# Open Geospatial Consortium

Approval Date: 2012-09-07

Publication Date: 2012-10-10

**External identifier of this OGC® document:** http://www.opengis.net/doc/bp/semantic-annotations

Reference number of this document: OGC 08-167r2

Version: 2.0

Category: OGC® Best Practice

Editor: **Frédéric Houbie**, **Philippe Duchesne**, **Patrick Maué**

# Semantic annotations in OGC standards



**Warning**

This document defines an OGC Best Practices on a particular technology or approach related to an OGC standard. This document is not an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an official position of the OGC membership on this particular technology topic.

| | |
|---|---|
| Document type: | OGC® Best Practice |
| Document subtype: | NA |
| Document stage: | Approved for public release |
| Document language: | English |

# Outline

# Figures

# LISTINGS

## i. Abstract

In this OGC Best Practice, the concept of **semantic annotations** is introduced. Annotation of Web Services or data compliant to OGC standards refers to the task of attaching meaningful descriptions to the service and the served geospatial data or processes.

## ii. Keywords

ogcdoc, semantic, annotations, metadata

## iii. Preface

Annotation of Web Services or data compliant to OGC standards refers to the task of attaching meaningful descriptions to the service and the served geospatial data or processes. This Best Practice extends the expressiveness of such annotations by including more sophisticated (semantic) descriptions. Semantic annotations can be applied on different levels (metadata, data model, and instance data). This Best Practice investigates implications on typical applications on all three levels, and discusses how they can be technically realized.

The first section introduces semantic annotations, mentions the three levels, and elaborates on semantic problems, which are evident in contemporary spatial data infrastructures. Such semantic conflicts are resolved if the proposed semantic annotations are incorporated by query processing routines in information retrieval (IR) systems like OGC Catalogs. This BP also discusses typical applications where semantic annotations can facilitate the use of OGC Web Services.

The second section provides a detailed discussion of the three different levels and relates them to standardization efforts of ISO/TC211. Implications on the find-ability of geospatial resources and their evaluation due to the additional reasoner support are discussed for each level. Potential benefits as well as the drawbacks are included. The technical realization of the different approaches are discussed and compared as well.

In the next section we discuss the implications for existing GIS standards like GML and KML, which are mostly from the OGC community. But also other standards like ISO 19115 and 19119 are included. Examples how to realize the semantic annotations for the different standards are listed.

This Best Practice concludes with a comparison of each approach, and with some suggestions when to use which method.

## iv. Document terms and definitions

This OGC Best Practice paper includes the following abbreviated terms:

- OGC Open Geospatial Consortium
- OWS OGC Web Services

## v. Normative References

- *OGC® Standard : Web Service Commons (**WS-Common**)*
  *URL:* http://www.opengeospatial.org/standards/common
- *OGC® Standard : OGC® KML **(KML)***
  *URL:* http://www.opengeospatial.org/standards/kml
- *OGC® Standard : Geography Markup Language **(GML)***
  *URL:* http://www.opengeospatial.org/standards/gml
- *OGC® Standard : Sensor Model Language (**SensorML**)*
  *URL:* http://www.opengeospatial.org/standards/sensorml
- *OGC® Standard : Web Feature Service (**WFS**)*
  *URL:* http://www.opengeospatial.org/standards/wfs
- *OGC® Standard : Web Processing Service (**WPS**)*
  *URL:* http://www.opengeospatial.org/standards/wps
- *OGC® Standard : Web Coverage Service (**WCS**)*
  *URL:* http://www.opengeospatial.org/standards/wcs
- *OGC® Standard : Web Map Service (**WMS**)*
  *URL:* http://www.opengeospatial.org/standards/wms
- *OGC® Standard : Catalogue Service (**CAT**)*
  *URL:* http://www.opengeospatial.org/standards/cat
- *OGC® Standard : Filter Encoding (**FES**)*
  *URL:* http://www.opengeospatial.org/standards/filter
- *ISO/IEC 13211: **Prologue***
- *ISO/IEC 13250: **Topic Maps***
  *URL:* http://www1.y12.doe.gov/capabilities/sgml/sc34/document/0322.htm
- *ISO 19109: Geographic information **:** Rules for application schema*
- *ISO 19119:2005: Geographic information -- Services*
- *ISO 19115:2003: Geographic information -- Metadata*
- *ISO/TS 19139:2007: Geographic information - Metadata - XML schema implementation*
- *W3C® Recommendation: **OWL** Web Ontology Language*
  *URL:* http://www.w3.org/TR/owl-features/

- *W3C® Recommendation: Semantic Annotations for WSDL and XML Schema (**SAWSDL**), URL:* http://www.w3.org/TR/sawsdl/
- *W3C® Member Submission: **SWRL**: A Semantic Web Rule Language Combining OWL and RuleML, URL:* http://www.w3.org/Submission/SWRL/
- *W3C® Member Submission: Web Service Modeling Ontology (WSMO) URL:* http://www.w3.org/Submission/WSMO/
- *IETF® RFC: Uniform Resource Identifier (**URI**): Generic Syntax URL:* http://tools.ietf.org/html/rfc3986

## vi. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

| Name | Organization |
|---|---|
| Patrick Maué | IFGI, University of Münster, Germany |
| Sven Schade | IFGI, University of Münster, Germany |
| Philippe Duchesne | Erdas, Belgium |

## vii. Revision History

| Date | Release | Editor | Primary clauses modified | Description |
|---|---|---|---|---|
| 11/07/08 | 0.0.1 | Patrick Maué | | Initial version |
| 01/05/09 | 0.0.3 | Carl Reed | Various | For publication as DP |
| 07/15/09 | 1.0 | Patrick Maué, Sven Schade | Various | Formatting, Relation to ISO, Updates to Section 3, Clarifications |
| 06/10/11 | 2.0 | Philippe Duchesne | Various | Updates on integration with GIS standards |

## viii. Changes to the OGC Abstract Specification

*The OGC® Abstract Specification does not require changes to accommodate the technical contents of this document.*

## ix. Foreword

*Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.*

*Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.*

# Semantic annotations in OGC standards

## 1. Introducing annotation

The World Wide Web is a vast collection of arbitrary resources, held together by standards that tell us how to locate and transport data. Such resources might be textual information on a Web site but also can be images, a video, download-able multi spectral satellite image, OGC Web Feature Service providing some vector data, or a Sensor Observation Service delivering up-to-date temperature values. Without proper descriptions, the use of such a resource is limited to a small user group. Before publishing a resource in the Web, it has to be **annotated** with descriptive metadata to make it usable to a broad audience. Otherwise people will neither be able to find the resource using search engines nor will they be able to evaluate if the discovered resource satisfies their current information need.

The OGC standards baseline provides accepted and well thought-out methods to make spatial resources (data and processes) served via Web Services accessible. Service capabilities (see figure 1 as example) describe, besides contextual information like contact information, how to access and invoke the service to retrieve the required geospatial data. The individual name and location of the operations are also listed in the Capabilities document of each OGC-conformal Web Service (as defined in OGC WS-Common). Since such operations and the format to encode the data are predefined in *OGC Implementation Standards* and *OGC Encoding Standards*, generic clients can, without knowledge about the nature of the data, display the resulting data on a map.

In this OGC Best Practice, we introduce the notion of **semantic annotation**. With the help of the OGC Capabilities-Document and the various individual descriptions of feature types, coverages, or processes, the standards of the OGC define how to access, invoke, and finally visualize spatial data. The OGC standards cover the functional dimension of a Web Services. But they lack a well-defined methodology to describe the thematic dimension of a Web Service. They don't tell much about what the served data (or process) represents, and in particular they lack a way to link the resources to external models. For example, the application knows how to load and visualize the data on a map, but the user has no idea how to read the displayed map. With the help of semantic annotations, data providers will be able to connect the standardized service descriptions to the modeled knowledge. Such models comprise conceptualized knowledge about the represented geographic phenomena. Having such a link established, reasoning algorithms will be able to infer if a Web Service matches an agent's query on the formal level. In addition it will allow for extracting valuable contextual information from the knowledge models, making it possible to display thematic information for the displayed data and helping the user to understand.

The in depth discussion of applications for semantic annotations in the context of OGC Standards can be considered to be the main contribution of this Best Practice. The concept of three different levels where annotations can be applied has been developed for this Best Practice paper. Additionally, this is the first time we discuss the semantics of the reference, and justify the

distinction between model references and domain references. This Best Practice paper does not suggest introducing new standards. We rather propose to extend the notion of already existing OGC standards and to apply standards from other organizations like the W3C to OGC Standards.

In the remainder of this section we are going to discuss which different levels for annotating OWS descriptions exist, why *Semantics* are needed in the annotation process, and what types of applications and scenarios can benefit from including knowledge model references in the annotation process. We use semantic discovery as main motivation, but include pointers to additional usage scenarios.

Figure 1 shows the metadata available for a WFS serving one feature type with the name `exploitationsponctualsproduction`. The example will be used throughout this Best Practice paper to illustrate why semantic annotations are needed, and how they should be realized. This example, and accordingly the service providing the associated data [1], has been developed within the European funded project SWING[2]. The methods have been also studied during research for the German funded project GDI Grid[3].

```
<WFS_Capabilities version="1.0.0">
    <Service>
      <Name>MapServer WFS</Name>
      <Title>Quarries</Title>
      <Abstract>Quarries from SchemaCa</Abstract>
      <Keywords> brgm, quarry,quarries, france, schemaca</Keywords>
      <OnlineResource>
        http://swing.brgm.fr/cgi-bin/carrieres?
      </OnlineResource>
    </Service>

    <Capability> ... </Capability>

    <FeatureTypeList>
        <FeatureType>
            <Name>exploitationsponctualsproduction</Name>
            <Title>Quarries With Production</Title>
            <SRS>EPSG:4326</SRS>
            <LatLongBoundingBox minx="-6.06258" miny="41.1632"
                    maxx="10.8783" maxy="51.2918" />
        </FeatureType>
    </FeatureTypeList>
</WFS_Capabilities>
```

**Figure 1 - Excerpt from an OGC WFS Capabilities document**

[1] Link to this particular WFS: http://swing.brgm.fr/dataaccess/wmswfs.

[2] Link to the official Web site: http://www.swing-project.org/.

[3] Link to the official Web site: http://www.gdi-grid.de

```
<schema targetNamespace="http://swing.brgm.fr/quarries" ... >

 <element name="exploitationsponctualsproduction"
          type="qua:exploitationsponctualsproductionType"
          substitutionGroup="gml:_Feature" />

 <complexType name="exploitationsponctualsproductionType">
   <complexContent>
     <extension base="gml:AbstractFeatureType">
       <sequence>
         <element name="msGeometry" type="gml:GeometryPropertyType">
         <element name="name" type="string"/>
         <element name="year" type="string"/>
         <element name="allowedproduction" type="string"/>
       </sequence>
     </extension>
   </complexContent>
 </complexType>

</schema>
```

**Figure 2 - The data model for the WFS of Figure 1.**

Figure 2 lists the GML schema for the `exploitationsponctualsproduction` feature type. But what does its data represent? The name is already confusing; the attributes don't give us any additional information. Whose name is the value of `name`, what does the attribute `year` refer to, and how is the `allowedproduction` measured?

## 1.1 Why semantic annotations

Before going into the technical details about semantic annotations, we need to discuss some theoretical background of semantics, in particular how semantics can increase the usefulness of geospatial information. We introduce the different types of semantic conflicts and the notion of semantic interoperability, the idea of conceptualization to resolve these conflicts, and the use of formal languages to specify such conceptualizations.

The major goals of standards like XML or GML is to achieve syntactic and structural interoperability between different software components[4]. If two agents agree on how to represent

[4] How the OGC defines interoperability: http://www.opengeospatial.org/ogc/faq/openness (#10)

and communicate the data, a seamless and conflict-free integration is established. Hence, semantic interoperability can be achieved if the two agents (both, humans and machines) agree on how to understand the data. The level of understanding can differ and depends mostly on the complexity of the formalized knowledge. The following listing discusses common conflicts (or heterogeneities) which can be addressed using conceptualizations on the domain level and the semantic annotations to relate OWS to these domain concepts.

**Application-specific knowledge:** The feature type modeled in figure 2 shows an example for problems that are caused by application-specific knowledge. First, certain identifiers or terms are used in a small community, which makes the underlying data hardly usable for other users. The casual user does obviously not know what an `exploitationsponctualproduction` represents, and he would fail to discover the data even if would provide the required search terms. The attribute `allowedproduction` remains unclear to the user as well. It refers to the maximum allowed production of the product (gravel, chalk, …) in tons per year. The attribute `year` is in this case the year this allowed production is meant for. Without a further description which is shared to the other parties, the use of this data is constrained to a very limited user group. Having elaborated semantic descriptions linked to these attributes and feature types provides a solution to this problem.

**Hierarchical Problems:** Probably the most common issue in service discovery is the different level of expertise between the seeking user and the data provider. A casual user looking for excavations is obviously taking the term "excavation" or "mine". The specialist publishing the data is using more specific terms, like "quarry" (an open-pit mine) or "chalkpit" (a quarry use to excavate chalk). Searching based only on keywords would yield no results here. Since one term is more specific then another (we refer to it as a hypernym [5]), they can be simply associated on the conceptual level using taxonomic relationships. The domain concept QUARRY is then simply modeled as sub-concept of the domain concept EXCAVATION.

**Multilingual community:** Yet often neglected in the Anglo-centric GIS community, the different languages spoken by users impair the find-ability of spatial resources as well. Especially in the European region the requirement for multilingual descriptions of geospatial data gains importance. Concepts at the domain level support labels in different languages, which makes it possible to match a user's query and service description even if the chosen languages differ. This approach assumes that terms in different languages can be referred to exactly the same concept. But the German term "Steinbruch" does not necessarily bear the same meaning as the American English term "Quarry" or the British English term "stone pit". Depending on the required complexity of the knowledge, language-specific concepts might need to be introduced as well.

Typical semantic problems result from the ambiguity of human language. Individual language depends greatly on the background knowledge, which is influenced by society, culture,

[5] From the WordNet Glossary (http://wordnet.princeton.edu/man/wngloss.7WN) : Hypernym is "The generic term used to designate a whole class of specific instances. **Y** is a hypernym of **X** if **X** is a (kind of) **Y**."

profession, and much more. If data should not be restricted to a user community speaking, by chance, the same language as the data provider, we need to solve the following semantic heterogeneities:

**Synonymy**: One term is a synonym of another different word, if they both denote the same object. A synonym for the word "quarry" is, for example, the "open-pit mine". On the conceptual level, we would identify the concepts QUARRY and OPENPITMINE, as similar. A reasoning algorithm is then able to infer that a user searching for "quarry" is, in the same moment, also looking for a "open pit mine". The user doesn't even have to realize that the query has be redefined, since all additional results are as relevant as the others.

**Homonymy**: Typical representatives of semantic heterogeneities are also homonyms. One word has multiple meanings, and only the context makes it understandable to the reader. "Mine", for example, is defined either as excavation in the earth[6] or as explosive device that explodes on contact. It can in both cases also be used as verb. On the conceptual level such terms are only used to label a concept. It is therefore quite simple to distinguish between homonyms. The words look equal, but they are linked to different concepts.

**Polysemy:** Typical semantic problems in geospatial applications are result of the ambiguity of place names (also called toponyms). Like homonyms, a polysem is described as a word which can be understood differently depending on contextual information. But in this case the denoted objects (and therefore also the concepts on the domain level) are similar. A toponym is very often also a polysem. Examples are "downtown", "Main Street", or even names of cities like "Springfield"[7]. Since something like "downtown" stays conceptually the same, even if it there is a large spatial distance in space, it cannot be fully covered by formal domain concepts on the domain level. Annotations of the individual data entities, on the other hand, might help.

## 1.2 Different perspectives on annotations

Semantic annotations establish a connection between the geospatial resource, its metadata, and the ontology. The following figure 3 illustrates the three different levels of (semantic) annotations which are possible for OGC Web Services, taking the WFS from Figure1 as example.

[6] We are using the wordnet definitions here.

[7] Wikipedia claims that there are 34 Springfields in the US

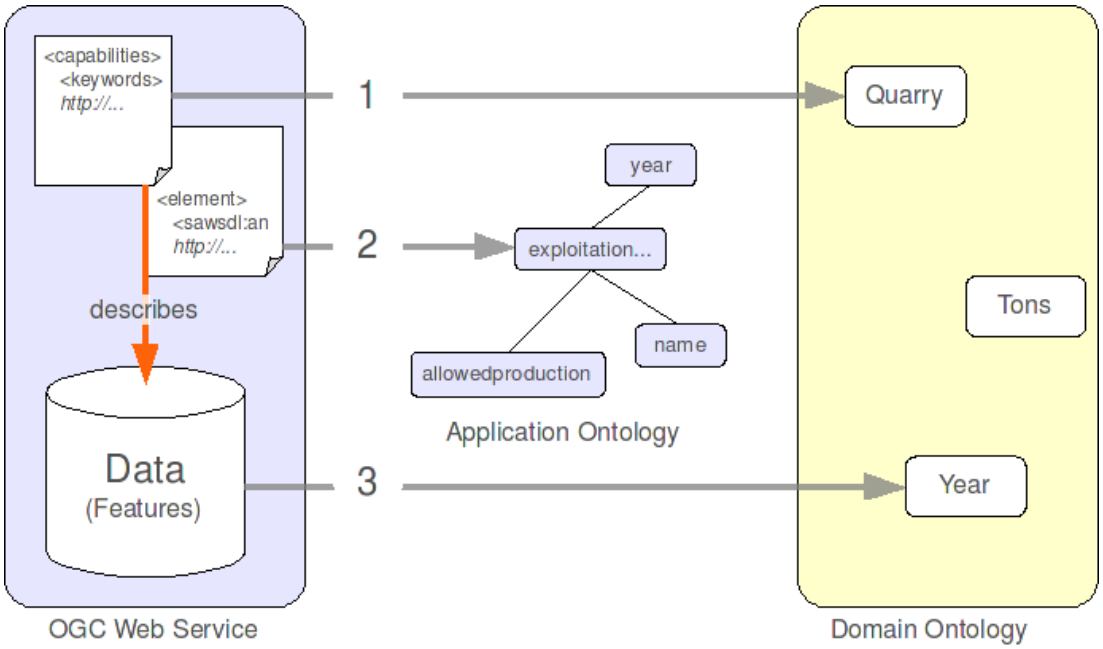


**Figure 3 - Semantic annotations at three different levels**

It is possible to distinguish between three locations where particular information about the resource can be acquired. The *Capabilities*-document defined in OWS-Common comprises functional properties telling the user how to access and invoke the service, as well as some **resource metadata** with information about the service provider, licensing, a title and description, or a keyword section (Figure 1). An additional document provided by every OWS is a XML-based schema representing the **data model** (figure 2), which comprises a description of the data model with focus on syntax and structure. Both documents, the metadata and the schema, are describing the underlying data, and therefore explicitly linked (highlighted by the orange arrow). The third source of information are obviously the **data entities** itself, encoded in the format predefined in the data model specified in the data schema.

A reference (the numbered arrows in the figure) to a knowledge model is feasible on all three levels. In figure 3, a WFS which serves quarry features (with `allowedproduction` as one attribute) is semantically annotated. We can directly link (1) a keyword within the Capabilities-Document to the corresponding concept in the domain ontology. We can annotate (2) the data model (the features and the attributes) using an application-specific ontology. Or we attach (3) semantic annotations directly to the feature instances. Each type of annotation has different implications on the find-ability of the Web Service, and the possibility of the user to evaluate if the served data satisfies his needs. The different levels with benefits and drawbacks are discussed in detail later on.

## 1.3 Semantics of the Reference

The link between elements in a XML document and concepts from a shared vocabulary encoded in another format is coined the *model reference* (introduced in the W3C Standards SAWSDL). Its

purpose is to bridge different languages, and is always a link between two models (e.g. XML Schema or UML modeling data, RDF modeling a domain vocabulary, and so on).

The *domain reference*, on the other hand, links between a local, application-specific model and a global, shared vocabulary. Whereas the model reference is a unique URI which points to the corresponding element in another model, the domain reference can be also expressed in form of complex rules. Here, n-ary links are required to accommodate for the complex relationship between application-specific features and common domain knowledge.

The following figure shows how these references relate. Note that the two types of references can overlap. In many cases a domain reference is in the same time also a model reference, since the reference performs both tasks (bridging in between two languages and linking from local to global). In this case, the domain reference should be used.

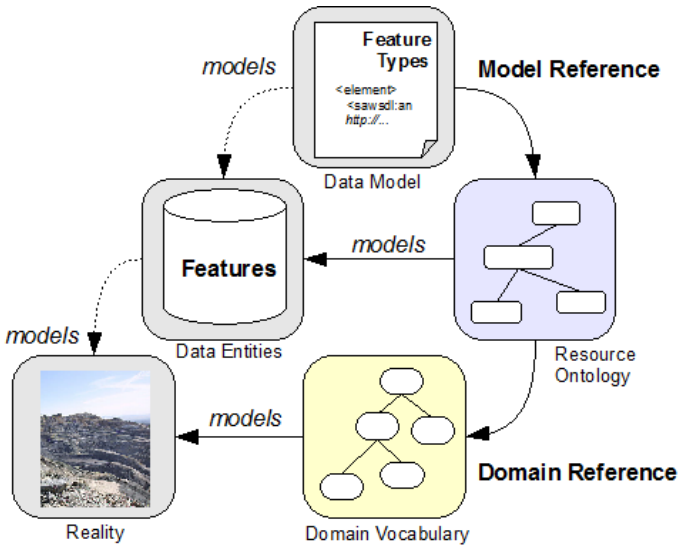


**Figure 4 - Two types of references**

The figure above represents a level 2 annotation: a data model (the feature type schema) is linked to the resource ontology via the model reference link, which again is linked to the domain ontology with the domain reference. If the features itself are to be annotated (e.g. placemarks in a KML file), we should use the domain reference if we link directly to a shared vocabulary, such as a gazetteer.

The distinction between two reference types (and the introduction of the resource ontology) adds an additional layer of complexity, but has also several benefits which in our opinion are crucial for the applications introduced later. Using only domain references (linking directly to domain vocabularies) for level 1 and level 2 annotations results in either too application-specific domain models or too generic annotations. Take for example the attribute `name` from the feature type example in Figure 2. Either it is linked to the concept IDENTIFIER, which means we lose the information what entity is identified with its value. Or we link it to the concept QUARRYNAME, which means we have to update the domain vocabulary for data models with complex relationships; the latter results in cluttered vocabularies barely usable for tasks like semantic

integration. With a resource ontology, we can model the implicit relationships between the different data feature attributes and the feature itself. On a local level we model that the name is the name of the quarry, or that the value of the year refers to the year where the value of the allowed production rate is valid.

## 1.4 Ontologies as Formalized Knowledge

Up to now we always referred to conceptualizations on the domain level, and avoided the term ontology. Analogue to Gruber's definition (Gruber, 1995) for ontologies, we see an ontology as particular specification of a conceptualization. They are not necessarily needed to solve every potential semantic problem listed in Section. Different levels of hierarchies are, for example, also resolved using thesauri like GEMET[8] from the European Environmental Agency (EEA) or AGROVOC from FAO[9], which serve synonyms and hypernyms for a defined collection of terms. But the proposed solution should cover all introduced challenges resulting from semantic heterogeneities. Dictionaries, controlled vocabularies and even taxonomies can be considered as less formal, less expressive, and therefore less powerful conceptualizations of domain knowledge. Ontologies and the underlying formal systems have the required expressiveness to formalize the conceptualizations and rules required to avoid the listed semantic conflicts. Note though, that the mentioned domain-specific thesauri represent the vocabulary commonly accepted within the information community. Engineering domain ontologies representing a certain domain's vocabulary should therefore always take common and well-accepted thesauri as starting point. Or, even better, the authorities controlling the thesauri also create the ontologies (in the case of AGROVOC).

The Ontology Definition Metamodel (ODM)[10], which is built on top of OMG's Model Driven Architecture (MDA), identifies four main components used to specify (and formalize) a conceptualization: relations, individuals, axioms, and the concepts used to define the vocabulary. This is a rather light-weight view on ontologies, but nonetheless allows for explaining the main difference to a less flexible knowledge base like a thesaurus. The ontology's axioms can be used to constrain the relationship between concepts, or to create new knowledge in the form of rules. Having these complex concept definitions (concept, its relations, the axioms constraining the relations, and the individuals of this concept) expressed in a language based on a formal deductive system like first-order logic or description logic enables the use of reasoning algorithms. Inference can be as simple as creating new facts from transitive relations, but also as complex as matching complex logical statements using techniques like Query Containment (Calvanese, 1998). It depends on the required expressiveness which kind of logic language is selected. But the more expressive the underlying language, the less decidable are the expressed statements. A reasoning algorithm which has to compute a potential match between services described using ontologies based on first-order logic requires more resources (in terms of time

[8] Example of EEA: http://glossary.eea.europa.eu/terminology/concept_html?term=quarry

[9] Find more information her: http://www.fao.org/**agrovoc**/

[10] Ontology Subgroup of OMG: http://ontology.omg.org/

and computing power) compared to ontologies used in a less expressive description logic (Baader, 2003).

In this Best Practice we discuss semantic annotations of existing OGC standards. We try to focus on the extensions in the existing service description documents, but nonetheless we have to introduce the standards used to express the conceptualization on the domain level. A wide variety of different languages and supporting tools is available, ranging from the very simple Topic Maps (ISO/IEC 13250) to complex descriptions in logic programming languages like Prolog (ISO/IEC 13211). We focus on three examples which are commonly used today and which have gained tremendous support in the rise of the Semantic Web.

The most common language is the RDF-based **Web Ontology Language (OWL)**, which has reached the status of a W3C Recommendation. The most frequently OWL variant (OWL-DL) is based on Description Logic (DL), which constrains its expressiveness and therefore applicability. But the decidability, good support, and wide availability of tools makes it the first choice for the majority of applications. OWL2[11] introduced the possibility to create profiles, with different supported reasoning capabilities in mind. OWL can be bundled with even more expressive rule languages like the Semantic Web Rule Language (SWRL). Rules can lead to undecidable statements in the ontologies, but still contribute significantly to the expressiveness of OWL.

The **Web Service Modeling Ontology (WSMO)** was developed to account for the requirements of finding and composing Web Services. The used language, the Web Service Modeling Language (WSML), comes in different flavours. Like OWL-DL, WSML-DL is constrained to the language constructs allowed for a Description Logic. But with WSML-Flight and WSML-Rule, two flexible languages are available which allow for expressing even more sophisticated conceptualization. A thorough introduction in WSML is available in the W3C submission.

The **Simple Knowledge Organization System (SKOS)** is a formal language designed for representation of thesauri, classification schemes, taxonomies or any other type of structured controlled vocabulary. SKOS is built upon RDF and RDFS, is simpler than OWL, and focuses on the easy publication of controlled structured vocabularies. Because it is based on RDF, SKOS entities can be integrated in OWL constructs. It must be noted that SKOS concepts are not OWL classes (they are instances thereof) and therefore cannot be used as instantiable classes, nor have class properties. SKOS has reached the status of W3C Recommendation.

The following two figures 4 and 5 show how the application ontology from figure 3 on page 12 would be encoded in OWL and WSML. (The SKOS example is detailed later) Sentences in OWL are represented as triples, because it has been modeled as extension to RDF and RDF(S). The WSML code has a RDF representation as well, but can also be serialized (and later again parsed) using an human-readable form like in figure 5. In the WSML example cardinalities have been added to the attributes. In this case the attribute `msGeometry` is mandatory, because it has a cardinality of exactly 1. All other attributes can be either present or not.

[11] More about profiling for OWL: http://www.w3.org/TR/owl2-profiles/

```
<owl:Class rdf:ID="exploitationsponctualsproduction"/>
<owl:DatatypeProperty rdf:ID="allowedproduction">
    <rdfs:domain rdf:resource="#exploitationsponctualsproduction"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#exploitationsponctualsproduction"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="year">
    <rdfs:domain rdf:resource="#exploitationsponctualsproduction"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

**Figure 5 - Application Ontology encoded in OWL**

```
concept exploitationsponctualsproduction subConceptOf gml#Feature
    msGeometry impliesType (1) gml#GeometryPropertyType
    Year impliesType (0 1) _string
    AllowedProduction impliesType (0 1) _string
    Name impliesType (0 1) _string
```

**Figure 6 - Application Ontology encoded in WSML**

Because SKOS concepts are not classes, as stated above, they cannot hold class-level properties as in the examples above. SKOS concepts are meant to hold static entities that form a controlled vocabulary. Below is a possible SKOS encoding of the OTEG thesaurus :

```
<rdf:RDF
    xmlns="http://oteg.esa.int/infodomain-v1.0.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:skos='http://www.w3.org/2004/02/skos/core#'>
  <skos:ConceptScheme rdf:about="http://oteg.esa.int/infodomain-
v1.0.owl"/>
  <skos:Concept rdf:about="#Land_monitoring">
    <skos:broader rdf:resource="#Information_Domain"/>
  </skos:Concept>
  <skos:Concept rdf:about="#Agriculture">
    <skos:broader rdf:resource="#Land_monitoring"/>
  </skos:Concept>
  <skos:Concept rdf:about="#Cloud">
    <skos:broader rdf:resource="#Agriculture"/>
    <skos:broader rdf:resource="#Meteorological_data"/>
  </skos:Concept>
  <skos:Concept rdf:about="#Meteorological_data">
    <skos:broader rdf:resource="#Wave_climate"/>
    <skos:broader rdf:resource="#Operational_oceanography"/>
    <skos:broader rdf:resource="#Weather"/>
  </skos:Concept>
    [...]
</rdf:RDF>
```

**Figure 7 - OTEG Thesaurus encoded in SKOS**

An ontology architecture provides an high-level view on the various types of ontologies, and how they are related to each other. There are two types of distinctions which will be used through-out this Best Practice.

**Local vs. global:** Local ontologies are used in a particular application, and are not shared or publicly available. Global ontologies, on the other hand, are available to everyone, and should therefor not contain individual knowledge. Concepts in local ontologies have to be linked to global concepts to make them useful, for example, to let the reasoner decide if two local concepts are denoting the same thing.

**Application, Domain, and Foundational Ontologies:** Local ontologies as mentioned above are usually coined application ontologies, in the following we prefer to use **Resource Ontology**. Such ontology comprises concepts used for a specific application, in our context the application is the provision and use of geospatial resources in the WWW. A Resource Ontology might include, for example, concepts representing feature types and their attributes (see Figure 1) which are defined in the feature type schema of a Web Feature Service. **Domain** Ontologies are shared and global conceptualizations which are used across different applications. A **Foundational** Ontology

(also called Upper Level, Top Level, or Formal Ontology) sits on top of domain ontologies, and allows for bridging between the different domains. This level of ontologies is not yet fully specified, and requires more research before being useful for applications.

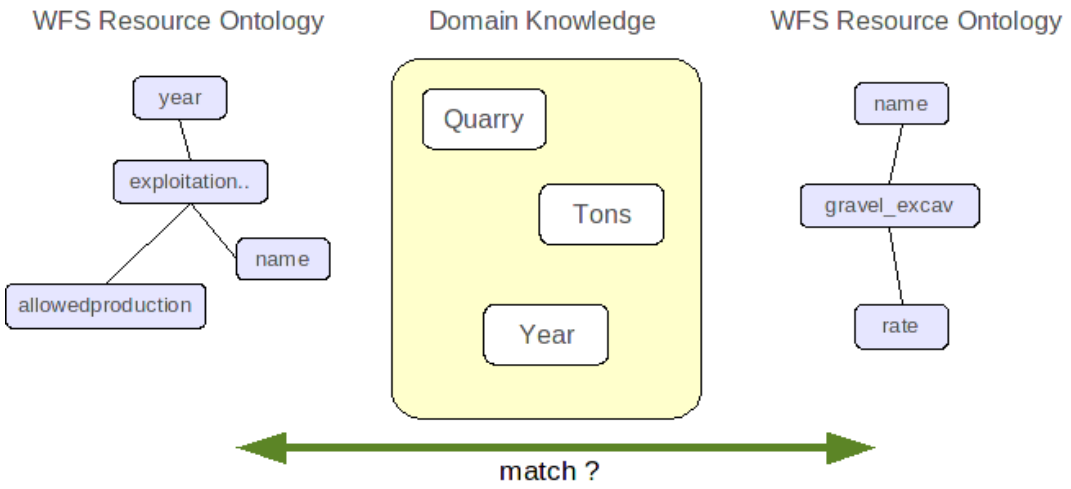


**Figure 8 - Ontology Architecture**

The strict distinction between domain ontologies which try to capture the common knowledge of one community, and the application-specific Resource Ontology which models the very specific inner workings of a particular dataset or process is fundamental for the understanding of semantic annotations. Information exchange requires commitment of all parties to the global knowledge modeled in the domain ontology. On the local level each participant can model the application ontologies in a way which best suits his purpose. Forcing him to use domain concepts only will result in loosing valuable application specific knowledge. Relating the local application ontologies to global ontologies allows for keeping this local knowledge as well making the served data findable for reasoner-supported IR systems. Such systems are now able to infer that services like the two illustrated in figure 6 do actually match because they serve related data and can be both of interest for a discovery task using, for example, the concept QUARRY.

A slightly more complex reasoning scenario involves subsumption reasoning. The local concept EXPLOITATION is related to the domain concept QUARRY, the other local concept GRAVEL_EXCAV is associated with the domain concept INDUSTRY. Since Quarry is modeled as being a sub-concept of INDUSTRY, the reasoning engine is able to infer the both local concepts are related (they are not similar, though).

## 1.5 Applications

We claim that semantic annotations can increase the usefulness of OGC Web Services in nearly many scenarios they can be deployed in. Four typical applications are discussed in detail below.

They are subject of interest for two research projects the authors are involved with. In the German GDI-Grid project we studied ways how to semantically validate GRID-enabled workflow. And in the European funded SWING project the applicability of Semantic Web Technologies in the context of geospatial decision making in the mining industry has been discussed.

### 1.5.1 Geographic Information discovery and retrieval

Due to the complex nature of Geographic Information, the discovery of such data is significantly more challenging then searching, for example, a Web site using a keyword-based search engine. Spatial data comprise several search-able dimensions: Space, Time, and Theme. OGC Standards like Filter Encoding have addressed the spatial dimension sufficiently, and even temporal characteristics of spatial data can be simply queried using a well-defined temporal reference system like the Calendar. The thematic dimension on the other hand is restricted to string-based filters, which neglects the semantic problems introduced in section 1.1 and makes the discovery of geographic resources representing a specific phenomena a tedious task.

We argue that GIR using semantic-enabled catalogs can improve user experience and usefulness of such IR systems significantly. Letting users specify not only keywords, but relating their queries to conceptualized domain knowledge using the proposed annotation approach will enable reasoning algorithms (a) to expand the user queries, (b) to increase the number of relevant records in the repository, and (c) to return a more precise result set in the end. Hence, semantic descriptions can be used to increase recall, i.e. discovering all services which fit the user requirements, and precision, i.e. including only relevant service in the result set.

Discovery of semantically annotated processing components with focus on operations is complex, and requires more sophisticated reasoning capabilities. Here, various kinds of matches have to be applied. The classical approach, the exact match, has to be separated from the predicate match, plug-in match, and an extended plug-in match (Zaremski 1996, Lutz 2006).

*Predicated matches* compare requested process descriptions with the pre- and post-conditions of the advertised services in one matchmaking step. Predicate matches are especially useful for simple queries such as *"give me all services that compute overlay"*. The predicate match fails if a requested precondition is more specific/restrictive then the precondition offered by some Web Service, although the Web Service could execute on the input provided by the requester. Hence, these matches may decrease *recall*, that is: some WPSs that provide the requested functionality are possibly not retrieved. If the provided precondition is more specific then the requested one, *predicate match* succeeds although an acceptable input cannot be guaranteed. Hence, this match possibly decreases *precision*, that is: some WPSs that are retrieved are possibly not able to fulfill the requested functionality.

The more sophisticated *Plug-In Match* compares the pre-conditions and post-conditions separately. This avoids the drawbacks from the predicate matches. It is still impossible to consider dependencies between inputs and outputs using this type of match. Having two polygons (A and B) as inputs, it is impossible to identify if it is difference (A – B) or (B - A). This type of match is not aware of possible permutations of the input variables. Since this can possibly deliver wrong results, it may decrease *precision* as well.

For a non-relaxed match, i.e. a match with high precision and high recall, the *Extended Plug-In Match* was developed. This algorithm is able to incorporate all the elements of a process description. But due to its complexity and the dependency on expressive ontology languages, it is very resource intensive and not (yet) applicable for large-scale applications. SWING D3.2 (Schade et al, 2006) contains more information on possible implementations using WSML-Flight, an ontology language variant based on logic programming.

### 1.5.2 Geographic Information validation

Once Geographic Information has been discovered, the agent (user or machine) has to evaluate if the available spatial dataset matches the user requirements. Several aspects play an important role here, and not many have been investigated sufficiently (like Service Level Agreements, Quality of Service parameters, License issues, Security, ..). The use of ontologies to describe specific aspects of a Web Service, or the served data, is a flexible and easily extensible approach. Technologies like RDF which are used to encode ontologies account for the dynamics of the modeled knowledge. They can easily be extended with additional (non-ontological) information without the need alter the underlying schemata. The flexibility of reasoning algorithms doesn't restrict metadata to specific content, they only reason on the relevant ontological subset of the document.

The evaluation of the thematic dimension of the spatial data is obviously improved as well. Since we are not restricted to ambiguous, error-prone and usually missing keywords and textual descriptions any more, it is now easier to evaluate what the discovered service represents. Formalized concepts on the domain level are sophisticated descriptions, including label in different languages, relationships to other concepts, and many more. Since these concepts are encoded in ontologies, the extent of potential information about them is not constrained.

### 1.5.3 Validation of Service Workflows

If we consider Web Services as atomic components, which either serve or process geospatial data, we also want to be able to compose more complex services by creating service workflows. Data services like the WFS or WCS deliver geospatial data, which is then processed and modified by a WPS. The output of the WPS could be again a Feature Collection, Coverage, or an image. Depending on the resulting data, the whole service composition will be encapsulated again as WPS, WFS, WCS, or WMS.

Creating the workflows is a matter of adding existing Web Services to the present composition and, at the end, hand over the workflow document to a workflow engine. Since only OWS are used, syntactic interoperability of the service components can be assumed. GML is output of the data services, and GML is the expected input of the processing services. Semantic interoperability, on the other hand, can only be achieved if the used Web Services are semantically annotated. Reasoners can match the outputs with the input of the subsequent service, and are therefore able to validate if the workflow not only syntactically, but also semantically valid.

### 1.5.4 Semantically supported integration of geospatial data sets

Integrating spatial datasets in any form assumes that both datasets match in geometry, structure and semantics. Geometry simply refers to the spatial nature of the represented data; both datasets are representing the data in the same way, for example in form of points in the case of feature types representing our quarries. Structurally matching feature types have either equal sets of attributes, or routines which are used to translate between attributes. A syntactical match is ensured if the same standard, meaning the same version of GML, is used for encoding the data.

Such tasks don't consider the semantic dimension yet. Two datasets may be compatible in their geometric, syntax and structure. But this doesn't imply that each attribute present in both feature types with the same identifier has the same meaning. The feature type of our example has the attribute `allowedproduction`, which is measured in tons per year. Another dataset my have the same attribute, but its value refers to the allowed weekly production. Integrating the two datasets would clearly result in unusable data. Semantically annotated attributes and a validation procedure, optionally supported by rules to translate from one value to another, will either warn the user in the case of semantically incompatible datasets, or use the rules to solve semantic conflicts.

Integrating of specific spatial entities is also an interesting application for semantic annotations on the instance level. If two spatially close entities are equally annotated, one can infer that both refer to the same geographic object, which means they have to be integrated. As example, two point features within a distance of less than 100 meters are related to the domain concept QUARRY. It is reasonable to assume that both features refer to the same geographic object, they can be combined to one feature. Rules formalized in the domain knowledge can support such assumptions, and can be automatically applied either during the annotation of instances, or during the integrating of datasets.

## 2. Semantic annotations at three different levels

Modelling knowledge and publishing it in a well-defined and machine interpretable format can result in increased usability of OGC Web Services. We consider ontologies as most promising format to capture such knowledge. But the possibilities to connect ontologies with OGC services are manifold. In this section we discuss how we can annotate OGC services on the already mentioned three levels:

1. the service metadata,
2. the data models and process descriptions, and
3. the actual data instances in the database.

The three annotations levels differ in their potential. This is due to different reasoning capabilities, i.e. the abilities to infer either new knowledge (making implicit knowledge explicit) or to identify conflicts in existing models. Accordingly, the applications for semantically supported OWS discovery and workflow validation vary. Requirements vary between projects, the following comparison of the three levels with implications on reasoning and potential benefits can therefore only outline possibilities.

Notably, although we illustrate semantic annotations in the OGC context, they can be similarly defined in relation to ISO/TC211. In particular the ISO General Feature Model (GFM) described in ISO 19109 and ISO 19115 (Metadata) has to be considered. In GFM, geospatial data is modelled as features and their properties and associations. Metadata can be associated for each of these elements.

Following our suggestion, a semantic annotation would either link instances of the GFM, i.e. concrete feature types, feature attributes and feature operation, or their instances, i.e. concrete feature collections and their content to concepts of an ontology. The first approach reflects level 2, the second level 3. Level 1 annotation would relate to service metadata (ISO 19119).

In any case, a semantic annotation can be seen as a specific kind of metadata and should therefore be considered as an official metadata element.

## 2.1 Implications for geospatial applications

Semantic annotations extend the already existing annotations like the OGC Capabilities document by linking particular metadata elements to conceptualized domain knowledge. These links point to one or more concepts, which are modelled using ontologies or the less formal taxonomies. The Quarries WFS from the example in figure 1 (p.8) can, for example, be annotated with the concept QUARRY, MINE, and AGGREGATE. Sophisticated semantic annotations include axioms, i.e. combinations of logic statements, which relate various domain concepts to each other in order to elaborate the internal relationships of the data. The feature type `exploitationponctualsproduction` has, for example, the attributes `allowedproduction` and `year`. With the help of axioms we are able to specify that `allowedproduction` is measured in Tons per Year, and that this Year is related to the other attribute `year`. We have successfully investigated such complex semantic annotations in the mentioned SWING project, detailed explanations are available in the deliverable 3.2 from the project web site (Schade et al, 2008).

### 2.1.1 Level 1 - Service Metadata

Adding knowledge model references in one of the existing sections in the data and/or services metadata (e.g. by extending the keyword section of OWS Capabilities) is the most pragmatic and still useful approach to semantic annotations. In our example, only limited information is available in the service metadata section (see Figure 1). The name "MapServer WFS" is automatically generated and bears no meaning, the title "Quarries" and the keywords "brgm, quarries, quarry, france" provide an only very limited description of the served geospatial data. Being able to semantically enrich metadata sections like the keywords makes understanding of the intended meaning a reality.

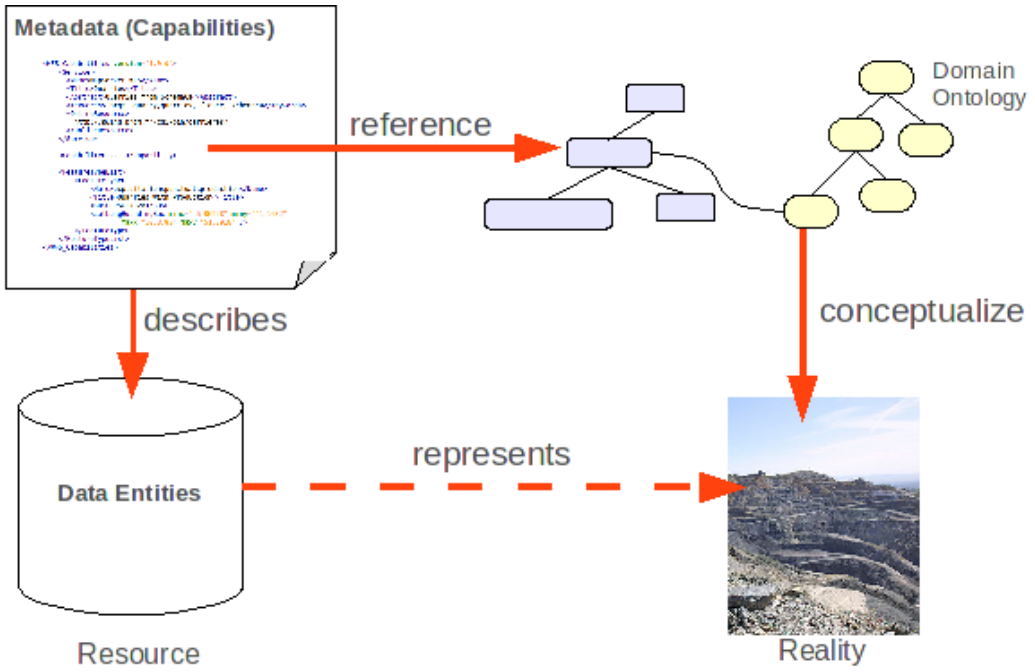


**Figure 9 - Annotations as part of the resource metadata**

Keywords themselves might act as the model references. They can be direct links (orange "reference"-arrow in figure 7) to the concepts or axioms in the domain ontology. This approach extends the notion of keywords to serve not humans only by becoming also interpretable by machines. In addition to the keywords "quarry" "france" and "brgm", we can add links to concepts within a shared knowledge model (which can be as simple as a thesaurus or as complex as domain ontology). This model is then a conceptualization of the reality, the concept QUARRY denotes real quarries.

The main benefit of this approach is low implementation costs, since we only extend the notion of existing metadata sections. But changing the purpose of this section to include model references decreases their usefulness in terms of readability for human users. Textual descriptions are one major source of information for evaluating the served data. Having cryptic URIs instead of terms does obviously not improve acceptance by the readers. In section 3.2.2 we list examples how adding concept identifiers to existing metadata sections may look like. In section 2.2.1 we introduce the idea of visualization tools to improve the readability by human users.

Looking at the discussed applications for semantic annotations, the discovery and evaluation of geospatial data already benefits from this even very light-weight approach. But the improvements just affect tasks concerned about the service metadata level. Replacing keywords with model references cannot cover the complexity of the data served by an OGC Web Service, because no formal connection to the served data has been established. The reasoning only works on the service metadata level. As result the GIR system can only infer that a discovered service is in some way related to the domain concept. Unless the semantic annotations are not directly applied to served resources, i.e. feature types, map layers, a searching user would not able to

identify the resource providing the needed information. In the case of WFS, he wants to know which feature types serve the information associated with this domain concept. And within a feature type, he wants to identify which attributes bear the requested information. In the following section we discuss semantic annotations directly attached to the resource, making it therefore possible to overcome these problems and enabling more sophisticated reasoning tasks.

### 2.1.2 Level 2 - Data Models

In order to relate service metadata more efficiently to domain knowledge, we need to take a closer look at the data models and the service operations. In this section we further discuss the annotation for data as it served by a WFS or a WCS. In the following section we address processes and operations, which are usually made available using the WPS standard. Considering data structures (like GML application schema) for semantic annotations enables reasoning on data model level. In the example of figure 8, the feature type and the feature attributes have a reference to concepts in the resource ontology. Within this ontology we are now able to relate the local service-specific concepts, which are directly associated to the data, to the global domain concepts (see Figure 6)

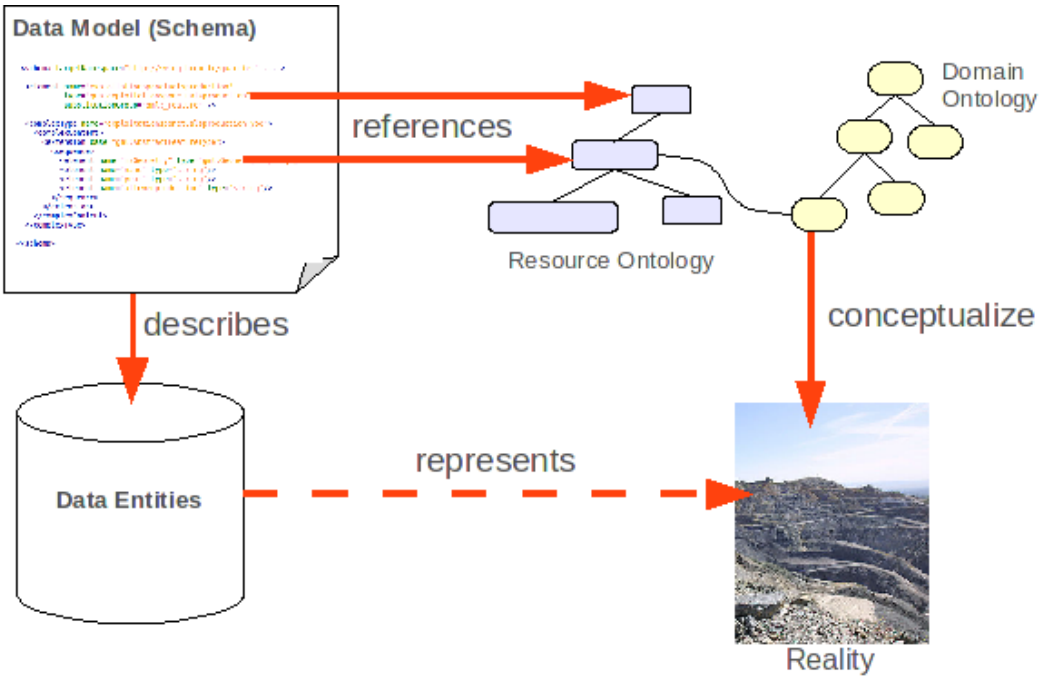


**Figure 10 - Referencing elements in the data model to domain ontologies**

Defining the semantics of a feature type, i.e. using an application ontology to capture the meaning of the elements and attributes used in the data model, enables reasoning on a more detailed level. Again, the specific reasoning capabilities rely on the expressiveness of the chosen logical language. Reasoner-supported IR systems for geospatial Web Services and data sets based on the semantic annotation approach have been implemented successfully (see Schade et al, 2008 and Hoffmann et al, 2008). With semantically and syntactically correct annotations we have shown

that discovery, in term of precision, can benefit significantly. It depends, however, on sophisticated semantic queries to benefit from the full complexity of the semantic annotations on this level. As explained in section 2.2.1 , we have investigated user interface which support the user in creating such more complex queries.

Even if there is no exact match between a user's query and the semantically annotated Web Service, the annotations help to increase recall. Once discovered, the annotation can be used to identify the exact attributes containing the desired information (see figure below). The references have to be followed the other way, from the ontology back to the data model and its specific elements. In the example, such links indicate that the attribute `allowedproduction` contains information about the allowed production of a specific substance in tons per year, the name information considers the name of the owner of the quarry, etc.

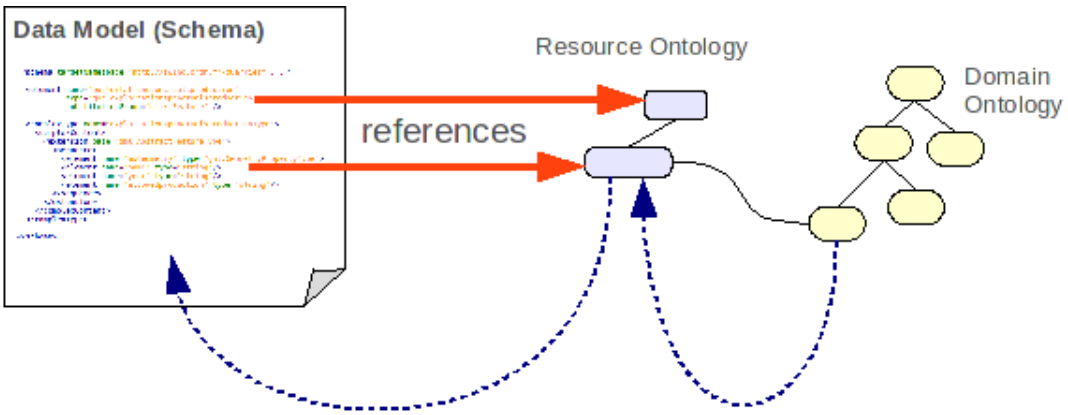


**Figure 11 - Using references to identify relevant elements in the data model**

### 2.1.3 Level 2 - Geospatial Processes

Approaching semantic discovery of processing functionality requires more then a Resource Ontology describing the output from a Web Service. As illustrated in the following figure, it is not sufficient to provide a domain vocabulary including all possible processes a Web service can implement. It is additionally required to model not only the output of the Web service (as we do it for the data models), but also the expected input data of the process and the dependencies between the output and the input (e.g. saying the resulting output area is the union of the two input areas). Taking not only the output (the postconditions), but also the input (preconditions) into consideration demands for more sophisticated reasoning capabilities, and is for example discussed in more detail in the Hoffmann et al. (2008).

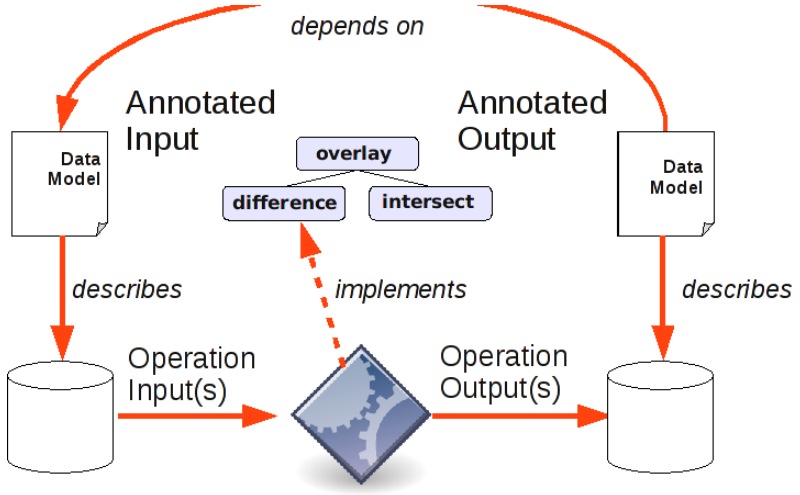


**Figure 12 - Elements for the Annotation of Processes.**

Due to the expected complexity we assume semantic annotations of processes is (for now) only applicable for a small number of applications. The semantic validation of workflows including processing functionality depends on the semantic annotation of all components. It is additionally useful for tasks where specific processing capabilities within a large pool of WPS are searched. Annotating processes is also interesting if non-standard operations are implemented and served via Web Services not conformal to the OGC Standards. In the case of WFS, WCS, and other OWS focussed on data only, the annotation can focus on the data since the semantics of the operations are predefined.

### 2.1.4 Level 3 - Data Entities

Semantic annotations are also possible on the data entity level. Regarding OGC Web Service, GML features and feature attributes are the entities which have to be annotated here. We can distinguish between two potential techniques: The individual entities can be either annotated with domain concepts or with individuals on the domain level. A specific quarry can, for example, be a chalk pit. A user might want to express this information by annotating it with the domain concept CHALKPIT. If this particular chalk pit is modelled in the domain ontology (in this case as individual of the concept CHALKPIT), he might even want to directly associate the data entity with this individual.Having annotations on the instance level has several implications on discovery and evaluation of geospatial resources. The annotations can be used to search for a subset of features (in this case only data entities representing chalk pits) by filtering out other features. The information available for individual entities can be increased significantly. Since ontologies are, by nature, a very flexible model, additional information can be added on runtime. Adding additional documentation to individual data entities is therefore simply a matter of adding it to the associated Concept/Individual in the Resource Ontology. One specific application, the merging of datasets including quality control, can benefit tremendously from semantic annotations on the data entity level. During merging, features annotated either with the same

individual can be regarded as equal. In addition, rules can be applied to formalize constraints on the thematic dimension to reduce error in data sets. This would, for example, not only ensure geometric consistency of datasets (a street network has to be connected), but also semantic consistency (features annotated with the domain concept HIGHWAY can not be directly connected to features representing a PEDESTRIANZONE).Technical realization of the individual levels

## 2.2 Technical realization of the individual levels

Up to now the implications on the applications have been discussed, together with drawbacks and benefits for each level. In the following three sub-sections we discuss how the annotations shall be technically realized.

### 2.2.1 Level 1 - Service Metadata

Only sparse information is available for the Quarries WFS. The name "MapServer WFS" is automatically generated and refers to the implementation, the title "Quarries" as well as the keywords "brgm, quarries, quarry, france" provide an only very limited description of the served geospatial data. The most light-weight and simplest implementation of a semantic annotation is to extend the keyword section by adding pointers to domain concepts. Another option is to use the METADATA field which is proposed as Service Content Metadata in WS-Common. The standard does not specify what types of metadata are meant to be entered here. It would be therefore possible to add the list of domain concepts to this field as well.

Each concept is uniquely identified by an URI, the accepted standard to encode locations in the World Wide Web. Using an URI as concept identifier presumes either a predefined knowledge base where such concepts might be retrieved from, or that the URI is also an URL which can be simply accessed using a standard Internet browser.

In Section 2.1.1 we argued that adding such URIs to the service metadata section have a negative impact on the readability. One potential solution are tools to visualize the concepts and their relations on the domain level as in figure 11. This tool has been developed within the SWING project to make the creation of semantic annotations also possible for non-experts. A tool bundled with this visualization component is available for download[12], a video[13] demonstrates its usage.

[12] http://swing.brgm.fr/mimsalpha/download

[13] http://swing-project.org/Demos/WP1/WP1-MiMS.html

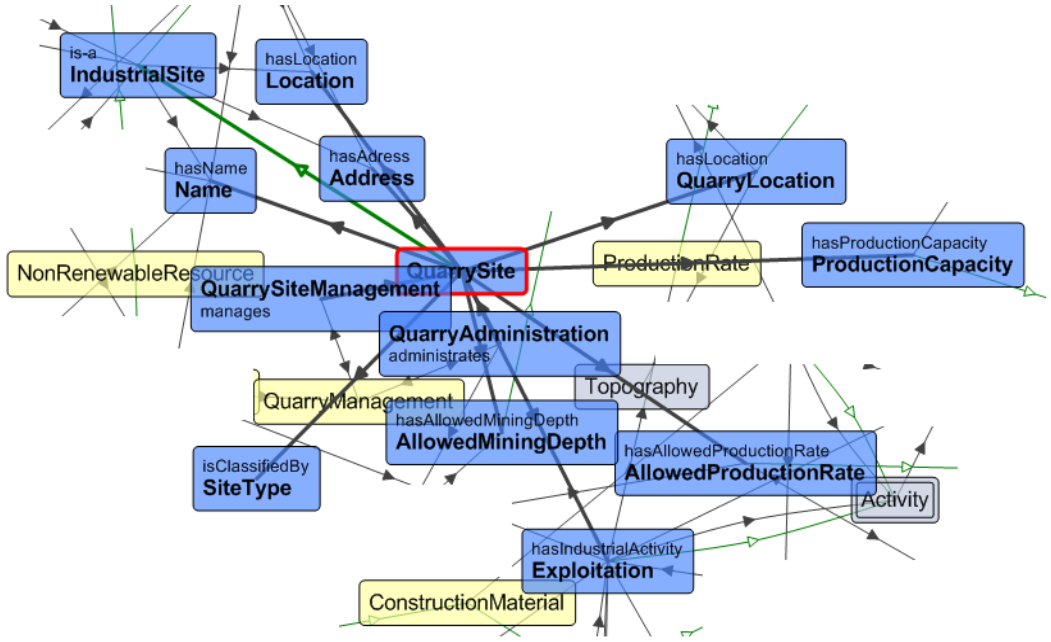


**Figure 13- Result of the translation from data model to Resource Ontology (using the developed visualization tool).**

## 2.2.2 Level 2 - Data Models

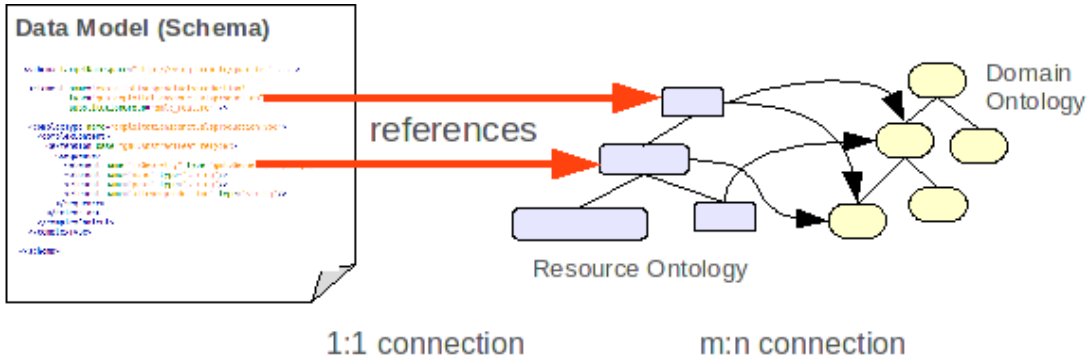


**Figure 14 - Using Data type Ontologies for flexible annotations.**

To benefit from reasoning on the data model level, the data schema is associated with its Resource Ontology. In the first step, the existing data schema can be simply translated into the according Resource Ontology. In figure 13 the result of such a translation is visualized (the resulting WSML code is listed in Figure 5. The association between the concepts within the XML-based data schema and the Resource Ontology is established by adding an additional attribute to the selected elements in the schema, with the identifier (as URI) of the concept as value. GIR systems like a CAT can now retrieve, for each registered Web Service, the

persistently linked Resource Ontology, and infer if a user's semantic query matches the semantic description of the service.

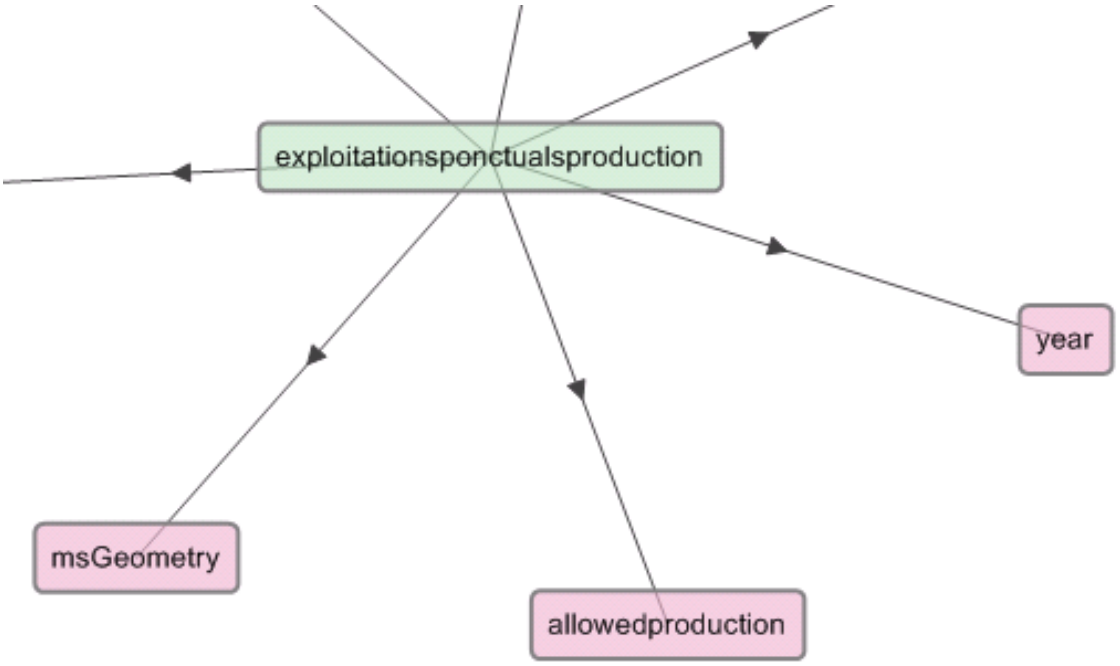


**Figure 15 - Result of the translation from data model to Resource Ontology (using the developed visualization tool)**

In 2007, the W3C submission "Semantic Annotations for WSDL and XML Schema (SAWSDL)[14] officially reached the W3C Recommendation status. This very light-weight standard suggests the use of the `modelReference` attribute to relate elements in XML schema (as well as elements in WSDL documents, which is not (yet) of interest for the OGC community) to concepts in knowledge models. The standard does not define the language used for the knowledge models, nor does it prescribe the complexity of the model. It can be as simple as a controlled vocabulary or a Topic Map or as complex as an ontology. Having an URI (see section 2.1.1 ) as unique identifier for the concepts in the knowledge model is the only prerequisite.

The SAWSDL standard suggests two additional attributes to enable a mapping between the language used to model the ontologies and the original data schema. If included, the two attributes `liftingSchemaMapping` and `loweringSchemaMapping` can point to XSLT documents which enable automatic transformation between two schemas (assuming the ontologies are encoded in a XML-based language). This transformation is addressing the data entities, which makes it possible to transform data coming from a service into entities processable by the reasoning algorithm. The potential benefits for applications which have to consider data quality and processing of data in general are enormous, but discussing this is clearly out of scope for this article. Examples how to add SAWSDL model references to existing data schemas and the implications on existing OGC standards, are further discussed in section 1).

[14] Find the recommendation here: http://www.w3.org/TR/sawsdl

The annotation process can be supported using graphical tools (see figure 14). Within the Resource Ontology we have the concept EXPLOITATIONSPONCTUALSPRODUCTION which has several relations to other concepts representing the attributes. Two concepts, the YEAR and the ALLOWEDPRODUCTION, have been related to concepts in the (global) domain ontology. The user is able to simply add domain concepts to the view, and establishing either an annotation link between service concepts and domain concepts (the red lines), or relations between domain concepts (the black lines). In this example we have now formally specified that the value of the attribute `allowedproduction` is representing the PRODUCTIONCAPACITY of the Quarry, measured in TONSPERYEAR. The YEAR, which is the negative exponent of the Unit Tons per year, is the value of the attribute `year`.

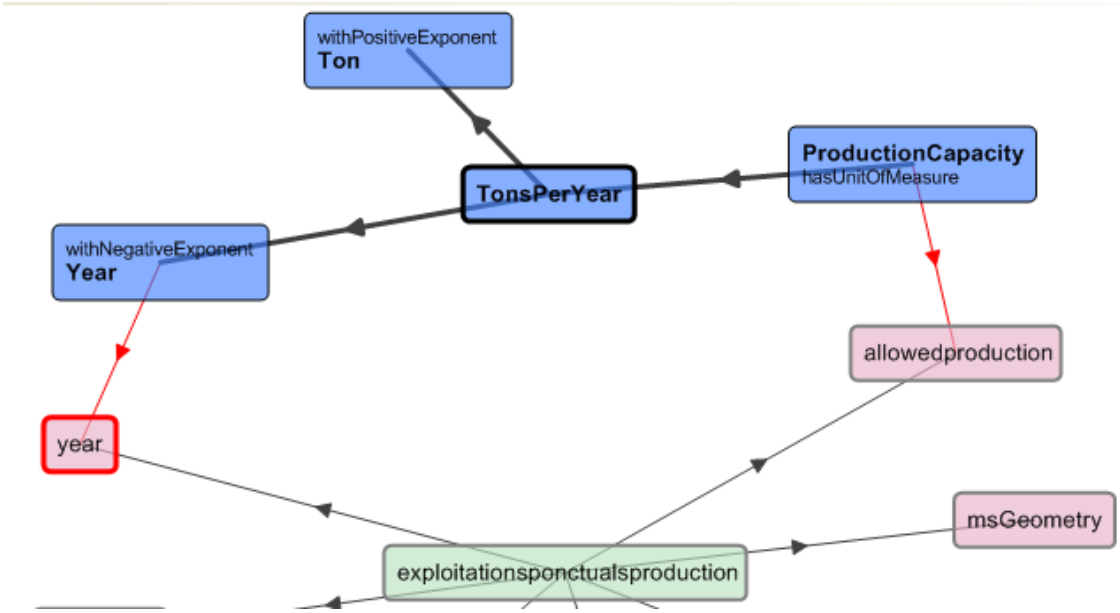


**Figure 16 - Annotation of the feature type `exploitationsponctualsproduction` (using the developed visualization tool)**

The following listing shows an example of WSML code expressing the annotation from figure 14 (which is automatically generated by the visualization tool). See figure 5 for the concept this axiom is referring to.

```
?ft[Year hasValue ?attr1,
    AllowedProduction hasValue ?attr2 ,
    msGeometry hasValue ?attr3]
  memberOf exploitationpunctualsproduction and
```

```
/* referencing the attributes to domain concepts */
?year memberOf domain#Year and
domainReference(?attr1, ?year) and
?capacity[hasUnitOfMeasure hasValue ?uom]
  memberOf domain#ProductionCapacity and
domainReference(?attr2, ?capacity) and

/* the production capacity refers to the attribute year */
?uom[withNegativeExponent hasValue ?year]
  memberOf domain#TonsPerYear .
```

**Listing 1 - Example of WSML annotation using the Resource Ontology**

### 2.2.3 Level 2 - Geospatial Processes

The description of *type signatures* (the input/output types) is a crucial part of *functional descriptions*. It ensures syntactic interoperability between requester and a Web Service and therefore guarantees that a Web Service can process the provided input type and that a requester can accept the delivered output type. Such descriptions can be realised using annotations with Resource Ontologies as described in previously (section 2.2.2 ).

The called, respectively requested, *operation* has to be linked to the appropriate concept representing the operation to make its interpretation unambiguous. The different overlay operations on polygons (difference, symmetric difference, intersection, and union) have all equal type signatures and equal constraints. Additionally, their input and output polygons adhere to a common spatial reference system. It is therefore impossible to distinguish them just by considering (even sophisticated) descriptions of input and output types. Having a unique domain concept like DIFFERENCE or UNION (along descriptions attached to the concepts) associated makes the signature of the WPS better understandable.

Annotations from process descriptions in the OGC Capabilities Document, respectively in the WPS *DescribeProcess*-response can be established in a similar way as described in sections 2.2.1 and 2.2.2 .

Additionally, depending on the chosen logic, constraints on service inputs, and dependencies between the in- and outputs may be formally specified. To ensure that the Web Service really processed the provided input in the intended way, constraints which further narrow down the possible input and output values are required. For example, a WPS computing the difference of two polygons requires all input values (besides being of type polygon) to adhere to a common coordinate reference system. A slightly more complex problem, the possible permutations of the input variables, does additionally require to formalize the dependency between output and input. If we only annotate the inputs, outputs, and the operations using axioms defined in the Resource Ontology, we are not yet able to express such relationships. It would be impossible to distinguish between Difference(A,B) and Difference(B,A), although their results are quite different.

Modelling these dependencies is unfortunately quite complex, and is illustrated here only for the sake of completeness. The intended module within WSMO to capture such dependencies is the WebService, with pre- and postconditions and shared variables. In the preconditions the required input is formalized using axioms, the same is performed for the output. Shared variable are used in both, the pre- and postcondition. An adapted reasoning engine is now able to consider dependencies, using the already mentioned extended Plug-In Match. The following figure has a complete example for the annotation of a WPS using the WebService component.

```
webService DifferenceWPS
 capability DifferenceWPSCapability
  sharedVariables { ?a, ?b, ?srs }

precondition UnionWPSPrecondition definedBy
  ?a[iso19107#hasSRS hasValue ?srs] memberOf iso19107#Polygon and
  ?b[iso19107#hasSRS hasValue ?srs] memberOf iso19107#Polygon and
  ?srs memberOf iso19107#projSRS.

postcondition UnionWPSPostcondition definedBy
  ?c[iso19107#hasSRS hasValue ?srs] memberOf iso19107#Polygon and
  GeoOperations#Difference(?a, ?b, ?c).
```

**Listing 2 - Complete description of a WPS in WSML**

### 2.2.4 Level 3 - Data Entities

Similar techniques as the one used for annotating data models (see section 2.2.2 ) can be applied to annotate data entities like feature instances. Supporting annotations on this level seems to be only useful for feature data provided by an OGC WFS. The structure of the language used to encode the data (GML) is predefined in the associated XML schema. Having additional attributes therefore requires modifications of this schema. This is discussed in detail in section 3.3.2 , p.45.

User interfaces supporting annotations on the instance level have to let the user attach links of concept definitions directly to features on a map. From a technical perspective there is no difference between annotating data models or data entities. The same technique and user interface strategy as discussed in section 2.2.2 (p. 32) can therefore be applied to annotate features.

## 2.3 Comparing the capabilities of the three approaches

We compiled a summary of the advantages and drawbacks of the three levels in a table, which is available as Appendix B starting on page 50. Note that we were only able to collect experience on semantic annotations of data models and processes. We are not aware of every potential argument in favor or in opposition to the individual techniques. As outcome to the discussion accompanying this paper we expect a more thorough overview of the drawbacks and benefits, and

reference documentation which guides the application developer through the implementation of geospatial applications supported by the semantic annotations.

# 3. Implications for existing GIS Standards

This section analyzes how the various standards involved in the GIS world may be affected by the proposed techniques, i.e. where would the use of model references fit best, and what would be the benefits or drawbacks. For each of the standards covered in the following subsections, the level of annotation that is relevant to address with that standard is considered. Metadata standards will mostly be used to do first and second level annotations (resp. service and data model), while data representation standards are the place to implement third level annotations (data instances).

Finally, because annotating resources is useless without a way to query those annotations, OGC Filters, as a query encoding standard, is the subject of the last section, and ways to express filters based on semantic annotations are considered. The proposals made in this section can be seen as grounds to build a set of best practices and change requests to generalize and harmonize semantic annotations across GIS standards.

## 3.1 Expressing semantic annotations

Before analyzing existing standards, it is best to first devise how semantic annotations should be ideally expressed.

A semantic annotation *must* contain the **URI** of the classifying concept. That is the sole mandatory and sufficient piece of information needed to fully qualify the annotation.

Extra attributes should be provided for convenience and performance, namely a **label**, **language**, the **thesaurus URI** and **thesaurus label** that the concept (or semantic entity in the broad sense) belongs to. These attributes, although not required to formally identify and process the annotation, are useful to implement tools that do not need to resolve and reach the potentially remote concept definition, thereby offering better performance and usability. These attributes should therefore be optional.

As a result, a valid example (outside the scope of any standard analyzed below) of a semantic annotation would be (using the GEMET thesaurus[15]) :

```
<annotation conceptUri="http://www.eionet.europa.eu/gemet/concept/4612"
            thesaurusUri="http://www.eionet.europa.eu/gemet/"
            thesaurusLabel="GEMET - INSPIRE Themes, Version 1.0"
            lang="en">
     Land Cover
```

[15] In this GEMET example, and others further down, the thesaurus URI is set to http://www.eionet.europa.eu/gemet/ , although the right value is unclear from available online resources.

```
</annotation>
```

where *thesaurusUri*, *thesaurusLabel*, *lang* and the element content (i.e. the concept label) are optional, i.e. the minimal valid annotation would be :

```
<annotation conceptUri="http://www.eionet.europa.eu/gemet/concept/4612"/>
```

**The example above assumes that the `annotation` element is located *within* the element being annotated.** Like for example in the OGC capabilities a `Keyword` element is located within the `Service` element being annotated.

If this is not the case, and the annotation stands on its own, then it must also hold a reference to the entity being annotated, i.e. (re-using the minimal annotation above) :

```
<annotation conceptUri="http://www.eionet.europa.eu/gemet/concept/4612"
            annotatedElement="http://server/element/1234"/>
```

This assumes that the elements being annotated can be referenced by URI, too.

Standards analyzed below will always present the first case, i.e. the annotation being included within the annotated element. The second case presented in the previous paragraph will therefore not be considered further.

Each standard below will be analyzed to see to what extent it can contain such an entity, and what changes or workarounds would be required to do so, if any.

Related work at the W3C is also worth considering in the scope of this Best Practice : http://www.w3.org/2005/Incubator/ssn/wiki/Report_Work_on_Semantic_Markup .

## 3.2 Metadata standards

Various organizations introduced standards for metadata, both for geospatial data and services. In this section we review standards of three common fields: ISO/TC211 metadata standards, the service capabilities as defined by OGC, and ebRIM, as introduced by OASIS.

### 3.2.1 ISO 19115 & 19119

ISO 19139, as the encoding standard for metadata compliant to ISO 19115/19119, is the first standard to consider when including the references needed for semantic annotations. Looking at the whole ISO 19139 structure, the most appropriate place to introduce Level 1 semantic references seems to be the `gmd:MD_Keywords` element. This element is meant to hold sequences of keywords, using `keyword` elements, and with extra attributes defining the context, such as the `type` or `thesaurusName`.

```
<gmd:MD_Keywords>
    <gmd:keyword>
        <gco:CharacterString>keyword1</gco:CharacterString>
    </gmd:keyword>
```

```
    <gmd:keyword>
        <gco:CharacterString>keyword2</gco:CharacterString>
    </gmd:keyword>
    <gmd:type>
        <gmd:MD_KeywordTypeCode
            codeList=<anyURI>
            codeListValue=<anyURI>/>
    </gmd:type>
    <gmd:thesaurusName> ... </gmd:thesaurusName>
</gmd:MD_Keywords>
```

**Listing 3 - ISO 19139 Structure**

This ISO19139 structure can be used without modifications to hold semantic references to concepts from a given ontology. If we assume that (see section 2.2.1 ) any concept can be uniquely referenced by a URI that can be split as a base URI and a concept ID:

| | <ontologyURIBase> | <conceptId> |
|---|---|---|
| e.g. | | |
| for HTTP URIs: | http://www.example.com/Ontology# | QUARRY |
| for URNs: | urn:x-test:landtypes: | CROPS |

**Listing 4 - Unique Identifiers**

Then the keyword elements can be used to hold the concept names describing the resource, and the codeList attribute of the type element can be used to hold the URI of the ontology containing the concepts. Using this approach, the following ISO19139 excerpt would contain valid references to such concepts[16] (Guidelines regarding the proper use of the `type` and `thesaurusName` elements are needed to ensure that this approach is valid).

```
<gmd:MD_Keywords>
    <gmd:keyword>
        <gco:CharacterString>Quarry</gco:CharacterString>
    </gmd:keyword>
    <gmd:keyword>
        <gco:CharacterString>brgm</gco:CharacterString>
```

[16] For the rest of this document, most examples will use HTTP URI as concept reference examples. Please keep in mind that URN references may also be used. This will especially be true when examples related to the catalog and ebXML will be described.

```
        </gmd:keyword>
        <gmd:type>
            <gmd:MD_KeywordTypeCode
                codeList="http://www.example.com/Ontology#"
                codeListValue="ontology"/>
        </gmd:type>
        <gmd:thesaurusName>
            ...
        </gmd:thesaurusName>
    </gmd:MD_Keywords>
```

**Listing 5 - Concrete Example of ISO 19139 Structure**

However, it would be better suited to be able to define keywords belonging to various thesauri, each having its own URI. To this end, the `gmx:Anchor` element (defined to point to registered definitions, according to the ISO specification, and usable as a substitute for `CharacterString`) is better suited (here using the GEMET thesaurus for the sake of the example):

```
<gmd:MD_Keywords>
    <gmd:keyword>
        <gmx:Anchor
         xlink:href="http://www.eionet.europa.eu/gemet/concept/4612">
          Land Cover
        </gmx:Anchor>
    </gmd:keyword>
    <gmd:thesaurusName>
        <gmx:Anchor xlink:href="http://www.eionet.europa.eu/gemet/">
           GEMET - INSPIRE Themes, Version 1.0
        </gmx:Anchor>
    </gmd:thesaurusName>
</gmd:MD_Keywords>
```

**Listing 6 - Concrete Example of ISO 19139 Structure**

Here we see that the `gmx:Anchor` element provides a way to substitute a mere string with a qualified and labeled entity (concept or thesaurus). It thereby fulfills the requirements for the abstract *annotation* element described in section 3.1 .

Regarding semantic annotations at the second and third level (data models and entities), model references have to be introduced as explicit metadata element. ISO 19115 foresees two options for this:

The abstract element `MD_ContentInformation` could point to ontological descriptions. Coverages and feature catalogue descriptions are currently the only non-abstract children of the

MD_ContentInformation class. Both provide potential extension points to include characteristic parameters and natural language descriptions for feature types and properties. Neither follows the distinction between information objects and models of the world. An interesting option would be to include a third class "MD_Reference" to this branch of metadata elements, which could contain explicit pointers to resource or domain ontologies.

The element MD_ApplicationSchemaInformation could also be used to link to ontologies. It offers the possibility to refer to an application schema used for a specific data encoding. Following ISO 19139, this is usually the XSD specifying the encoding of geospatial information items. Once a pointer to such implementation model is available, semantic annotations can be established from there. Details considering this approach are given in section 3.3 . This approach would not require any change of the recent metadata standards, but a specific implementing rule.

### 3.2.2 OWS Capabilities

To be able to efficiently describe and discover services using semantic annotations, the OWS specification must be considered. The OWS Common specification is common to several OGC services (WCS, WFS, CSW, …) and defines the way a service can advertise its content, mainly through the specification of the GetCapabilities operation response. Again, a Level 1 semantic annotation could be established using keywords.

**Using Keywords**

The definition of the content of a GetCapability response reproduces partially the ISO19139 structure described above in its ows:Keywords element. In particular, the type element is mapped into a ows:CodeType xml element that retains the properties needed for the approach described in the previous section.

Therefore, as a direct consequence, we can apply the same approach to the OWS Capabilities; using this approach, the GetCapabilities document of an OGC service would look like the following:

```
<ows:ServiceIdentification>
  <ows:Title>Some WFS Service</ows:Title>
  <ows:Keywords>
    <ows:Keyword>Quarry</ows:Keyword>
    <ows:Keyword>brgm</ows:Keyword>
    <ows:Type codeSpace="http://www.example.com/Ontology#">
     ontology
    </ows:Type>
  </ows:Keywords>
  <ows:ServiceType>WFS</ows:ServiceType>
  <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
</ows:ServiceIdentification>
```

**Listing 7 - Annotation of the GetCapabilities-Document**

Here the `Keywords` element is shown as part of the `ServiceIdentification` element, i.e. the service description. It can also be used at the level of resources of the services, e.g. feature types for the WFS, coverage offerings for the WCS, process descriptions for the WPS, and more.

It must be noted that there can be only one 'Type' element per 'Keywords' section, hence having all keywords within a 'Keywords' section belonging to the same thesaurus (if any). If keywords from different thesauri are needed, multiple 'Keywords' sections can be used.

This construct is not as flexible as the abstract *annotation* element defined above. In particular, there is no way to express a conceptURI that is not an extension of its thesaurusURI. Therefore, a change request should be issued to add the ability to have full-fledged annotations in OGC capabilities, i.e. having each a mandatory concept URI, and optional label, language and thesaurus URI/label. Much like what `gmx:Anchor` offers.

### 3.2.3 WMS Capabilities

This approach is valid for all OGC services, with one noticeable exception, though: the WMS specification, as per its 1.3.0 version, does not comply with the OWS Common specification. As a result, the `Keywords` element is replaced by a `KeywordList` element, and the `Type` element does not exist.

The WMS specification should comply with the OWS Common specification by its next version. In the mean time, if the WMS specification in its current state must be used with semantic references, the only way is to use the `vocabulary` attribute available on each `Keyword` element, defined by the specification as "*an attribute to indicate the defining authority for that keyword*". The `Keyword` content would then be the concept URI. Therefore, the semantic annotation becomes :

```
<WMS_Capabilities version="1.3.0">
  <Service>
      <Name>WMS</Name>
      <Title> Some WMS Service </Title>
      <KeywordList>
          <Keyword vocabulary="http://example.com/Ontology#">
              http://example.com/Ontology#Quarry
          </Keyword>
          <Keyword vocabulary="http://example.com/Ontology#">
              http://example.com/Ontology#BRGM
          </Keyword>
      </KeywordList>
```

**Listing 8 - Metadata Annotation for WMS**

However, this scheme prevents the expression of the concept label. This is a limitation of the WMS capabilities as they stand today.

### Using MetadataUrl

The WFS and WMS specifications define a `MetadataUrl` element that provides a link to metadata documents describing respectively FeatureTypes and MapLayers. Such `MetadataUrl` elements have a format attribute defining the metadata format; its values are taken from a finite set, containing e.g. ISO19115 or FGDC. This element can be used to provide links to documents that describe the resource semantically. This kind of semantic annotation would not be a mere model reference linking between entities and concepts as described earlier, but a reference to a full ontology including these concepts. In the case of a feature type, such ontology can be used to semantically annotate the feature type structure, as described in section 2.2.1 (p. 31). To achieve this, more formats should be added to the format attribute value set, such as RDF, OWL, SKOS or WSML.

```
<wfs:WFS_Capabilities version="1.1.0">
<FeatureTypeList>
    <FeatureType>
       <Name>rivers</Name>
       <MetadataURL type="0" format="rdf/xml">
          http://example.com/ontology#rivers
       </MetadataURL>
    </FeatureType>
```

**Listing 9 - Using the MetadataURL for the feature types in a WFS**

## 3.2.4 ebXML

Previous sections described how to add model references to metadata documents. However, for this to be useful, semantic annotations must be handled properly by the components that index such metadata documents. So, although ebXML is not a metadata representation language per se, it is the model used to represent metadata in one of the two CSW profile specifications[17]. This section will thus discuss ebXML and see how it can be used to efficiently store such semantic annotations.

[17] The other CSW profile (ISO AP), based on the ISO19115 specification, can follow the recommendations made in section 3.2.1 .

The `ClassificationNode` and `Classification` elements in the ebRIM specification can be used to store respectively concepts and semantic annotations, with all the information described in section 3.1 , and this is how they are used already in the CSW ebRIM profile. Therefore, the CSW ebRIM profile in its current definition already holds the functionality needed to represent model references.

However, as a recommendation, it would be good to suggest that semantic annotations as proposed in the previous sections (ISO19115, Capabilities) be harvested by CSW ebRIM catalogs by representing the model references using the `ClassificationNode` and `Classification` ebRIM elements. That way, semantic annotations contained in the metadata harvested in a catalog would be explicitly expressed as such and are therefore query-able in an efficient way. Examples of such querying will be provided in section 3.4 regarding OGC Filters.

## 3.3 Data representation standards

The previous section explored how to use semantic references in metadata documents describing GIS resources. Tagging resources can also be done at a finer-grained level, as described in sections 2.2.2 to 2.2.4 . To achieve that, we will now consider data representation standards, and how techniques proposed in previous sections can be applied to them.

### 3.3.1 GML Schema

As explained in section 2.1.2 , XML schemas can be augmented with semantic annotations using the SA-WSDL standard, which suggests to add a `modelReference` attribute to XML element type definitions. Since GML schemas are instances of XML schemas, the same proposal can be applied here as well. Here is an example of a GML schema where the SA-WSDL proposal is applied to annotate the schema (for the sake of conciseness, the ontology complete URI is replaced by the XML entity `&quarries;`) :

```
<element   name=”exploitationsponctualsproduction”
           type=” exploitationsponctualsproductionType”
           sawsdl:modelReference="&quarries;exploitationFeatureType">
<complexType name="exploitationsponctualsproductionType">
 <complexContent>
  <extension base="gml:AbstractFeatureType">
   <sequence>
    <element  name="msGeometry" type="gml:GeometryPropertyType"
           sawsdl:modelReference="&quarries;QuarryLocation"/>
    <element name="name" type="string"/>
    <element name="year" type="string"
           sawsdl:modelReference="&quarries;Year"/>
     <element name="allowedproduction" type="string"/>
```

```
            sawsdl:modelReference="&quarries;Community"/>
    </sequence>
   </extension>
  </complexContent>
</complexType>
```

**Listing 10 - Applying SAWSDL to GML Schema**

In this example, one can see how the `modelReference` attribute is used to semantically annotate both the feature type itself, and some of its elements. In terms of specification, this piece of XML schema is valid, since any attribute can be added to elements such as `xs:complexType` or `xs:element`.

Again, here `modelReference` offers the minimal expressivity described in 3.1 , i.e. the URI. The additional attributes (label, language, ...) are not available, and would require a different construct. However, it is arguable that GML schemas need such convenience attributes, as they are mostly meant for machine processing.

### 3.3.2 GML instances

As explained in section 2.1.4 (p.30), the most fine-grained option to semantically annotate data is to do it at the level of each feature. To achieve that in GML data, one can add to its feature type's specific attributes to hold model references for each data instance. In this example, a `modelReference` attribute is added to the type of stone provided by a quarry:

```
<complexType name="quarries">
    <complexContent>
        <extension base="gml:AbstractFeatureType">
            <sequence>
                […]
                <element name="allowedproduction" type="anyURI"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

**Listing 11 - Applying SAWSDL to GML-encoded Data Entities**

Such a solution simply adds a feature attribute that can hold `anyURI` .This is enough to carry the semantic annotations, but a stronger-typed solution may be better, such as:

```
<xs:simpleType name="modelReference">
```

```
    <xs:restriction base="anyURI"/>
</xs:simpleType>

<complexType name="exploitationsponctualsproductionType">
    <complexContent>
        <extension base="gml:AbstractFeatureType">
            <sequence>
            ...
                <element   name="allowedproduction"
                           type="gml:modelReference"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

**Listing 12 - A stronger-typed solution for applying SAWSDL**

By explicitly defining an attribute as being a `modelReference`, component storing such data (e.g. a WFS) will be able to properly index it for further querying.

### 3.3.3 SensorML

It is useful to consider here the SensorML specification, where a similar problem has been tackled at various levels. In this specification, several types of elements need to be semantically annotated, namely observable properties and processes.

As a result, several constructs are available to express semantic annotations.

First, the `AbstractDataComponentType` element type is the abstract type parent of all data components and observable properties, and has an optional `definition` attribute, of type `anyURI`, defined as "*pointing to semantics information defining the precise nature of the component*":

```
<xs:complexType name="AbstractDataComponentType" abstract="true">
    <xs:complexContent>
        <xs:extension base="gml:AbstractGMLType">
            ...
            <xs:attribute name="definition"
                          type="xs:anyURI" use="optional">
                <xs:annotation>
                    <xs:documentation>
                    Points to semantics information defining the
                    precise nature of the component
                    </xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    /xs:complexContent>
</xs:complexType>
```

*Listing 13: Semantic Annotations in SensorML*

Then, the process can be described using process metadata groups. These metadata groups can hold so-called *identifiers* and *classifiers*, that can refer to semantic entities defined in external ontologies (see SensorML specification, OGC 07-000, sec. 8.11). They can also contain keywords qualified by a codespace, as in OWS Common capabilities.

```
<keywords>
    <KeywordList codeSpace="urn:x-nasa:def:gcmd:keywords">
        <keyword>weather</keyword>
    </KeywordList>
</keywords>
<identification>
    <IdentifierList>
        <identifier name="longName">
            <Term definition="urn:ogc:def:identifier:OGC:longname">
                <value>Davis 7817 External Temperature Sensor</value>
            </Term>
        </identifier>
    </IdentifierList>
</identification>
<classification>
    <ClassifierList>
        <classifier name="orbitType ">
            <Term definition=" urn:ogc:def:property:OGC:orbitType ">
                <codeSpace=" urn:ogc:def:dictionary:CEOS:eop:OrbitTypes:v01"/>
                <value>GEOSTATIONARY</value>
            </Term>
        </classifier>
    </ClassifierList>
</classification>
```

As an example of such annotations, with a classifier referring to the ESA OrbitTypes dictionary :

The `Term` element used here is defined in the SensorML specification, and contains

- a `definition` attribute that can hold the concept URI
- an optional `codespace` sub-element that can contain the thesaurus URI
- a `value` subelement that can contain the concept label

SensorML therefore already has several means to specify model references, complying with the minimal constraints expressed in this document. However, it must be noted that some optional elements described in section 3.1 are not supported : *identifier* and *classifier* elements do not provide a way to express the label language, while the *definition* attribute of AbstractDataComponentType only supports the expression of the concept URI.

### 3.3.4 KML

KML itself is a standard to formally encode simple features without explicit definitions of their attributes. We consider KML to be a lightweight example of GML instances; the problems identified there are therefore also valid for KML. It is not an option to extend the KML schema in a way to include the semantic annotations. But in version 2.2.0 of the OGC KML specification, the `ExtendedData` field provides a flexible way to annotate existing features (e.g. the whole Document, individual Placemarks or their collections, the Folders). The following example shows that a Placemark can be annotated with a domain reference linking to a concept from a shared domain vocabulary

```
<Placemark>
  <name>Münster</name>
  <ExtendedData>
    <Data name="http://purl.org/net/concepts/domainReference">
    <value>
      http://www.geonames.org#Muenster
    </value>
    </Data>
  </ExtendedData>
</Placemark>
```

**Listing 14 - Semantic Annotations in KML**

Note that the value of the attribute “name” has to include either the model reference URI (if a resource ontology exist which is specific for this particular KML file) or the domain reference URI. Only the reference URI specifics that we have a semantic reference, the URI itself therefore has to be agreed upon beforehand.

### 3.3.5 Contexts

The Web Map Context specification (as per version 1.1 ) defines a `KeywordList` element, but which can contain a mere list of string keywords. It does not provide any mean to express the thesaurus URI separately from the concept names. Therefore, the only way to use concept references to tag a context is to use full concepts URIs in the keywords list, as in:

```
<ViewContext version="1.1.0">
<General>
    ...
 <KeywordList>
   <Keyword>http://www.example.com/Ontology#Quarry</Keyword>
   <Keyword>http://www.example.com/Ontology#BRGM</Keyword>
 </KeywordList>
```

**Listing 15 - Semantic Annotations in a Context document**

This is not very strong typed, and does not offer a way to express optional attributes described in section 3.1 .

## Future Context specification

A new Context specification is being defined and should be ready by end of 2011. This new specification builds on the Atom format. As a result, semantic annotations in that format will certainly reuse the 'category' element available in the Atom spec, i.e.

```
<feed>
 ...
<entry>
   <category term="Quarry" scheme="http://www.eionet.europa.eu/gemet/" />
   <category term="Quarry" scheme="http://www.eionet.europa.eu/gemet/" />
```

**Listing 16 - Semantic Annotations in an Atom-based Context document**

The *category* element, as per the Atom specification, contains a way to express the attributes described in section 3.1 (apart from the thesaurus label) . As an example, here's how one can define a category with all those attributes in use :

```
<category term="http://www.eionet.eu.int/gemet/concept/4612"
          scheme="http://www.eionet.europa.eu/gemet/"
          label="Land Cover"
          xml:lang="EN" />
```

Here the `term` attribute is used to hold the full concept URI, although the Atom specification originally intends it to be a mere string code defined in the scope of the thesaurus.

The Atom *category* element does not provide a way to express the thesaurus label; this is something to investigate and suggest to the OGC Context group.

## 3.4 OGC Filters

Assuming that there exists data and metadata annotated with semantic references, one must be able to discover them efficiently. If we consider only the OGC services, expressing queries to discover data is mainly done using OGC Filters.

The OGC Filter Encoding Standard does not provide means to express semantic concept matching, not to mention semantic predicates validation. The only operator that comes close to this is in the CQL specification, part of the Catalogue specification, where an optional `ClassifiedAs` operator is defined, although it is not meant to references concepts by URI, as would be needed in the scope of this discussion. So, to be able to discover data tagged with semantic references, a proposal is made to add specific operators to the OGC Filter specification.

### 3.4.1 The ClassifiedAs-Operator

The first operator that would be needed is an operator similar to the `ClassifiedAs` operator in CQL, but taking a concept URI as parameter. This operator would take a property name as first argument `arg1`, and a concept URI as second argument `arg2`. It would return the boolean `true` value if the value of `arg1` references a concept that is equivalent or is a subclass of the concept defined by `arg2`. In terms of XML schema, this can be defined as part of the Filter.xsd schema as:

```
<xsd:element name="ClassifiedAs"
             type="ogc:ClassifiedAsType"
             substitutionGroup="ogc:comparisonOps"/>
<xsd:complexType name="ClassifiedAsType">
   <xsd:complexContent>
      <xsd:extension base="ogc:ComparisonOpsType">
         <xsd:sequence>
         <xsd:element ref="ogc:expression"/>
            <xsd:element name="concept" type="xsd:anyURI"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
   </xsd:complexType>
```

**Listing 17 - How to apply the ClassifiedAs Operator**

This definition can be enhanced by declaring the `concept` element as a new type, e.g. `ConceptRefType`, which would explicitly reference the concept in terms of its thesaurus URI and concept name.

### 3.4.2 Usage

**Usage in CSW ebRIM :**

In the scope of an ebRIM catalog, the `ClassifiedAs` operator can be used to match records that are subject to a classification, where the classified object is the first element of the operator, and the second element is the URI of the ClassificationNode used in the classification. It must be reminded here that the URI of a concept can be a URN, and in particular can be a UUID.

Using this operator in CSW queries will help performing semantic discovery on service annotations, and on data model annotations:

```
<csw:GetRecords  xmlns:csw="http://www.opengis.net/cat/csw"
       xmlns:ogc="http://www.opengis.net/ogc"
       xmlns:gml="http://www.opengis.net/gml"
       version="2.0.0" outputSchema="EBRIM" maxRecords="50">
    <csw:Query typeNames="ExtrinsicObject">
        <csw:ElementName>/ExtrinsicObject</csw:ElementName>
        <csw:Constraint version="1.0.0"><ogc:Filter>
            <ogc:classifiedAs>
                <ogc:PropertyName>/ExtrinsicObject</ogc:PropertName>
                <ogc:Literal>
                    http://www.example.com/Ontology#Quarry
                 </ogc:Literal>
            </ogc:classifiedAs>
        </ogc:Filter>
        </csw:Constraint>
    </csw:Query>
</csw:GetRecords>
```

**Listing 18 - Example for ClassifiedAs Operator**

**Usage in WFS:**

OGC Filters can be used when querying a Web Feature Service to search for features that match a certain concept. This implies that features have been semantically tagged, as described in section 2.1.2 (p.28) and 3.3.1 (p.44). Assuming features are properly annotated using the feature type schema proposed in 3.3.2 (p.45), this filter would return all quarries of the type ChalkPit or any subclass of it:

```
<wfs:GetFeature service="WFS" version="1.0.0" […] >
  <wfs:Query typeName="quarries">
    <ogc:Filter>
      <ogc:classifiedAs>
        <ogc:PropertyName>Type</ogc:PropertyName>
        <ogc:Literal>
          http://www.example.com/Ontology#ChalkPit
        </ogc:Literal>
      </ogc:classifiedAs>
   </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

**Listing 19 - Using the modelReference in a WFS query**

# 4. Summary

The need for better thematic descriptions of OGC Web Services is evident, and we believe that linking OWS and other OGC-compliant resources to ontologies with the help of semantic annotations can help to address most of the problems identified in the first section. We introduced three different levels where the annotations can be applied: on the resource metadata level (the Capabilities document for OWS), the data model and processes, and the actual data entities.

The previous sections showed that most existing standards already provide sufficient means for simple semantic annotations. Agreeing on harmonized best practices (such as the many proposed here) would be a good step forward to offer good semantic annotation capabilities throughout the OGC stack. From what is observed in the existing standards, it should be noted though that the need for semantic annotations has already been recognized at several levels and in several domains (SensorML is a good example). But in the end no harmonized effort was conducted, which resulted in mixed and partial solutions. Perhaps such a harmonizing effort, preferably materialized as a definition of a unique *annotation* element in OGC core schemas (similar to the `gmx:Anchor` element ?), should therefore be the next step on the quest to semantically enable OGC Web Services.

# 5. References and further Readings

F. Baader (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

D . Calvanese, De Giacomo, G., and Lenzerini, M. (1998). On the decidability of query containment under constraints. In Principles on Database Systems 1998 (PODS'98), pages 149-158.

T.R. Gruber (1995). Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.,* 43(5-6):907-928.

J. Hoffman, N. Steinmetz, and D. Fitzner (2008): Deliverable 2.4 - Semantic Web Geoprocessing Services. SWING Project Deliverable.
Url: http://swing-project.org/deliverables/document/110

E. Klien, S. Schade, and J. Hoffmann (2007): Deliverable 3.1 - Ontologies in the SWING Application - Requirement Specification
Url: http://swing-project.org/deliverables/document/86

M. Lutz (2007). Ontology-based descriptions for semantic discovery and composition of geoprocessing services. *GeoInformatica*, 11(1):1-36.

P. Maué, M. Braun, and H. Michels (2008): Meilenstein 5.04 - Anforderungsanalyse zur Dienste-Annotation in den Szenarien . GDI Grid Project Deliverable.

P. Maué (2008). An extensible semantic catalogue for geospatial web services. *International Journal of Spatial Data Infrastructures Research*, 3.

S. Schade, E. Klien, P. Maué, D. Fitzner, and W .Kuhn (2008): Deliverable 3.2 - Report on Modelling Approach and Guideline. SWING Project Deliverable.
Url: http://swing-project.org/deliverables/document/86

M. Zaremski (1996). Signature and Specification Matching. School of Computer Science, Carnegie Mellon University. PhD Thesis.

# Appendix A - Glossary

- **Annotation:** Attaching descriptive metadata to a resource, with the intention to simplify its discovery and evaluation.
- **Data Model:** Describes the structure of geospatial data, usually in GML schema. A data model defines how the → **data entities** have to represented.
- **Data Entity:** A specific instance of data. Adheres to the structure defined in the → **Data Model**.
- **Domain Ontology:** Domain-specific knowledge shared by one user community is usually captured in such a → **global ontology**.
- **Global Ontology**: Ontologies capturing knowledge which needs to be shared to other users are categorized as global ontologies. In contrary to → **local ontologies**, global ontologies are well maintained by information experts and are made accessible to everyone.
- **Knowledge Model**: Conceptualizations of knowledge, structured into concepts with unique identifiers and relations between the concepts. Ontologies are sophisticated forms of knowledge models.
- **Local Ontology:** refers to application specific knowledge models like the → **Resource Ontology**. Local Ontologies are not shared, and only the service provider is responsible for the creation and maintenance (in contrary to → **Global Ontology**).
- **Model Reference:** A link between an element in the schema-based metadata and a concept in the → **knowledge model.** The model reference is used to link to an entity with equivalent semantics but a different encoding.
- **Ontology**: Ontologies are the most sophisticated means to structure and specify knowledge, either on the local level as → **Resource Ontology** or shared as → **Domain Ontology**.
- **Reasoning**: Machine-supported inference to gain either new knowledge out from the existing ontologies or to solve semantic heterogeneities. Reasoning assumes to have the → **knowledge model** expressed in a language processable by the algorithm.
- **Resource Ontology:** Application-specific → **local ontology** describing the specific characteristics of one particular service or data set. The → **data model** of a WFS, for example, is fully represented in the Resource Ontology (as well as the process served by a WPS)
- **Semantic Annotation:** A specialized form of → **Annotation** where the attached metadata is partly expressed in a formal language, which makes interpretation by a machine possible. → **Reasoning** algorithms can then support the discovery and evaluation.
- **Taxomony**: A less structured → **knowledge model**, consisting of a vocabulary and one predefined relation between the terms.

# Appendix B - Semantic Annotations in a nutshell

In the following, we present a tentative list of the benefits, drawbacks, and potential applications for the proposed annotation at each of the three levels. This list provides a summary of our findings. This list is a first draft only, the authors expect from future discussions within the OGC additional ideas to further extend this list.

## Level 1: Source Metadata

**Advantages:**

- Easy to implement
- Easy to understand (also for non-experts)
- Helps to (quickly) assess what kind of data is served
- Needs no modification of underlying data schema
- Much better recall then no annotation at all (due to semantic-enabled query processing)
- Only solution for binary data like images (served by WCS or WMS)
- Good solution for WMS (since there is no real data schema anyway)

**Drawbacks:**

- Semantic-enabled discovery resources requires specialised interfaces letting users select the needed concepts
- Only week reasoning support if directly linked to domain concepts (via domain reference)
- Can impair readability of service metadata
- Consistency Problems: Changes in underlying data model requires changes of annotations on metadata level
- Not possible to identify what part of the served data represents the needed information

**Suggested Applications:**

- Service Discovery
- Service Evaluation

## Level 2: Data Models

**Advantages:**

- Higher precision and better recall since we can model the inner workings
- More details, results in higher recall
- We can identify which specific elements yield the requested information
- Better assessment possible, since one can better study the data model
- Less problems with consistency, since changes to the data model have direct impact on annotations

**Drawbacks:**

- Loosing the flexibility of Level 1 – annotations
- Describing complex data models tedious, requires additional documentation
- Semantic-enabled discovery resources requires specialised interfaces letting users select the needed concepts (but it does have no impact on the normal, keyword-based discovery)
- Annotations of processes (in a WPS) rather complex

**Suggested Applications:**

- Service Discovery
- Service Evaluation
- Workflow Validation

## Level 3: Data Entities

**Advantages:**

- Still lightweight approach, since we still just add a modelReference attribute to entities
- Flexible annotation of features
- Filtering using annotations possible
- Good solution for data without explicit data models, i.e. application schema

**Drawbacks:**

- Needs annotation tools in the Viewer application (Desktop GIS or Web mapping tools)
- Can be costly, due to the potential high number of entities
- Increased data volume

**Suggested Applications:**

- Resource Discovery (with filtering)
- Service Evaluation
- Quality Control for Resources
- Merging of Datasets

# Appendix C – Summary of annotated (meta)data standards

Below is a summary of the standards described in this document, each with a template of semantic annotation that highlights how to express the elements of an annotation, i.e. conceptURI, conceptLabel, theusarusURI, thesaurusLabel and language.

| | | |
|---|---|---|
| **ISO19115/ 19119** | <gmd:MD_Keywords><br><gmd:keyword><br><gmx:Anchor xlink:href="**{conceptURI}**"><br>**{conceptLabel}**<br></gmx:Anchor><br></gmd:keyword><br><gmd:thesaurusName><br><gmx:Anchor xlink:href="**{thesaurusURI}**"><br>**{thesaurusLabel}**<br></gmx:Anchor><br></gmd:thesaurusName><br></gmd:MD_Keywords> | |
| **OWS Capabilities** | <ows:Keywords><br><ows:Keyword>**{conceptURI}**</ows:Keyword><br><ows:Type codeSpace="**{thesaurusURI}**"><br>**{thesaurusLabel}**<br></ows:Type><br></ows:Keywords> | No provision for Concept label, language |
| **WMS Capabilities** | <KeywordList><br><Keyword vocabulary="**{thesaurusURI}**"><br>**{conceptURI}**<br></Keyword><br></KeywordList> | No provision for Concept label, thesaurusLabel, language |
| **KML** | <Placemark><br><ExtendedData><br><Data name="http://purl.org/net/concepts/domainReference"><br><value>**{conceptURI}**</value><br></Data><br></ExtendedData><br></Placemark> | Only conceptURI is expressed, just as any other property. |

| | | |
|---|---|---|
| **Context (WMC)** | <KeywordList><br><Keyword>**{conceptURI}**</Keyword><br><Keyword>**{conceptURI}**</Keyword><br></KeywordList> | Only conceptURI is expressed. |
| **Context (Atom-based)** | <category term=" **{conceptURI}** "<br>scheme=" **{thesaurusURI}** "<br>label=" **{conceptLabel}** "<br>xml:lang=" **{language}** " /> | No provision for thesaurusLabel |
| **SensorML** | <identification><br><IdentifierList><br><identifier name="longName"><br><Term definition="**{conceptURI}**"><br><value>**{conceptLabel}**</value><br></Term><br></identifier><br></IdentifierList><br></identification><br><classification><br><ClassifierList><br><classifier name="orbitType "><br><Term definition=" **{conceptURI}**"><br><codeSpace="**{thesaurusURI}**"/><br><value>**{conceptLabel}**</value><br></Term><br></classifier><br></ClassifierList><br></classification> | No provision for thesaurusLabel, language |