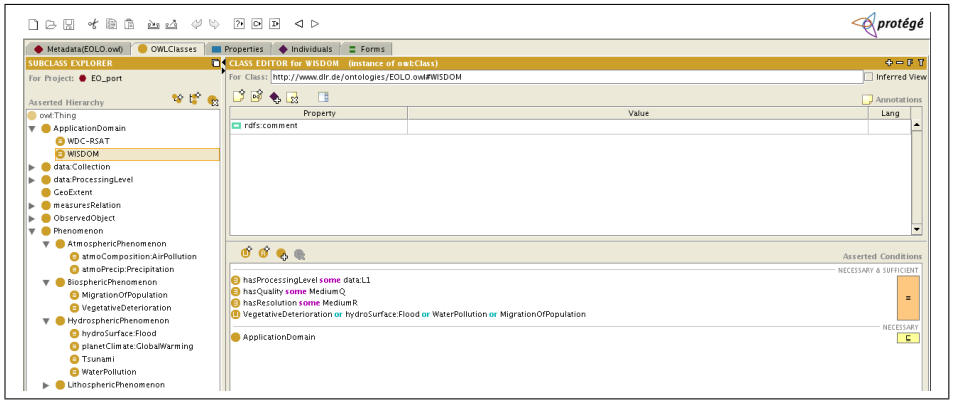

Figure 6.31: Modelling *necessary and sufficient* condition axioms for the OWL class `WISDOM`

are referred by it and which `data:Collection` properties are required for the `ApplicationDomain`. The `ApplicationDomain WISDOM` declares the following axioms: `hasProcessingLevel some data:L1` $\wedge$ `hasQuality some MediumQ` $\wedge$ `hasResolution some MediumR` $\wedge$ `(VegetativeDeterioration` $\vee$ `hydroSurface:Flood` $\vee$ `WaterPollution` $\vee$ `MigrationOfPopulation)`. The modelling of the axioms is illustrated in figure 6.31. As indicated by the dashed lines in figure 6.19, knowledge engineers from the EO application domain can collaborate with knowledge engineers from the EO datacenter domain in the definition of `ObservedObjects, PhysicalProperties`, and `measuresRelations`

as these concepts are familiar to them because they often result from application domain research and activities.

6.5 Deployment at the DFD

This section documents the deployment of the EOLO ontology and its web demonstrator into the infrastructure at the DFD (see figure 6.32) The EOLO search web demonstrator is deployed into the JBOSS application server on the host `zebra`. The application is packed into the WAR archive `EOLOsearch.war` that can be copied to the application server's deploy directory. The EOLOsearch root context is `http://zebra:8080/EOLOsearch`. Both, the *ontologyDB* and the *collectionDB* are deployed at the server host *giraffe*. The WAR archive contains all 3rd party libraries that are required by its business logic such as the JDBC drivers for the *MySQL* and *Informix* databases, the Protégé-OWL API for the high-level access to the data encoded in OWL format, the Pellet reasoner API, and the Log4J-API for logging output (cf. appendix C). The EOLOsearch web demonstrator can be configured via property files without any code modifications. The WAR archive contains the property files `informix_db.properties` and `mysql_db.properties` for the connection credential and setup configuration of the *ontologyDB* and the *collectionDB*. Furthermore, the names of the SQL attributes and tables of the entity and relation types that occur in the SQL statements can be adjusted in the `db_schema.properties` files. Additionally, global constants about the default namespace of the EOLO ontology and the used reasoner with its URL and port settings can be configured in the `global.properties` file. All property files in printed form and a list of the used libraries can be found in appendix A and C.

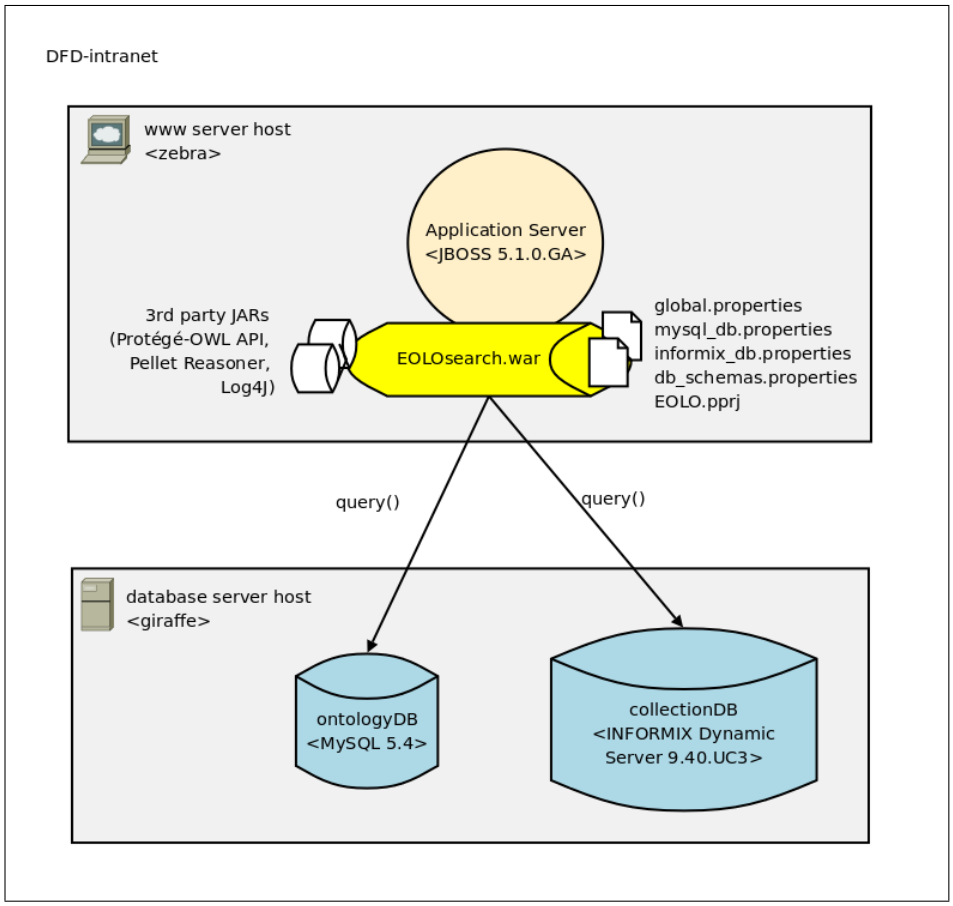

Figure 6.32: Deployment of the EOLOsearch web demonstrator at the DFD

Chapter 7

Evaluation

The modelling of the EOLO ontology for the EO domain has been discussed in previous chapters. This chapter covers the evaluation of the modelled EOLO ontology and its frontend, the EOLO Search web application. First of all, the overall feasibility of such semantic modelling approach for the EO domain, and specifically for the DFD is discussed. Next, the workflows for experts in both worlds, the EO application domain experts and the EO datacenter experts are illustrated. Finally, prospective projects and ideas are outlined.

7.1 Feasibility

The feasibility of the modelling approach is essential to its usage and application. Several aspects must be considered when evaluating the feasibility. In general, the feasibility of a software system depends on the satisfiability of its quality criteria. Moreover, the specifics of the semantic modelling approach should be compared to the traditional ER-based modelling of information systems. In the following subsections, the feasibility is discussed in more detail.

7.1.1 Quality criteria

During the development of the EOLO ontology and its web demonstrator, it has been focussed on the quality criteria *scaling performance, maintenance effort, and search effectivity.*

7.1.1.1 Scaling performance

Number of collections	**synchronization [s]**	**reasoning [s]**
50	3.479	4.854
100	4.861	10.034
150	6.176	17.445
200	7.577	24.987
250	6.52	35.715
300	7.3	49.694
350	8.493	68.377

Table 7.1: Performance test results for synchronization and reasoning of `data:Collection`s

The performance of the EOLO web application refers to its scalability for a continuously increasing number of `data:Collection`s. Generally, it can be assumed, that the more `data:Collection`s are available at the DFD, the longer it will take to perform search and other activities upon it. The tests have been performed on an Intel® Core™ 2 CPU T7200 with 2.00 GHz clockspeed and 2 GB of RAM. The performance analysis is separated into two tasks, the *synchronization of* `data:Collection`*s* and the *reasoning for* `data:Collection`*s*. The tests have been iterated from 50 `data:Collection`s to 350 `data:Collection`s with steps of 50. The results in figure 7.1 and table 7.1 show, that the synchronization is linear whereas the reasoning task is exponential. Currently, the EO datacenter at

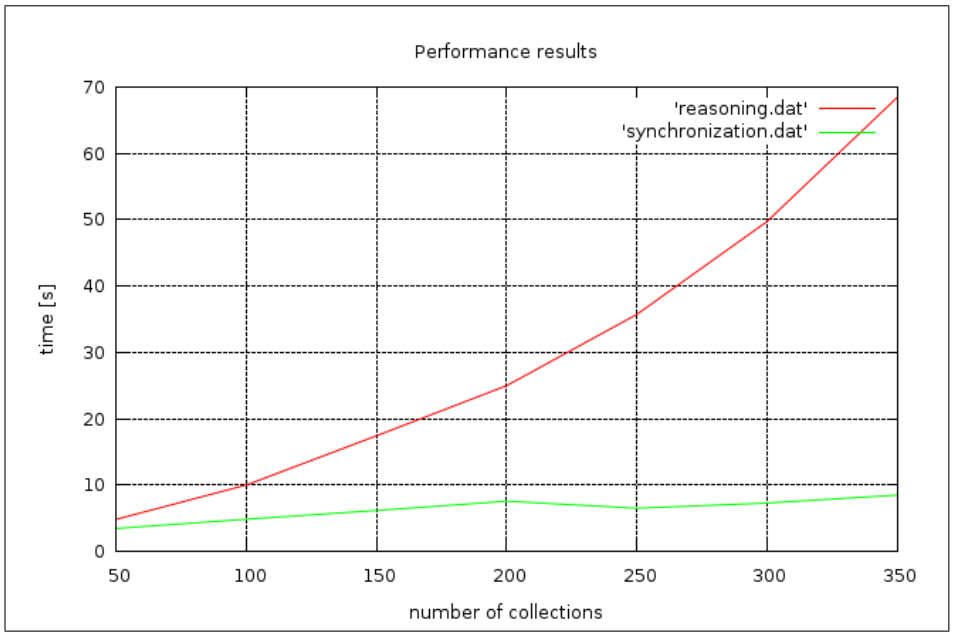

Figure 7.1: Performance results for the synchronization and reasoning of `data:Collections`

the DFD maintains about 80-100 `data:Collections` in the *collectionDB*. The fact, that the reasoning time takes over 20 seconds for more than 150 `data:Collections` seems to be a major disadvantage of the semantic modelling approach for the EO domain as it hinders the deployment in interactive user interfaces. However, by comparing it with the traditional ER-based modelling (cf. section 7.1.2) it reveals to be only a minor trade-off.

7.1.1.2 Maintenance effort

The EO domain is very dynamic and ever evolving due to new satellite or airborne missions and new application domains requiring EO data as their research base. Therefore, it is essential for the developed system to be extensible and easy to maintain. New `data:Collections` with existing axioms can be integrated easily into the EOLO ontology while preserving the axioms which classify the newly created `data:Collection`, dynamically

without any additional changes to the ontology. New measurement capabilities which are eventually introduced by a new sensor type can be modelled as new `measuresRelation`s at one single point in the ontology. Furthermore, new phenomena and application domains can be integrated into the ontology easily by creating the appropriate OWL classes with its appropriate axioms. After integrating a new *collection* into the *collectionDB* with existing operating tools used for *collection* maintenance, the synchronization of the *ontologyDB* can be started by invoking a single URL. Detailled workflows showing how to maintain the ontology have been described in the sections 6.3 and 6.4. Moreover, the reusability of previously defined concept definitions and axioms limit the necessary efforts when integrating new knowledge into the ontology. This is due to the evolvement of the ontology covering more and more concept definitions, properties and axioms from the EO domain. The process is accelerated by the collaborative modelling of knowledge engineers from diverse domains on the base of a formally defined model for their domains of interest.

7.1.1.3 Search effectivity

The search effectivity of the developed web application is a critical quality criteria. The question is how to measure the effectivity. The employment of metrics for the search term and its corresponding results would help in obtaining an objective and repeatable statement for the application precision. In the scope of this diploma thesis, only subjective feedback from application domain experts and datacenter experts can be used to argue about the search effectivity. Measurements of the search effectivity can be used to improve the modelled ontology for gaining better search results. As the EOLO ontology captures the knowledge of `ApplicationDomain`s, `Phenomena`, and its corresponding `data:Collection`s, the semantic links that exist between these concepts and its related concepts `Quality`,

`Resolution`, `ProcessingLevel`, `GeoExtent`, and `TemporalExtent` reflect the reality in the EO domain. Therefore, it is assumed that the search results which are computed by reasoning match semantically with a user's manual selection of `data:Collection`s. Various application domain users at the DLR have evinced their interest in the developed ontology. However, at the time of this writing, no user feedback has yet been submitted to the author which could have been used to start a new ontology modelling iteration.

7.1.2 OWL-DL vs. ER based modelling

The semantic modelling in OWL-DL has a drawback, the reasoning performance (cf. section 7.1). Therefore, one could think that the ER-based modelling should be preferred for gaining better scalability. Why is it still feasible to employ a DL-based modelling of the EO domain then? Considering the modelling primitives, "both in Description Logics and the ER model, the domain of interest is modelled through classes and relationships [..]"[5, p. 177]. Regarding the relationships in more detail, it is clarified that the "ER model allows relations of arbitrary arity, while in traditional Description Logics only unary and binary relations are considered."[5, p. 177]. An ontology design pattern for creating n-ary relations has been published by the W3C.[36] The schema-based flexibility introduced by the modelling facilities of OWL-DL ontologies can be illustrated by the following example: consider an entity type *collections* with an attribute *geoExtent*. After inserting several entities into the database table *collections*, it is realized that one collection should be able to have more than one geoExtent. The common solution to this issue is to create a relation type *collection_geoExtents* with 1:N multiplicity, to migrate the

[36] Defining N-ary Relations on the Semantic Web, `http://www.w3.org/TR/swbp-n-aryRelations/`, last accessed: July, 9th, 2009

geoExtents from the *collection* table into the new relation, and to remove the attribute *geoExtent* from the *collection* table. Moreover, the SQL statements for the application code must be adapted to reflect the changes of the database schema and perform the relevant joins. With semantic modelling based on OWL-DL, the *geoExtent* is related to the *collection* by the object property *hasGeoExtent*. The initial multiplicity 1:1 can be changed to 1:N by setting the property option *functional* to `false`. Additional geoExtents can be added as property fillers for the *hasGeoExtent* object property. Application code adjustments can be avoided when the initial code already loops over all property fillers. Continuing with the comparison between ER and DL modelling, one must recognize that with OWL-DL "[..] the schema (can) [..] be viewed and queried explicitly, something normally not available when using a raw DBMS directly."[5, p. 32] Moreover, in DL, the schema can be exchanged easily by the usage of the portable representation language OWL. Hence, collaborative effects of knowledge acquisition and management will be achieved easier with the DL-based modelling than with the traditional ER-based modelling. "The main difference between databases and knowledge bases is that while the former concentrate on manipulating large and persistent models of relatively simple data, the latter provide more support for inference-finding answers about the model which had not been explicitly told to it and involve fewer but more complex data."[5, p. 501]

Concluding the comparison between ER and OWL-DL modelling for the EO domain, it is observed that the modelling flexibility of the OWL-DL based modelling approach immensely outweighs its performance drawbacks because the model in the EO domain continuously evolves due to new application domains and satellite or airborne missions. However, the performance is a real problem regarding the usage scenarios of the ontology in various interactive interfaces. As a workaround for the performance issue,

an ER-based schema could be generated from the ontology and integrated into a more performant DBMS.

7.1.3 Modelling issues

During the modelling of the EOLO ontology the following modelling issue has been encountered. On the one hand, a very crisp modelling of a domain of interest results in very precise search results. However, reasoning that classifies concepts based on these definitions becomes useless, since the domain is modelled so precisely, that the result can be retrieved without any classifier via SPARQL[37] queries that process the RDF statements of the ontology. On the other hand, a vague modelling of the domain of interest employs the classification capabilities of the reasoner. In general, the search results will then be more coarse-grained and the information gain will decrease. This is a trade-off situation that is illustrated in figure 7.2. In the EO domain, it is difficult to employ subsumption that is solely based on physical properties like *temperature* or *height*, because of their occurrence in various `Phenomena` and `ApplicationDomains`. The different occurrences can be modelled by aggregations of the concepts `ObservedObject` and `PhysicalProperty` in the following ways: *temperature of sea, temperature of land, height of cloud, height of river*. However, the trade-off still exists as the *temperature of sea* can be observed for several phenomena (e.g. el Niño, algae bloom). Therefore, more concepts should be added which will create even better distinctions but will render the classifier more and more useless.

7.2 Bounding generalization

Some filter concepts provide a specified inclusion hierarchy for the inference of generalizations in a filter concept. For instance, the inclusion hierarchy

[37] SPARQL RDF Query Language `http://www.w3.org/TR/rdf-sparql-query/`

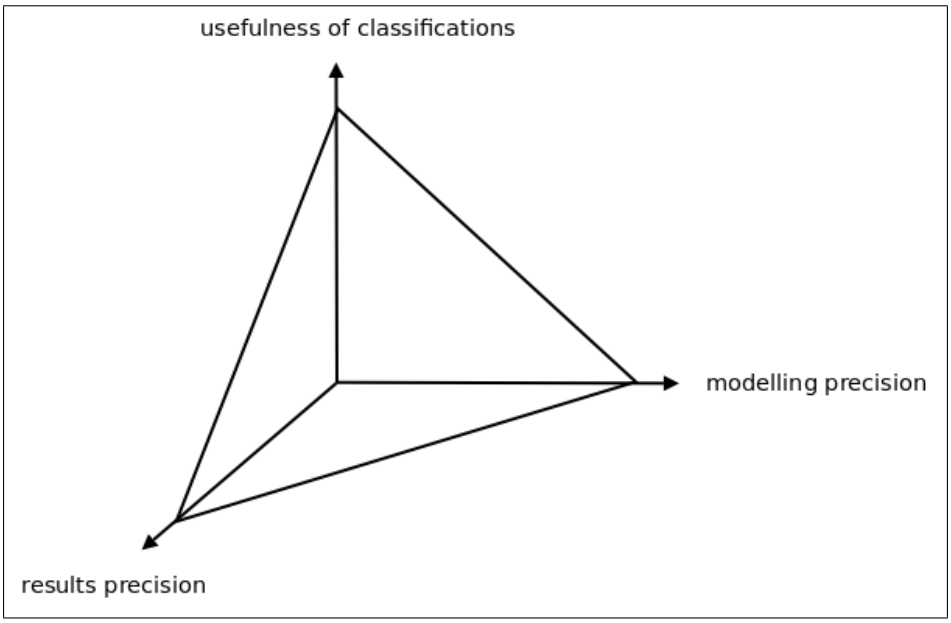

Figure 7.2: Trade-off situation between *modelling precision, usefulness of classifications*, and *results precision*

for the concept `Quality` is represented by its specified class taxonomy illustrated in figure 5.8. While the generalization up to the root class of the `Quality` concept is feasible, there are concepts where it is aspired to limit the generalization to implement only a bounded inclusion hierarchy that is dependent on a specific `data:Collection` or the user's selections. Considering the concept `GeoExtent` (cf. section 5.4.1.5), an inclusion hierarchy based on the percentual area inclusion can be calculated and thus the generalization can be bounded by the percentage parameter. Why is such bounding necessary? When a user searches for air pollution `data:Collection`s of the city Munich, a complete inclusion hierarchy for the area of Munich would let the reasoner infer, that Munich is also contained in a `data:Collection` with continental or global `GeoExtent`. Geographically speaking this is true, but for the application domain user focussing on Munich, it makes no sense to perform qualitative research activities on `data:Collection`s with a much bigger `GeoExtent` compared

to the actual `GeoExtent` of Munich, because the resulting data will be too unprecise. Thus, the inclusion hierarchy must be bounded to include only those `data:Collection`s where the percentual area inclusion of the `GeoExtent`s has at least some user-defined percentage.

Similar to the `GeoExtent` based approach to bound the inclusion hierarchy, a complete inclusion of the `TemporalExtent`s *Annual, Monthly, Weekly, Daily, Hourly,* and *Instant* are not desired. For instance, if the user is interested in `data:Collections` which are updated *Hourly* it makes little sense to present `data:Collections` which deliver data on an *Annual* basis due to their imprecision for the application context. However, *Instant* or *Daily* data will still be quite useful.

7.3 Interoperability of the EOLO ontology

The usage of the Web Ontology Language (OWL) and the Semantic Web for Earth and Environmental Terms (SWEET2.0 Beta) ontologies, renders the EOLO ontology both syntactically and semantically interoperable. The ontology covers knowledge from the EO domain, that can be used in several domains of interest and integrated into legacy systems. Collaborative effects will be most probably encountered, whether the ontology is used as a base for more abstract domain ontologies. The imported resources from the authoritative SWEET2.0 Beta ontologies serve as an interface for both, the EO datacenter domain and the EO application domains. The employment of accepted terminology from earth sciences and their domain-specific refinements modelled by taxonomies and properties create a common communication base for the domains and thus reasoning between them becomes a feasible task.

7.4 Future work

The work that results from this diploma thesis can be extended in various directions. This chapter lists some ideas for future work that improves the quality and usage scenarios of the EOLO ontology.

7.4.1 Competency Questions

The knowledge of the EO domain can be formalized into an ontology. The first question that falls into mind immediately is: does the ontology really reflect the EO world? The second question is: how can one measure the matching between the real world and its modelling in the ontology? The answers to both questions can be given by the employment of "Competency Questions"[12]. Domain-specific questions can be formulated that help to reveal modelling errors. Then, the questions are presented to domain experts to evaluate the modelled concepts in the ontology from their point of view. "The ontology engineer needs to check, [..] whether the ontology based application supports or answers the competency questions [..]"[22, p. 42]. Hence, the usage of *Competency Questions* can help to improve the quality and precision of the modelled domain ontology.

7.4.2 Spatial, temporal, and spatio-temporal reasoning

The EOLO ontology could be extended to support spatial, temporal, and spatio-temporal reasoning. An interval-based temporal logic that is "[..] both expressive and computationally feasible [..]"[3, p. 841] could be used for temporal reasoning. It contains thirteen possible interval-interval relationships listed in figure 7.3. Moreover, "Regional Connection Calculus 8 (RCC8)"[8] is a spatial logic that describes eight basic relations for connectivity of regions. "The RCC8 relations constitute a constraint language which is of fundamental importance."[8, p. 295] Considering

Relation	Symbol	Symbol for Inverse	Pictoral Example
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYYY
X finishes Y	f	fi	XXXX YYYYYYY

Figure 7.3: The thirteen possible relationships for the Allen Interval Algebra (source: [3, p. 835, figure 2])

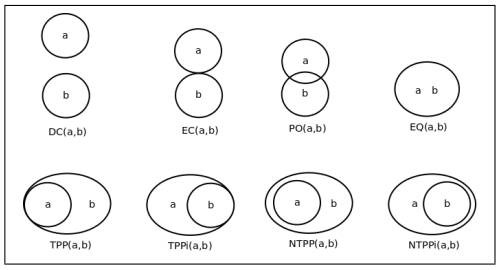

Figure 7.4: The eight relations of RCC8 (source: [8, p. 282, figure 3])

the performance of RCC8, there are algorithms with "[..] polynomial complexity in the number of instances"[8, p. 295]. The eight relations *DC (disconnected), EC (externally connected), PO (partially overlapped), TPP (tangential proper part), TPPi (tangential proper part inverse), NTPP (nontangential proper part), NTPPi (nontangential proper part inverse)* , and *EQ (equal)* which are defined by RCC8 are illustrated in figure 7.4. Spatio-temporal reasoning can be achieved by combining both, the spatial and the temporal logic calculi. The benefit of such a formalism is the ability "[..] to describe spatial configurations that change over time."[10, p. 312] Such formalisms can be used in EO application domains for crisis information when the extent of a flood is monitored over several days or a moving ship is tracked. The main idea of the spatio-temporal formalism

I: (X {DC,EC} Y)	I: (Y {TPP} Z)
J: (X {PO} Y)	J: (Y {DC} Z)

Table 7.2: Temporalizing RCC-8 using Allen's Algebra (source: [10, p. 313]

is to "temporalize RCC-8 using Allen's Algebra"[10, p. 313]. During certain intervals, different regional connections exist between regions. In the example in table 7.2, there are two intervals I and J. During Interval I, the region X is either disconnected or externally connected to region Y and region Y is a tangential proper part of Z. During interval J, the region X is partially overlapping region Y and region Y is disconnected from region Z. The complexity of the combined calculus is NP-complete.[10, p. 316] Furthermore, the usage and performance of spatial, temporal, and spatio-temporal reasoning is dependent on existing reasoner implementations.

7.4.3 Integrating further information resources

Besides the existing links to `data:Collection`s in the EOWEB system, the EOLO ontology could be extended to link the `ApplicationDomain`s and `Phenomena` to further information resources. Such approach could create a "[..] linked information space in which data is being enriched and added."[20, p. 100] The resource types can range from management and information services to the concrete GDAS primary data access (cf. [14]). The modelling of the new resources can be realized similarly to the current approach with `data:Collection`s and `measuresRelation`s by creating new OWL classes and individuals. Moreover, the axioms for the `Phenomenon` should be adjusted in order to infer membership of the additional resources to a certain `Phenomenon` and thus to `ApplicationDomain`s.

7.4.4 Semantic Modelling of EO processing chains

The EOLO ontology models the processing level of a `data:Collection` with the property `hasProcessingLevel` and its range class `data:ProcessingLevel`. Data processors geo-correct, scale, intersect or union data from one or several `data:Collections`. The resulting value-added data is again organized in a `data:Collection` with a higher processing level. The higher the processing level of a `data:Collection`, the more valuable and specific the data will be. Thus, the formalized connections between higher-level and lower-level `data:Collection`s could assist application domain experts and data ingestion operators in their daily work. Moreover, such modelling could reveal interesting connections between the processing chains of data processors and the `ApplicationDomain`s, as some application domain experts order data that is explicitly processed by the data ingestion operators.

Chapter 8

Conclusion

This diploma thesis has faced the problem of bridging the semantic gap in the EO domain that exists between its subdomains EO datacenter and EO application. EO application users do not have information about the relevance of data maintained by EO datacenter operators. Conversely, it is not possible for EO datacenter operators to name and structure EO data so that suits all requesting EO applications. Both EO subdomains are developing in different velocities and provide their own vocabulary, taxonomies, and relations. Consequently, a semantic gap emerges between the subdomains when information from the other subdomain is requested. Nethertheless, a high-level information model has been created out of the definition of two use cases for EO application domains, namely the WISDOM project for water management in the Vietnamese Mekong Delta and the WDC-RSAT measuring chemical compounds in the atmosphere. The employment and refinement of the authoritative SWEET ontologies for earth sciences have provided a neat interface for both EO subdomains. The high-level information model and the SWEET interfaces have influenced the formalization of the Earth Observation Lightweight Ontology (EOLO) covering both, the EO datacenter domain and the EO application domain. Most importantly, the semantic links between both EO subdomains have

been modelled into the EOLO ontology as logical axioms based on Description Logics (DL) instead of static connections. Therefore, reasoning can be employed to reveal the power of semantic classifications that bridge the semantic gap in the EO domain. EOLOsearch, a web demonstrator prototype has been implemented and deployed at the DLR to demonstrate the applicability of the developed EOLO ontology. The feasibility of the semantic modelling approach has been evaluated with several quality criteria such as *scaling performance, maintenance effort* and *search effectivity*. The evaluation results have pointed out that the usage of semantic modelling technologies and reasoning is a feasible task for the EO domain and only minor trade-offs have to be faced. Prospective projects can easily reuse and collaborate on the developed EOLO ontology as it has been formally specified in the Web Ontology Language (OWL) and an interoperable XML interface to retrieve `data:Collection`s is provided within the EOLOsearch web demonstrator. The formalization of the EO knowledge into an ontology provides a semantic communication model between various EO users, EO knowledge engineers and machine-based EO applications. By reusing and maintaining the EOLO ontology, new EO applications and services can emerge benefitting from an integrated formally specified model for the EO domain.

Acknowledgements

I would like to thank my supervisor at the University of Passau, Prof. Dr. Burkhard Freitag and his research assistants Dr. Franz Weitl, Christian Schönberg, and Alfons Ruch for their theoretical guidance and support during the stages of the diploma thesis.

I would also like to thank my supervisor at the DLR, Stephan Kiemle who continuously provided me with many interesting examples regarding the EO domain and pointed out modelling issues. The various review and discussion sessions with him improved the quality of the achieved work, significantly. Moreover, I would like to thank Bernhard Buckl for his guides considering the reuse of existing concepts from earth sciences and the DIMS team, especially Daniele Dietrich for the introduction into the EOWEB system and her assistance during the deployment of the developed EOLOsearch web demonstrator at the DLR.

Appendix A

Property files

```
1 # This properties file sets the database credentials for
2 # the informix db server
3 DS_HOST = giraffe
4 DS_PORT = 1525
5 DS_DBNAME = pro_eoweb
6 DS_USER = dimstest
7 DS_PASSWORD = ****
8 DS_IFXSERVER = dims
```

Listing A.1: `informix_db.properties` in package `model.db`

```
1 # This properties file sets the database credentials for
2 # the mysql db server
3 DS_HOST = giraffe
4 DS_PORT = 3306
5 DS_DBNAME = ontologydb
6 DS_USER = dimstest
7 DS_PASSWORD = ****
```

Listing A.2: `mysql_db.properties` in package `model.db`

```
1 # This properties file defines the attribute names from
2 # the 'm_productcollectio' entity type
```

```
3  TABLENAME_ENTITY_COLLECTIONDB = m_productcollectio
4  COLUMNNAME_UNIQUE_ID = unique_id
5  COLUMNNAME_ITEMTYPENAME = itemtypename
6  COLUMNNAME_LONGNAME = longname
7  COLUMNNAME_STARTTIME = starttime
8  COLUMNNAME_ENDTIME = stoptime
9  COLUMNNAME_INSERTIONTIME = insertiontime3
10 COLUMNNAME_COLLECTIONTYPENAME = collectiontypename
11 COLUMNNAME_QUALITY = quality
12 COLUMNNAME_PROCESSINGLEVEL = processinglevel
13 COLUMNNAME_RESOLUTION = resolution
14 COLUMNNAME_EOWEBURL = eoweb_url
15 COLUMNNAME_GDASURL = gdas_url
```

Listing A.3: `schema_entity_collectionDB.properties` in package `model.db`

```
1 # This properties file defines the attribute names from
2 # the 'geo_extents' entity type
3 TABLENAME_ENTITY_GEOEXTENTS = geo_extents
4 COLUMNNAME_UNIQUE_ID_GEO_EXTENT = unique_id
5 COLUMNNAME_LABEL_GEO_EXTENT = geo_label
6 COLUMNNAME_AREA_GEO_EXTENT = area
```

Listing A.4: `schema_entity_geoextents.properties` in package `model.db`

```
1 # This properties file defines the attribute names from
2 # the 'temporal_extents' entity type
3 TABLENAME_ENTITY_TEMPORALEXTENTS = temporal_extents
4 COLUMNNAME_UNIQUE_ID_TEMP_EXTENT = unique_id
5 COLUMNNAME_LABEL_TEMP_EXTENT = temporal_label
6 COLUMNNAME_DURATION_TEMP_EXTENT = durationInHours
```

Listing A.5: `schema_entity_temporalextents.properties` in package `model.db`

```
# This properties file defines the attribute names from
# the 'collection_geo_extents' relation type
TABLENAME_RELATION_GEOEXTENTS = collection_geo_extents
COLUMNNAME_FK_GEO_EXTENT = geo_extent
COLUMNNAME_FK_COLL_GEO_EXTENT = unique_id
```

Listing A.6: schema_relation_geoextents.properties in package model.db

```
# This properties file defines the attribute names from
# the 'collection_temporal_extents' relation type
TABLENAME_RELATION_TEMPORALEXTENTS =
                        collection_temporal_extents
COLUMNNAME_FK_TEMPORAL_EXTENT = temporal_extent
COLUMNNAME_FK_COLL_TEMP_EXTENT = unique_id
```

Listing A.7: schema_relation_temporalextents.properties in package model.db

```
## This properties file defines global properties for
## the EOLO Search web application.


## ONTOLOGY-RELATED CONFIGURATION

# The namespace of the EOLO ontology as specified as
# 'default namespace' in the EOLO ontology itself
NAMESPACE_EO = http://www.dlr.de/ontologies/EOLO.owl#

# The path to the Protege project file that stores the
# settings to load the EOLO ontology, relatively to the
# location of the ModelTransformer class (currently in
# package model.db)
PATH_TO_PROTEGE_PROJECT_FILE = EO_port.pprj
```

```
16
17
18
19 ## REASONER-RELATED CONFIGURATION
20
21 # Support for the reasoner Pellet
22 REASONER_PELLET = Pellet
23
24 # Support for the reasoner Racer
25 REASONER_RACER = Racer
26
27 # URL settings for Racer Reasoner
28 REASONER_RACER_URL = http://localhost:8081
29
30 # Support for a DIG-compliant reasoner
31 REASONER_DIG = DIG
32
33 # URL settings for the DIG-compliant reasoner
34 REASONER_DIG_URL = http://localhost:81
35
36 # The currently used reasoner, one of the supported
37 # Reasoners
38 USED_REASONER = Pellet
39
40
41
42 ## DATABASE-RELATED CONFIGURATION ##
43
44 # Support for the mysql database
45 DB_MYSQL = MYSQL
46
47 # Support for the informix database
48 DB_INFORMIX = INFORMIX
```

```
49
50 # The configured database type for the collectionDB,
51 # (either MYSQL or INFORMIX)
52 COLLECTION_DB = INFORMIX
53
54 # The configured database type for the ontologyDB,
55 # (either MYSQL or INFORMIX)
56 ONTOLOGY_DB = MYSQL
```

Listing A.8: `global.properties` in package `model`

Appendix B

HTTP GET parameters for Actions

Class name	Parameter name	Parameter value	Parameter description
EOLOServlet	action	export2OWL	Dispatches to the Export2OWLAction
		filterCollections	Dispatches to the FilterCollectionsAction
		retrieveSelectionTree	Dispatches to the RetrieveSelectionTreeAction
		searchCollections	Dispatches to the SearchCollectionsAction
		syncOntology	Dispatches to the SyncOntologyAction
		testPerformance	Dispatches to the PerformanceTestAction
Export2OWLAction	---	---	---
FilterCollectionsAction	filter_geo_extent	true	GeoExtents filter is activated
	geoExtentPercentage	a numeric value	This value represents the required percentual inclusion of the selected geo extent with the geo extents contained in all collections
	geoExtentType	defineGeoExtent	GeoExtents filter mode „define GeoExtent“
	geoExtent	{Global, Continental, Administrative, Regional, Metropolitan, Maritime}	The user selects one of the available parameter values of the geoExtent
		defineArea	GeoExtents filter mode „define area“
	geoExtentArea	a numeric value	The user selects inputs a numeric value representing the area
		defineGeoPolygon	GeoExtents filter mode „define GeoPolygon“
	nMostLat	a numeric value such that nMostLat > sMostLat	The north most Latitude of the GeoPolygon

Figure B.1: HTTP GET parameters for *actions* in package `controller.action`

wMostLon	a numeric value such that wMostLat < eMostLat	The west most Longitude of the GeoPolygon
sMostLat	a numeric value such that sMostLat < nMostLat	The south most Lattitude of the GeoPolygon
eMostLon	a numeric value such that eMostLon > wMostLon	The east most Longitude of the GeoPolygon
filter_temporal_extent	true	TemporalExtents filter is activated
temporalExtentPercentage	a numeric value	This value represents the required percentual inclusion of the selected temporal extent with the temporal extents contained in all collections. This parameter is only sent if
temporalExtent	{Annual, Monthly, Weekly, Daily, Hourly, Instant}	The user selects one of the available parameter values of the temporalExtent
filter_resolution	true	Resolution filter is activated
resolution	{HighResolution, MediumResolution, LowResolution}	The user selects one of the available parameter values of the resolution
filter_quality	true	Quality filter is activated
quality	{HighQuality, MediumQuality, LowQuality}	The user selects one of the available parameter values of the quality
filter_processing_level	true	Processing level filter is activated

Figure B.2: HTTP GET parameters for *actions* in package `controller.action`

processingLevel	{L0, L1, L1B, L2, L3, L4}	The user selects one of the available parameter values of the processingLevel
filter_historic_or_latest	true	Processing level filter is activated
historicOrLatest	{historic, latest}	The user selects one of the available parameter values of the historicOrLatest
filter_op	{0, 1}	The type of filter operation: 0 = AND, 1 = OR
PerformanceTestAction		
maxCollections	an integer value	This specifies how many random collections should be generated at most
RetrieveSelectionTreeAction		
treeType	{appTree, phenomenonTree}	Either the application domain tree or the phenomenon tree can be requested
SearchCollectionsAction		
concepts	One or several ApplicationDomains or Phenomena from the EOLO ontology separated with semicolons (;)	The selection of application domains or phenomena for reasoning. The URIs of the concepts are sent as parameters, but the „#“ must be replaced with a „.“, because of the used application server.
SyncOntologyAction		
---	---	---

Figure B.3: HTTP GET parameters for *actions* in package `controller.action`

Appendix C

Development tools and 3rd party libraries

Name	**version**
Protégé Ontology Editor	3.4 Beta
Eclipse Galileo IDE	3.5
CMAP Tools knowledge modeling kit	4.11.01
Jambalaya Ontology Visualization plugin	2.7.0
SQirrel SQL client	3.0.1
Dia	0.96.1
gnuplot	4.2

Table C.1: Used development tools

Name
antlr-2.7.5.jar
arq.jar
arq-extra.jar
aterm-java-1.6.jar

axis.jar
commons-discovery-0.2.jar
commons-lang-2.0.jar
commons-lang-2.2.jar
commons-logging.jar
commons-logging-1.1.1.jar
concurrent.jar
edtftpj-1.5.2.jar
ekitspell.jar
geodetic.jar
icu4j_3_4.jar
ifxjdbc.jar
ifxjdbcx.jar
iri.jar
jaxrpc.jar
jcalendar.jar
jdom.jar
jena.jar
jep-2.4.0.jar
json.jar
kazuki.jar
log4j-1.2.12.jar
lucene-core-2.3.1.jar
mysql-connector-java-5.1.7-bin.jar
orphanNodesAlg.jar
owlapi-api.jar
owlapi-apibinding.jar
owlapi-change.jar

owlapi-debugging.jar
owlapi-dig1_1.jar
owlapi-functionalparser.jar
owlapi-functionalrenderer.jar
owlapi-impl.jar
owlapi-krssparser.jar
owlapi-metrics.jar
owlapi-oboparser.jar
owlapi-owlxmlparser.jar
owlapi-owlxmlrenderer.jar
owlapi-rdfapi.jar
owlapi-rdfxmlparser.jar
owlapi-rdfxmlrenderer.jar
owlapi-util.jar
owlsyntax.jar
pellet.jar
protege.jar
protege-owl.jar
protege-pellet.jar
relaxngDatatype.jar
saaj.jar
stax-api-1.0.jar
swrl-jess-bridge.jar
wsdl4j.jar
wstx-asl-3.0.0.jar
xsdlib.jar

Table C.2: Imported 3rd party libraries in `WEB-INF/lib`

List of Figures

Bibliography

[1] Daniel J. Abadi, Adam Marcus 0002, Samuel Madden, and Kate Hollenbach. Sw-store: a vertically partitioned dbms for semantic web data management. *VLDB J.*, 18(2):385–406, 2009.

[2] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist. Modelling in RDF, RDFS and OWL.* Morgan Kaufmann Publishers, Burlington, MA, 2008.

[3] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[4] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer.* The MIT Press, Cambridge, Massachusetts, 2nd edition, 2008.

[5] Franz Baader and Werner Nutt. Basic description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logics Handbook: Theory, Implementations, and Applications*, chapter 2, pages 43–95. Cambridge University Press, Cambridge, 2nd edition, 2007.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[7] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: An architecture for storing and querying rdf data and schema

information. In Henry Lieberman Dieter Fensel, James Hendler and Wofgang Wahlster, editors, *Spinning the Semantic Web. Bringing the World Wide Web to Its Full Potential*, chapter 7, pages 197–222. The MIT Press, Cambridge, Massachusetts, 2005.

[8] Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1(3):275–316, 1997.

[9] John Davis, Dieter Frensel, and Frank van Harmelen. *Towards the Semantic Web. Ontology-driven Knowledge Management.* John Wiley and Sons, Chichester, West Sussex, 2003.

[10] Alfonso Gerevini and Bernhard Nebel. Qualitative spatio-temporal reasoning with rcc-8 and allen's interval calculus: Computational complexity. In *ECAI*, pages 312–316, 2002.

[11] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Knowledge Systems Laboratory, Technical Report, Compurter Science Department, Stanford University, California*, KSL 92-04:907–928, 1993.

[12] M. Gruninger and M. Fox. The role of competency questions in enterprise engineering. In *Workshop on Benchmarking- Theory and Practice.*, number WG5.7, Trondheim, Norway, 1994.

[13] John Hebeler, Matthew Fisher, Ryan Blace, and Andrew Perez-Lopez. *Semantic Web Programming.* Wiley Publishing, Inc., Indianapolis, 2009.

[14] T. Heinen, B. Buckl, T. Erbertseder, S . Kiemle, and D. Loyola. Standardized data access services for gome-2/metop atmospheric trace gas monitoring. In *EUMETSAT - Meteorological Satellite Conference*, Darmstadt, 2008.

[15] Ian Horrocks, Peter F. Patel-Schneider, Deborah L. McGuiness, and Christopher A. Welty. Owl: a description-logic-based ontology language for the semantic web. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logics Handbook: Theory, Implementations, and Applications*, chapter 14, pages 458–486. Cambridge University Press, Cambridge, 2nd edition, 2007.

[16] Stephan Kiemle. From digital archive to digital library - a middleware for earth-observation data management. In *Research and Advanced Technology for Digital Libraries*, volume 2457/2002, pages 61–73, Springer Berlin / Heidelberg, 2002.

[17] Stephan Kiemle and Burkhard Freitag. Providing context-sensitive access to the earth observation product library. In *Research and Advanced Technology for Digital Libraries*, volume 4675/2007, pages 223–234, Springer Berlin / Heidelberg, 2007.

[18] Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logics Handbook: Theory, Implementations, and Applications*, chapter 1, pages 1–39. Cambridge University Press, Cambridge, 2nd edition, 2007.

[19] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga. *Knowledge Engineering and Management. The CommonKADS Methodology.* The MIT Press, Cambridge, Massachusetts, 3rd edition, 2002.

[20] N. Shadbolt, Tim B. Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[21] C. M. Sperberg-McQueen, François Yergeau, Eve Maler, Jean Paoli, and Tim Bray. Extensible markup language (XML) 1.0 (fifth edition). W3C proposed edited recommendation, W3C, feb 2008. `http://www.w3.org/TR/2008/PER-xml-20080205`.

[22] York Sure and Rudi Studer. A methodology for ontology-based knowledge management. In John Davis, Dieter Frensel, and Frank van Harmelen, editors, *Towards the Semantic Web. Ontology-driven Knowledge Management*, chapter 3, pages 33–46. John Wiley and Sons, Chichester, West Sussex, 2003.